Material point method after 25 years: theory, implementation and applications*

Alban de Vaucorbeil[1,2], Vinh Phu Nguyen[†3], Sina Sinaie[4], and Jian Ying Wu[5]

[1]Department of Materials Science and Engineering, Monash University, Clayton 3800, VIC, Australia
[3]Department of Civil Engineering, Monash University, Clayton, Victoria 3800, Australia
[4]Melbourne eResearch Group, Computing & Information Systems, Melbourne School of Engineering, The University of Melbourne, VIC 3010, Australia
[5]State Key Laboratory of Subtropical Building Science, South China University of Technology, 510641 Guangzhou, China.
[2]Institute for Frontier Materials, Deakin University, Geelong, VIC, 3216, Australia

November 26, 2019

Contents

*Submitted to Advances in Applied Mechanics
†Corresponding author, phu.nguyen@monash.edu

Abstract

It has been 25 years since Sulsky and her co-workers developed the first version of the material point method (MPM): a quasi particle method to solve continuum mechanics problems. In the MPM, the continua are discretized by Lagrangian particles moving over a fixed Eulerian background grid. As a result, large deformation and contact can be treated effortlessly. Since then, many improved instances of the MPM have been developed and the MPM has found applications in many fields from geoengineering to movie industry. As the MPM has now been matured and a large body of literature on it exists, it is a good time to ponder and reflect on the developments of the method to date. To this end, this manuscript provides a concise introduction to the MPM, covering theory, implementation and applications. All the algorithms required to have a working MPM implementation for the simulations of solids, fluids and their interactions are provided. We have coded these algorithms in in-house open source programs and used them to study the performance of different MPM variants for large deformation solid mechanics problems. These problems exhibit large plastic deformation, fractures and contacts. Convergence of different MPMs (CPDI, GIMP, B-splines, Total Lagrangian MPM, improved MPMs) are studied. Furthermore, MPM formulations for fluids/gases and heat conduction are also covered. Potential areas for improvement on the method have been identified. The paper is the first review of the MPM and presents a state-of-the-art of the current MPM literature covering 339 references.

***Keywords*—** Material Point Method, solid mechanics, large deformation, contact, fracture, fluid mechanics, FSI

1 Introduction

Computer simulations have become a mainstream tool in virtually every engineering fields ranging from civil engineering, mechanical engineering, geo-engineering, biomedical engineering to material sciences. They are not only useful for problems that are too complex to be solved analytically, but also increasingly replacing real experiments which are costly and time consuming. Furthermore, they can provide us tremendous information at scales of space and time where experimental visualization is difficult or impossible.

Nowadays computer simulations are mainly carried out in the framework of the finite element method, abbreviated by FEM, which was developed back in 1943 [Courant, 1943]. Although the method is well established and widely used in both academia and industry, when it comes to problems that involves very large deformation, the FEM reaches its limits. It is because the FEM relies on the discretization of space into a finite number of elements which will become, for large deformation problems, so distorted that accuracy is lost or even worse the simulations could crash prematurely. While this problem can be resolved via remeshing, deriving an optimal mesh adaptation strategy is by no means an easy task. Meshless or meshfree methods (MMs) were developed to remedy these problems.

Meshless methods are so named as the space is discretized into a number of points, or particles, in which each point interacts with its neighboring points in a flexible manner: not via a rigid mesh as in the FEM. It should be noted that up to now, there is no unified framework for meshfree methods. This is reflected by the plethora of existing methods[1] in the literature. The oldest developed method is Smoothed Particle Hydrodynamics (SPH) introduced by Gingold and Monaghan [1977], Lucy [1977] which was used for modeling astrophysical phenomena without boundaries such as exploding stars and dust clouds. Next, the Generalized Finite Difference Method of Liszka and Orkisz [1980] was proposed followed by the Diffuse Element Method (DEM) by Nayroles et al. [1992], the Element Free Galerkin (EFG) by Belytschko et al. [1994]; the Material Point Method [Sulsky et al., 1994], the Reproducing Kernel Particle Method (RKPM) by Liu et al. [1995]; the $h-p$ cloud method [Duarte and Oden, 1996]; the Natural Element Method [Sukumar et al., 1998]; the Meshless Local Petrov Galerkin (MLPG) by Atluri and Zhu [1998]; the Maximum entropy [Arroyo and Ortiz, 2006]; the Particle Finite Element Method (PFEM) of Idelsohn et al. [2006], Sabel et al. [2014], the optimal transport meshfree method (OTM) of Li et al. [2010], just to name but the most popular meshfree methods. For a comprehensive consideration of MMs, we refer to various review articles, for instance Belytschko et al. [1996], Babuška et al. [2002], Nguyen et al. [2008], Chen et al. [2017], Doblaré et al. [2005] (focused on the applications of meshfree methods in biomechanics), and Jacquemin et al. [2019] discussing generalized finite difference and collocation methods. Some excellent textbooks on this topic have been written [Liu and Liu, 2003, Liu, 2002, Li and Liu, 2007, Fasshauer, 2007].

In recent years, it seems that the MPM has received a lot of attention in geotechnical engineering. Indeed, the method was featured in Alonso's 2017 Rankine Lecture and two international conferences on the MPM were organized by the Anura3D MPM Research Community in 2017 and 2019. Moreover, an increasing number of papers describes its use for various geotechnical problems, for instance see Fern et al. [2019]. See also Fig. 1 to see an increasing number of works on the method. The MPM has also received significant attention in computer graphics. It has been integrated into the production framework of Walt Disney Animation Studios and has been used in featured animations including Frozen, Big Hero 6 and Zootopia [Jiang et al., 2016]. Due to its popularity, one could anticipate that there would be many researchers who will adopt the MPM.

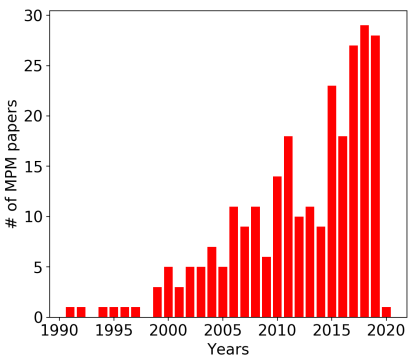


Figure 1: Number of papers on the MPM since the early 90s. The data contains 278 references including papers, Master and PhD theses. A Python script was used to create this plot based on a bib file.

[1]The wikipedia page on meshfree methods lists about 30 methods and new methods are being created, https://en.wikipedia.org/wiki/Meshfree_methods.

For people new to the MPM, a textbook such as that of Zhang et al. [2016] is undoubtedly valuable. Since this book is already three years old, we believe that a review would provide updated literature and be more accessible. Furthermore, an exhaustive examination of the MPM and its applications will not only provide an opportunity to re-examine previous contributions, but also re-organize them in a coherent fashion and anticipate new advancements. Finally, after more than two decades, a comprehensive review of the MPM is in order.

With that in mind, in this paper a unified presentation of all common variants of the MPM is presented. This review covers theory, implementation, applications and limitations of the method. More specifically, the following is presented:

- The standard MPM originally developed by Sulsky et al. [1995], the BSMPM [Steffen et al., 2008a], the Generalized Interpolation Material Point (GIMP) method [Bardenhagen and Kober, 2004], the Convected Particle Domain Interpolation (CPDI) of Sadeghirad et al. [2011], improved MPMs [Sulsky and Gong, 2016, Wobbes et al., 2019] and finally the Total Lagrangian MPM (TLMPM) of de Vaucorbeil et al. [2019];
- Implementation of all the above MPM variants with and without contacts;
- MPM formulations employing either a Cartesian grid or an unstructured mesh;
- Basic implementation of weakly compressible fluids and gases;
- Basic implementation of a coupled thermo-mechanical MPM;
- Convergence analysis of the MPM;
- A state-of-the-art literature review of the MPM covering 339 references up to late 2019.

All the points discussed here are based on our own results, obtained using various in-house codes but mostly with `Karamelo`, our 3D parallel open-source MPM code. Equations and algorithms necessary to reproduce them are clearly presented. Futhermore, all figures are original and some results are new.

As many other codes exist, one of the most popular being Uintah MPM [Parker et al., 2006], why yet another code? We believe that to fully understand a numerical method, one has to get their hands dirty. It is fun. Other codes are not well documented and thus requiring a steep learning curve for new users, particularly when one has to modify them.

`Karamelo` is a fully modular, 1D, 2D, and 3D parallel code written in C++. Its structure is based on that of the popular molecular dynamics simulator LAMMPS [Plimpton, 1995]. `Karamelo` is at the forefront of the continuum mechanics simulation work done at the Institute of Frontier Materials at Deakin University as well as at the Departments of Civil Engineering and Materials Science and Engineering at Monash University, Australia. The code is made open source to ensure full reproducibility of all the presented simulation results.

We present a state-of-the-art literature review of the MPM covering 339 references, of which more than 278 papers are directly related to the MPM, up to late 2019. It is obvious that our list of references is not exhaustive and absence of relevant items are due only to our focused idiosyncrasy and negligence. Due to our lack of expertise in geo-technical engineering, MPM algorithms specific to this field are not discussed and we refer to Fern et al. [2019] for details. To prevent this paper from being a book, details of some interesting topics have been omitted, including implicit dynamics MPM, quasi-static MPM solver, fluid-structure interaction etc. However, readers are directed to the relevant references for more details.

Our journey starts with this introduction which is structured as follows. First the basic idea of the MPM is given in Section 1.1. What are common MPM variants? How the MPM handles contacts and fracture? What are the sources of errors? Then, in Section 1.2, applications of the MPM in different areas are discussed. Section 1.3 summarizes the literature on mathematical analysis of the MPM algorithms. Some useful resources on the MPM, including open source and commercial codes are discussed in Section 1.4. Finally, the layout of the remaining of this paper is then stated in Section 1.5 followed by a discussion on notations in Section 1.6.

1.1 A brief introduction to the MPM

The Material Point Method is one of the latest developments in particle-in-cell (PIC) methods. The first PIC technique was developed in the early 1950s by Harlow [1964, 2004] at Los Alamos National Laboratory and was used primarily in fluid mechanics. The first PICs suffered from excessive energy dissipation which was overcome in 1986, by Brackbill and Ruppel with the introduction of FLIP-the Fluid Implicit Particle method [Brackbill and Ruppel, 1986, Brackbill et al., 1988]. In computer graphics, PIC/FLIP has become the de facto standard method for fluid simulations [Zhu and Bridson, 2005]. The FLIP was later modified and tailored for applications in solid mechanics by Sulsky and her co-workers [Sulsky et al., 1994, 1995] at University of New Mexico and has since been referred to as the Material Point Method [Sulsky and Schreyer, 1996].

In FLIP, the strain and stresses are stored at the cell centers. Yet, in the MPM, they are carried by the particles themselves. Thus, the MPM particles carry the full physical state of the material including position, mass, velocity, volume, stress, temperature etc. Note that in PIC, the particles carry only position and mass.

The MPM is built on the two main concepts already used in PIC that are the use of Lagrangian material points that carry physical information, and a background Eulerian grid used for the discretization of continuous fields (i.e., displacement field). For a short description of the Lagrangian and Eulerian descriptions, see Fig. 2.

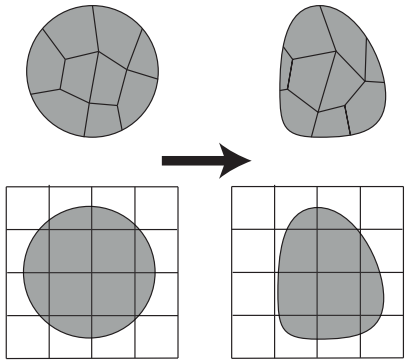


Figure 2: Lagrangian description (top) versus Eulerian description (bottom). In a Lagrangian description, the grid is attached to the solid and thus it deforms during the deformation process of the solid. Each point in the grid is always associated to just one single material point, thus making modeling history-dependent materials easy. The solid boundary is also well defined. However, the grid can become distorted. On the other hand, the Eulerian grid is fixed in space and material flows through the mesh. Mesh distortion never happens.

1.1.1 Lagrangian particles and Eulerian grid

In the MPM, a continuum body is discretized by a finite set of n_p Lagrangian material points (or particles) that are tracked throughout the deformation process. The terms *particle* and *material point* will be used interchangeably throughout this paper. In the original MPM, the subregions represented by the particles are not explicitly defined. Only their mass and volume are tracked. In advanced MPM formulations such as GIMP or CPDI, the shape of these subregions is tracked though. Each material point has an associated position $\mathbf{x}_p^t$ $(p = 1, 2, \ldots, n_p)$, mass m_p, density ρ_p, velocity $\mathbf{v}_p$, deformation gradient $\mathbf{F}_p$, Cauchy stress tensor $\boldsymbol{\sigma}_p$, temperature T_p, and any other internal state variables necessary for the constitutive model. Collectively, these material points provide a Lagrangian description of the continuum body. Since each material point contains a fixed amount of mass at all time, mass conservation is automatically satisfied.

The original MPM developed by Sulsky is effectively an updated Lagrangian scheme. For this MPM, the space that the simulated body occupies and will occupy during deformation is discretized by a grid, called background grid where the equation of balance of momentum is solved. On the other hand, in the Total Lagrangian MPM [de Vaucorbeil et al., 2019], the background grid covers only the space occupied by the body in its reference configuration. We refer to Fig. 3 for a graphical illustration of material points overlaying on a Cartesian grid for both ULMPM and TLMPM. The use of a grid allows the method to be quite scalable by eliminating the need for directly computing particle-particle interactions. The particles interact with other particles in the same body, with other solid bodies, or with fluids through a background Eulerian grid. Most often, a fixed regular Cartesian grid is used throughout the simulation for efficiency reasons.

Remark 1 *After submission of this paper, we got to know the so-called Discontinuous Galerkin MPM (DGMPM) presented in Renaud et al. [2018, 2019]. This DGMPM also adopts a Total Lagrangian formulation. Even though the DGMPM has many promising features (close to finite volume methods, easy for mesh adaptation etc.), it needs to be tested for more challenging problems to see the true benefits compared with other MPM variants.*

1.1.2 The basic MPM algorithm

The MPM was originally developed to solve fast transient impact solid mechanics problems [Sulsky et al., 1994]. Therefore, the MPM has been developed using an explicit solver as for such problems they are more efficient than implicit ones. The method was then applied to other applications in which the loading rates are low. For these problems, implicit solvers are more suitable. Since the explicit MPM algorithm is simpler than the implicit, in what follows, the updated Lagrangian MPM

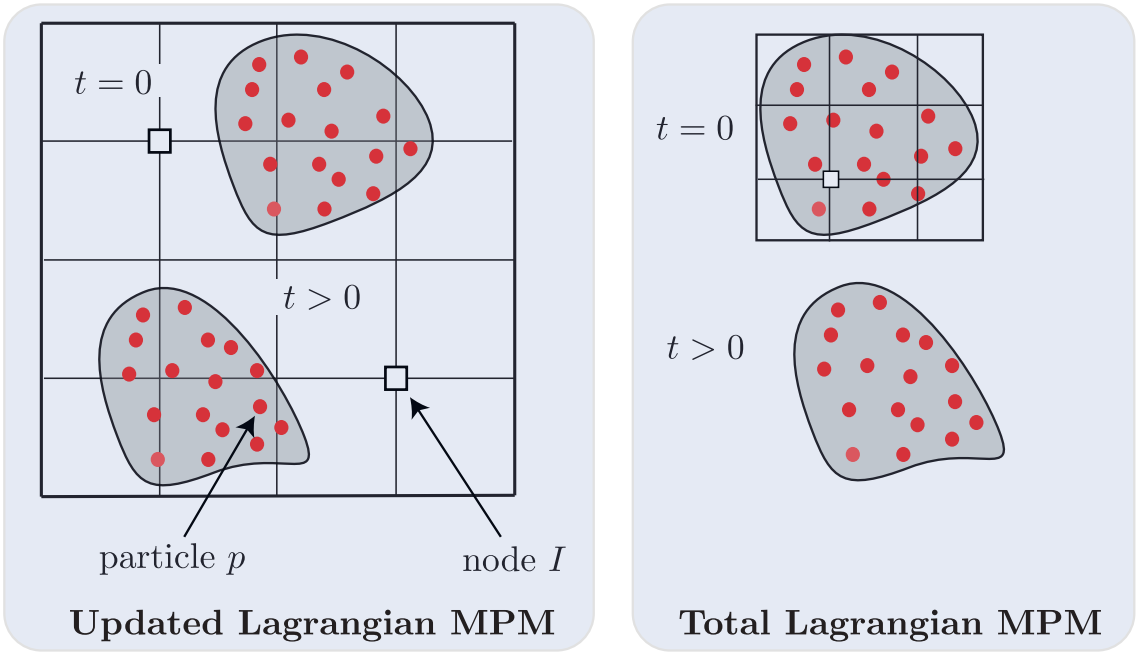


Figure 3: The MPM discretization: the space is discretized by a background grid which can be either a Cartesian grid or an unstructured grid (not shown), while a solid is discretized using particles. The updated Lagrangian MPM grid covers the entire deformation space whereas the total Lagrangian MPM grid only covers the initial configuration.

algorithm is presented using an explicit solver. Implicit MPM formulations are discussed in Remark 18. From the updated Lagrangian MPM, the total Lagrangian MPM is obtained by making only slight modifications.

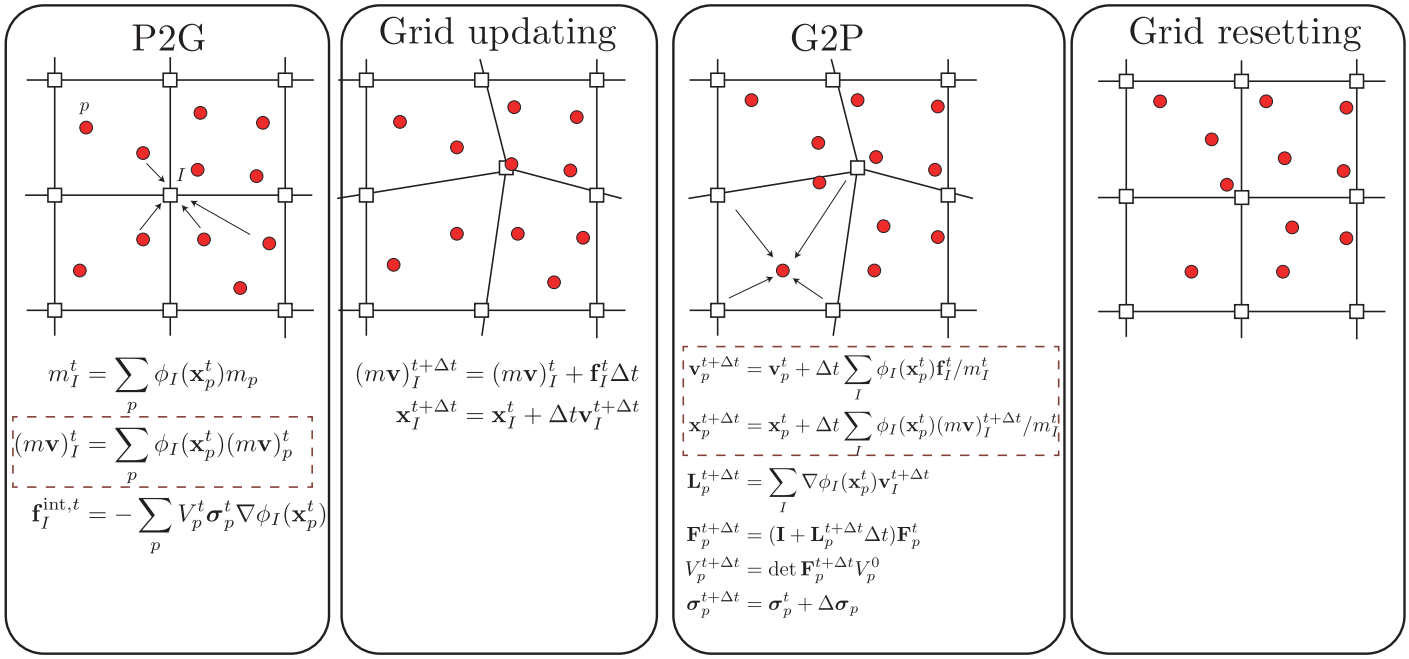


Figure 4: Material point method: a computational step consists of four steps: (1) P2G (Particle to Grid) in which information is mapped from particles to nodes, (2) Grid Updating in which momentum equations are solved for the nodes, (3) G2P (Grid to Particles) where the updated nodes are then mapped back to the particles to update their positions and velocities and (4) Grid resetting where the grid is reset. The operations in dashed boxes are not present in the ULFEM.

A typical explicit ULMPM computational cycle is given in Fig. 4. We refer to Table 1 for a list (not exhaustive) of notations and Table 3 for abbreviations. This algorithm is somewhat premature in the sense that some terms have not yet been precisely defined, but it is presented at this stage to give some perspective. The first step is mapping information from the particles to the grid (P2G) since the grid is reset at every cycle. Next, the discrete equations of momentum are solved on

the grid nodes (Grid updating). Then, the particles' position, velocity, volume, density, deformation gradient, stresses and all relevant internal variables are updated (G2P). These last two steps are equivalent to updated Lagrangian FEM. Therefore, it is incorrect to state that the MPM uses a Eulerian kernel as in Gupta et al. [2011]. Finally, the grid is reset to its original state. Due to this grid resetting mesh distortion never occurs making MPM a good method for large deformation problems. Note that the grid needs not to be reset at every time step. For instance, reset can be done every N time steps. N could be as large as one wants. The grid can also never be reset [Guilkey et al., 2006]. Moreover, a completely new grid can be used. But to the best of our knowledge, this is not yet implemented in any code.

It is quite difficult to precisely categorize the MPM due to the combination of Lagrangian particles and a background grid. In our view, since the MPM solves the momentum equations in their weak form, it can be seen as a Galerkin meshfree method, similar to EFG, the RKPM, the OTM. What differentiates the MPM from other Galerkin MMs is the ease with which the shape functions are constructed. Indeed, they are simple and efficient polynomials defined on a fixed Eulerian grid. Note that most of meshfree shape functions are computationally expensive rational functions defined on a cloud of nodes. When the grid is not fixed, the MPM is very similar to the OTM (or vice versa). The difference being that the OTM adopts the max-ent approximation [Iaconeta et al., 2017]. Contrary to OTM, the MPM uses a background grid which if not fixed would generate mesh entanglement problems, similarly to updated Lagrangian FEM.

1.1.3 Advantages and disadvantages of the MPM

Advantages of the MPM include:

- the absence of mesh-entanglement problems;
- error-free advection of material properties via the motion of the material points;
- a no-slip, no-penetration contact algorithm is automatic to the method. That is, it comes at no additional computational expense;
- a straightforward and efficient treatment of frictional contacts of multi-bodies thanks to the background Eulerian grid;
- suitability for image-based simulations, as it is easy to convert images into an MPM model;
- an easy computer implementation of the MPM compared to existing meshfree methods. The MPM algorithm is easily programmed for parallel, distributed-memory computers through decomposition of the computational domain;
- the leverage of existing well-studied FEM algorithms due to the similarity of the MPM with the FEM.

In addition to these advantages that the MPM offers, as with any numerical method, it has its own set of shortcomings:

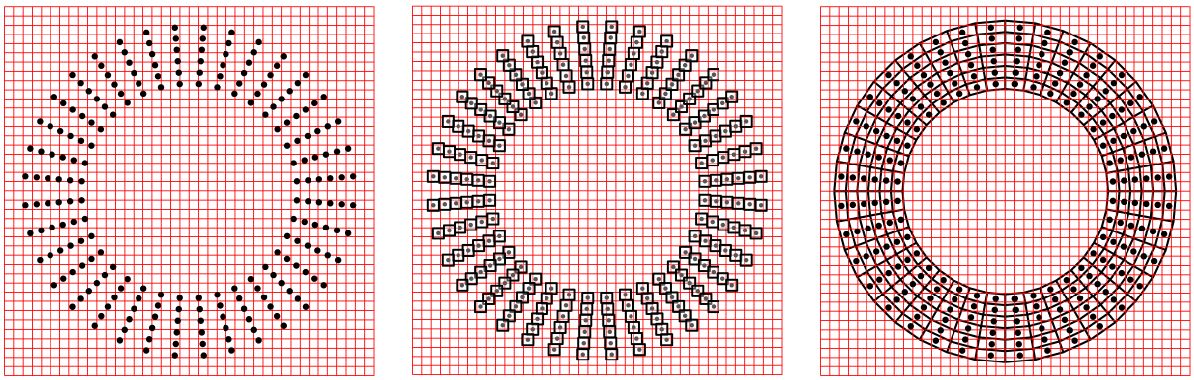
- large memory footprint since the grid has to cover the entire region that the bodies occupy;
- formal analysis (convergence, error and stability) of the MPM is extremely difficult;
- enforcement of boundary conditions is difficult compared with the FEM;
- lower accuracy than the FEM as the material points do not generally lie at the optimal positions for numerical integration.

The first item applies only to the ULMPM not the TLMPM since in the later the grid covers only the initial undeformed configuration. The formal analysis of the MPM is extremely difficult due to the irregular distribution of the particles but also due to their relative motion with respect to the grid. If such an analysis is to be carried out, many assumptions are required to make the analysis manageable i.e., 1D, linear elastic materials, particles do not move from one cell to another [York et al., 1999]. The difficult enforcement of boundary conditions is due to the lack of an explicit representation of boundaries. But, it is still easier than some other methods (e.g., SPH) [Raymond et al., 2018]. Finally, the low accuracy only applies to small and moderately large deformation problems.

1.1.4 Existing MPM formulations

The main differences between MPM variants emerge from the use of (i) different grid basis functions, (ii) different time integration schemes, and (iii) different types of the grid (Cartesian or unstructured). The discussion herein limits to the ULMPM.

Use of different grid basis functions. In the original MPM, dubbed herein as the standard/conventional MPM, the grid basis functions $\phi_I(\mathbf{x})$ are linear hat functions. Because these functions are only C^0, the standard MPM suffers from the so-called cell-crossing instability when particles cross the cell boundaries. In an attempt to remedy this issue, Bardenhagen and Kober [2004] introduced the generalized interpolation material point (GIMP) method. In GIMP, contrary to the original MPM, particles are not point like, but rather have finite extent, see Fig. 5. The resulting grid basis functions of GIMP are C^1 smooth functions which resemble B-splines functions used by Steffen et al. [2008c]. The MPM with B-splines, named BSMPM, is now quite popular [Stomakhin et al., 2014a, Tielen et al., 2017, Gan et al., 2018].



(a) MPM (b) GIMP (c) CPDI

Figure 5: Particles in the standard MPM (MPM with hat and B-splines functions), GIMP and CPDI.

Another issue related to particles moving from one cell to another is numerical fracture (unphysical separation of the solid). This happens when two originally adjacent particles are separated by a distance high enough such that they no longer interact with each other. This distance depends on the type of grid function used.

Even though in GIMP, particle domains are tracked to some extent, gaps between them remain leading to low accuracy under arbitrary deformations. The Convected Particle Domain Interpolation (CPDI), the latest development in GIMP, proposed in Sadeghirad et al. [2011, 2013] solves this problem. In CPDI, particles are modeled as quadrilaterals and tetrahedrons, in 2D and 3D, respectively. Therefore, the deformed solid is tiled without gap for arbitrary loadings, see Fig. 5c. CPDI is very good at handling extreme tensile deformations without exhibiting numerical fracture and is able to faithfully represent complex geometries. However, it is not exempt from issues. First, CPDI suffers from mesh-distortion problems that go against the spirit of the conventional MPM, see Wang et al. [2019]. Second, the solid must be meshed to prevent gaps. Again, this goes against one of benefits of MMs compared to the FEM: the discretization of a solid as a set of points is much faster and less complex than the meshing process in the FEM. Third, parallelization is more complicated in CPDI [Homel et al., 2016] as particles might have a domain of influence large enough to span over many different CPU domains.

Use of different time integration schemes. The most common implementation of the MPM uses an explicit integration scheme, but implicit MPM also exists. Explicit time integration schemes are easy to implement, efficient and stable for short-duration dynamic problems. On the other hand, for low strain rate and quasi-static problems e.g., manufacturing problems like metal rolling, upsetting, and machining, it is more efficient to use an implicit time integration scheme. In an implicit MPM, one can either form explicitly the Jacobian matrix or adopt a matrix-free solver. In the following, we review the implicit MPM for both dynamics and quasi-static problems.

The first to use an implicit scheme for the MPM were Cummins and Brackbill [2002]. They applied it to the simulation of quasi-static loading of granular materials. Implicit integration schemes are known to be computationally expensive. Therefore, to reduce computational time (by avoiding the construction of the tangent matrix), they adopted the matrix-free Newton-Krylov algorithm. Sulsky and Kaul [2004] reported a similar method and Love and Sulsky [2006a,b] extended it for hyperelastic-plastic materials. These papers adopted a three-field formulation to avoid volumetric locking (Section 5.2 provides a discussion on volumetric locking). Conservation of linear and angular momentum of the MPM was discussed in detail in these references. Nair and Roy [2012] implemented this matrix-free implicit dynamics for GIMP. This matrix-free approach was extended to quasi-static problems in Sanchez et al. [2015] where it showcased the advantage that a consistent

material tangent, which is usually hard to obtain for complex constitutive models, is not required.

In contrast, Guilkey and Weiss [2003], Wang et al. [2016] explicitly formed the tangent stiffness matrix and used the Newton-Raphson method together with the well known Newmark integration scheme to solve the equilibrium equations in time. They reported that time steps hundreds of times larger than those used in explicit MPMs. Moreover, the use of a consistent tangent allows time steps to be much larger than those for the matrix-free formulation of Cummins and Brackbill [2002], Sulsky and Kaul [2004]. We refer to Iaconeta et al. [2017] for a detailed presentation of the algorithm of Guilkey and Weiss [2003].

Cartesian grid versus unstructured grid. In the MPM a uniform Cartesian grid is usually used. This eliminates the need for computationally expensive neighborhood searches during particle-mesh interaction (i.e., to which nodes a particle maps its data and vice versa). Furthermore, it is easier to develop smooth C^k ($k \geq 1$) basis functions with a Cartesian than with an unstructured grid.

On the other hand, in the geo-technical engineering community, unstructured grids are usually adopted. Wieçkowski et al. [1999], Wieçkowski [2004] were the first to use an unstructured grid for silo discharging applications. Since then, unstructured meshes have been used in later works of related research groups in University of Stuttgart, Germany and Delft University of Technology, The Netherlands e.g., Beuth et al. [2011], Jassim et al. [2013]. In contrast to a Cartesian grid, the search for which element contains a given particle in an unstructured mesh is not trivial and is very time-consuming. However, the use of an unstructured grid facilitates the enforcement of complex boundary conditions (i.e., boundary conditions on curved surfaces).

In a line of research parallel to that of GIMP and CPDI, Zhang et al. [2011] proposed the dual domain MPM (DDMPM) for unstructured grids. DDMPM was motivated by the difficulty to develop C^1 functions for an unstructured grid to mitigate the cell-crossing issue. The basic idea is to map the particle stresses to the grid nodes and then interpolate them to obtain a continuous stress field at any point of the domain. This dual mapping process makes a smoother gradient emerge. More recently de Koster et al. [2019] developed C^1 basis functions over unstructured grids using Powell-Sabin functions [Powell and Sabin, 1977]. We refer to Section 4.4 for a presentation of the MPM using unstructured grids.

High order MPMs. Even though impressive simulations have been done with the MPM and its previously presented variants, rigorous analyses of the method for simple problems show that the convergence rate is poor (rarely of second order) as explained further in Section 1.3. Aiming to improve the order of convergence, Wallstedt and Guilkey [2011] presented a weighted least square MPM for solids and Edwards and Bridson [2012] proposed a moving least square (MLS) MPM for fluids. An improved MPM (iMPM) was presented by Sulsky and Gong [2016]. In the iMPM, the hat functions are used for all the mapping except the velocity mapping from particle to node for which MLS is used. Additionally, one point quadrature is used i.e., the quadrature points are the cell centers. This removes cell-crossing instability. They demonstrated second-order convergence for iMPM, but only for moderately fined meshes. No convergence was observed for very fine meshes. Similar ideas can be found in Wobbes et al. [2019]. This high order of convergence were only demonstrated for 1D problems. Very recently, Liang et al. [2019] also use one-point quadrature but the cell center data are reconstructed, not via MLS, but with an extra staggered grid. A presentation of the iMPM is given in Section 5.5.

MPM variant	Efficiency	Quad. error	Cell crossing	Num. fracture	Grid type	Contacts
MPM	☺ ☺ ☺	☹ ☹ ☹	yes	yes	Cartesian/unstructured	☺
GIMP	☺ ☺	☹ ☹	no	yes	Cartesian	☺
CPDI	☺ ☺	☹	no	no	Cartesian	☺
TLMPM	☺ ☺ ☺ ☺	☹	no	no	Cartesian/unstructured	☹
iMPM	☺	☹	no	n/a	n/a	n/a

Table 1: Overall characteristics of common MPM variants.

Another stability issue of MPM is the so-called null space issue – the mapping of non-zero particle values can result in zero nodal values. This problem arises from the difference between the number of particles and the number of the nodal grid points. Null space issue might be the culprit to the non convergence of iMPMs when used with very fine meshes. Methods to solve this issue are presented in Gritton and Berzins [2017], Tran and Sołowski [2019].

A table presenting the characteristics of common MPM variants is given in Table 1. Note that even though the iMPM was applied to the standard MPM only, it can be used with other MPM formulations.

1.1.5 Multiphysics MPM

Although the MPM was originally developed for mechanical problems, it also has been extended to the simulation of multi-physics problems. Such simulations involve solving a coupled system of more than one partial differential equations. For example, for thermo-mechanical processes, the equation of motion is coupled with the energy balance equation (or heat diffusion equation) as presented in Chen et al. [2008], Nairn and Guilkey [2015], Fagan et al. [2016], Tao et al. [2016], Gritton et al. [2017], Tao et al. [2018], Leroch et al. [2018]. They show that when the explicit MPM solver is used, incorporating the heat diffusion is straightforward using an staggered solver. Among these works, Fagan et al. [2016] demonstrated that the MPM is able to simulate friction stir welding (FSW). FSW is a recent and complicated thermo-mechanical process used to join different materials which is still not well understood. And Gritton et al. [2017] reported first coupled chemical/mechanical MPM simulations of the deformation of a silicon anode. Since these are important developments in the MPM, the algorithms used to solve thermo-mechanical coupling is presented in Section 5.3.

1.1.6 Contacts

A contact happens when two deformable solids touch each other. It is a key element to understand many engineering problems such as pile-soil interaction, wear, metal forming processes, etc. The simulation of contacts remains challenging and various algorithms (e.g., note-to-segment contact, segment-to-segment contact, penalty and Lagrange multiplier methods) have been developed [Benson, 1992]. However, it is much easier in the MPM due to the use of a background grid. Indeed, it was developed mainly to handle contact problems.

A no-slip no-penetration contact is inherent in the MPM i.e., contact of solids is automatically handled without any extra numerical treatment. This is due to the use of a single-valued velocity field for updating the positions of the material points. This automatic no-slip contact ability of the MPM has been used in many works that involve complex contacts. For example, Bardenhagen et al. [2005], Brydon et al. [2005] used GIMP to simulate the compression of foam microstructures to full densification. This is a challenging problem which involves the combination of discretizing complex microstructures, simulating large deformations and multiple contacts. This work demonstrated that particle methods are well suited for the simulations of solids with complex geometries because body-fitted meshes are not needed. Nairn [2006][2] applied the MPM to study the transverse compression and densification of wood. Liu et al. [2015a] studies honeycomb sandwich panel subjected to high-velocity impact. And more recently Sinaie et al. [2019] simulated the large deformation response of cellular structures which involve many contacts.

In contrast, frictional contact between different bodies or self contact within a single solid requires a modification of the standard MPM algorithm.

Multi-body frictional contacts. The first the Coulomb frictional contact algorithm in the MPM was probably presented by Wieçkowski et al. [1999] to model between a deformable body and a rigid wall in silo discharging simulations. At the same time, York et al. [1999, 2000] observed that two bodies sometimes "stick" to one another unphysically when they should separate. To alleviate this problem, they proposed the first contact algorithm for the MPM that allows the contacting bodies to release from one another. But the contact is still no-slip. Bardenhagen et al. [2000, 2001] extended York's contact algorithm to frictional contact and used it to model interaction between grains in granular materials. Bardenhagen's algorithm, known as *multi-material* or *multi-body* contact, is very efficient as it is linear in the number of grains and allows separation, sliding and rolling. The algorithm's basic idea is to modify the velocities of contact nodes (i.e., those receive contributions from more than one material/body) to account for the collision. This algorithm is very popular. For instance, Nairn [2013] used it with modifications to model imperfect (debonding) interfaces. An improved version of Bardenhagen's algorithm was given in Lemiale et al. [2010] to model a metal forming process called the equal channel angular pressing technique. This work emphasized on the treatment of contact at interfaces between rigid and deformable bodies. All prior applications have been limited to Coulomb friction. Most recently, Nairn et al. [2018] generalizes the MPM approach for contact to handle any friction law with examples given for friction with adhesion or with a velocity-dependent coefficient of friction. The Bardenhagen's algorithm is presented in Section 5.1.

In the field of geotechnical engineering, a first contact algorithm for the quasi-static MPM was presented in Beuth [2012]. They adapted zero-thickness interface elements, commonly used in FEM, to model contact between soil and rigid surfaces. This technique applies only to the case where the contact surface is known prior the start of the simulation. Bardenhagen's algorithm was modified in Al-Kafaji [2013] for adhesive contact using an explicit dynamics MPM code. Yet another modification of Bardenhagen's contact model suitable for geotechnical engineering (that involves contact between stiff structural elements and soil) was presented in Ma et al. [2014].

[2]This work received the George Marra Award for paper of the year from the Society of Wood Science and Technology.

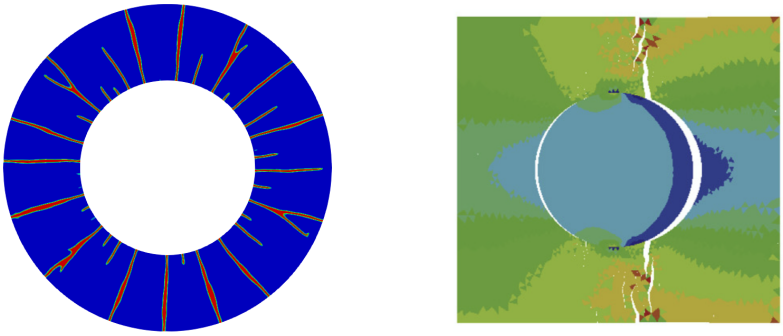
Frictional self contact. Homel and Herbold [2017] was the first to propose an algorithm for frictional self contact. Its basic idea is to automatically and dynamically detect contact nodes. This is achieved with a scalar field (actually its gradient) that defines potential contact surfaces. The same idea (named DFG) was applied to fracture and resulted in a first model for fracture and post-fracture frictional contacts i.e., contacts between crack faces.

1.1.7 Fracture

Fracture is defined as the separation of a solid into two or many pieces. This phenomenon has been extensively studied since the pioneering works of Griffith [1920], Irwin [1957]. They developed the field of fracture mechanics which has found tremendous applications in many engineering disciplines notably aerospace engineering.

Modeling the initiation and propagation of cracks in a solid is a challenging problem as one needs to track a set of evolving internal surfaces across which the displacement field is discontinuous. Basically, there are three approaches to modeling fracture: discontinuous, continuous and mixed continuous-discontinuous approaches [Rabczuk, 2013]. We also refer to the interesting article of Boyce et al. [2016] that presents a number of different techniques used by different research groups to simulate ductile fracture.

In the discontinuous approach, the cracks are explicitly represented (fracture mechanics theory). In the continuous approach, a crack is however not explicitly modeled (damage mechanics [Kachanov, 1958, Lemaitre and Chaboche, 1994]). In the later approach, the presence of cracks is treated via damage variables used to degrade the stresses. Fig. 6 presents fracture simulations using these two methods. The combined continuous-discontinuous approach to fracture combines the two approaches, as its name implies.



(a) Continuous approach (b) Discontinuous approach

Figure 6: Continuous versus discontinuous approaches to fracture modeling. In (a), fracture of a cylinder under internal pressure impulse is given. The red color denotes the damage variable close to 1 which corresponds to failed elements. In (b), cracking of a fiber-reinforced composite material is shown [Nguyen, 2014a].

Discontinuous approach. In the MPM, explicit cracks have been modeled as strong discontinuities by allowing *multiple velocity fields* at nodes whose supports are cut by the cracks. This is similar to duplicated nodes in FEM. The crack can be modeled using either the LEFM (Linear Elastic Fracture Mechanics) [Nairn, 2003, Tan and Nairn, 2002, Nairn, 2007a, Guo and Nairn, 2004, Gilabert et al., 2011, Wang et al., 2005] or cohesive zone models [Daphalapurkar et al., 2007, Nairn, 2007b] or a combination of both [Nairn, 2009, Bardenhagen et al., 2011]. Explicit fracture simulation is computationally expensive (as one needs to track evolving surfaces) and prohibitive for large scale simulations. Furthermore, the implementation is intricate particularly for complex crack patterns. The latest development in this direction is the work of Moutsanidis et al. [2019a] that introduces a single velocity field for cracks by modifying the grid basis functions.

Remark 2 *Although a majority of these works involve simulations under small deformation. We don't see any limitations for the discontinuous approach to be used for large deformations. We do not think that the MPM is better than the FEM for small (or moderately large) deformation fracture mechanics problems. This observation is backed up by the outcome of the first Sandia fracture challenge [Boyce et al., 2014] where the MPM was used by one team [Yang et al., 2014].*

Continuous approach. In the MPM, fracture has been modeled using various continuous methods. They all share one common feature: no need to explicitly represent the crack surfaces as in the discontinuous approach. For penetration problems, fracture is usually treated using a strain-based failure criterion and particle erosion. That is, a particle is set to be failed when

it satisfies a certain strain-based fracture condition. When that happens, its deviatoric stress is set to zero, but it remains part of the simulation, and its mass contributes to the overall inertia of the material. Thus mass conservation is enforced. Strain-based failure combined with particle erosion was also used in Ionescu et al. [2006] to study failure of soft tissues penetrated by a low-velocity projectile. They reported convergent results for fine grids, but no quantitative evidence was provided. An elastic-plastic material with particle erosion was employed in Huang et al. [2011] to model penetration of thin steel plates and perforation of thick aluminum plates. Sensitivity of the results with respect to grid size and number of particles were studied, and some quantities showed convergence. A constitutive model belongs to t his category is given in Appendix B.3 for some simulations presented later in Section 6.

For low strain rate and quasi-static problems, standard constitutive models are often used. For example, a softening Mohr-Coulomb plasticity model was used in Alonso and Zabala [2011] to model the failure of the Aznalcóllar dam. More recently, a softening Drucker-Prager plasticity coupled to the Grady–Kipp damage model was presented in Raymond et al. [2019] to model failure of aggregate materials. They reported mesh-convergent results even though no special treatment was done to deal with softening. Homel and Herbold [2017] adopted a Rankine damage model with linear strain-softening.

Also for static and low strain rate dynamic problems, Schreyer et al. [2002], Sulsky and Schreyer [2004] used a smeared crack model (they referred it as a decohesion model), see e.g., Rots et al. [1985], Rots [1991]. Chen et al. [2002] improved this model by using a strain-based damage diffusion equation combined with a tensile damage model. Similar works include Shen and Chen [2005], Shen [2009], Yang et al. [2012, 2014]. Implementation details of this decohesion model in the MPM are given only recently in Sanchez [2011] and he demonstrated the results are mesh biased. That is, accurate solutions are achieved only for cases in which the crack orientation coincides with the orientations of grid cell line.

In the original damage mechanics framework, damage is determined using local variables such as stress and strains. This has proven to create mesh-dependent softening. To alleviate this issue, non-local models can be used. By non-local, we mean that damage is determined using non-local variables such as averaged stress and strains in the vicinity. The idea is to introduce a length scale in the equation. Burghardt et al. [2012] were the first ones to implement a non-local plastic model in the MPM.

Non-local damage can also be simulated using a phase field damage/fracture (PFF) theory see e.g., Bourdin et al. [2008], Miehe et al. [2010], Wu [2017] and the most recent review of Wu et al. [2019]. PFF was implemented in an MPM code for the first time by Kakouris and Triantafyllou [2017b,a]. Unfortunately only problems involving small deformation without contact were demonstrated then. Recently, Cheon and Kim [2019] also presents a similar method. They use adaptive grid refinement and particle splitting to capture the gradients of the phase field. In the computer graphics community, a similar idea has been recently proposed in Wolper et al. [2019] with impressive fracture simulations typically seen in this community.

Continuous-discontinuous approach. The first continuous-discontinuous brittle fracture model in the MPM (precisely CPDI) is presented in Homel and Herbold [2017]. In this model, fracture is first treated with a continuum damage model and then a self-contact algorithm is applied to the cracked nodes to separate the materials on the two sides of the crack. The damage model is enhanced with random material properties to mitigate mesh bias and the fracture energy is scaled with the grid size. Post-failure contact between the faces of a crack is allowed using the DFG algorithm. Recently, Homel and Herbold [2018] applied their model for mesoscale simulations of porous materials.

Remark 3 *With the development of variational approaches to fracture (or phase-field fracture models), the modeling of the initiation and propagation of complex crack networks in solids on a fixed mesh with conventional continuum finite elements can be done. Therefore, the statement that meshfree methods are better suitable for fracture seems no longer convincing. Nevertheless, it is very attractive to have a method that can handle large deformation, contact, fracture and post-failure contacts between fragments. The MPM might be such a method even though more works must be done before this goal is achieved.*

1.1.8 Fluids and gases

Although the MPM was developed for applications in solid mechanics and its main application area is still solid mechanics, it has also been applied to fluid mechanics problems. This probably arises from the need to simulate fluid-structure interactions (FSI) using just the MPM – to avoid coupling the MPM solid solver with a fluid solver which is not an easy job. Herein, we only discuss MPM algorithms for fluids. For works where the MPM used for solids is coupled to a fluid solver, we refer to Section 1.2.2.

There are basically two ways to handle fluids (and gases) using the MPM. The first is the so-called weakly compressible MPM. This way is almost identical to the solid MPM solver except that a constitutive model for fluids is used. This is done by describing the relation between the fluid pressure and density using an artificial equation of state (EOS). This fluid MPM solver was first presented in York et al. [1999], and it has been used for gas dynamics problems in York et al. [2000], Tran et al. [2010], Ma et al. [2009a]. It has also been used for fluid flow problems [Li et al., 2014, Mast et al., 2012]. Using this

fluid MPM solver, comparative studies of SPH and the MPM for fluid mechanics are presented in Zhao et al. [2017], Vargas et al. [2018], Sun et al. [2018]. Overall, it has been shown that both MPM and SPH predictions are quite simular but the MPM computational time is smaller. This weakly compressible fluid/gas MPM solver is presented in Section 5.4.

There are two main issues with the weakly compressible fluid MPM. First, if using an explicit time solver, very small time step is needed owing to the need to use a very large bulk modulus. Second, pressure oscillation occurs. To solve these problems, the second way is the truly incompressible MPM. Stomakhin et al. [2014b] in the computer graphics community were the first to present this formulation. They used the Chorin's operator splitting method [Chorin, 1968]. In the engineering community, Zhang et al. [2017], Kularathna and Soga [2017] presented similar formulations.

1.1.9 The MPM versus other methods

Which meshfree method should I use? This is the constant question facing every person new to MMs every time they need to solve a large deformation problem, if they are not forced to use any particular method. It is rather certain that no one is able to answer this question and neither are we. Nevertheless, to them, herein we discuss the comparison between the FEM, SPH, the discrete element method (DEM) , Galerkin MMs, and the MPM based on the literature.

SPH is probably the most popular meshfree method as it has been used in engineering and computer graphics and incorporated into many commercial softwares such as AUTODYN, PAM-CRASH, LS-DYNA and ABAQUS. Even though not a continuum-based numerical method, the ability to deal with large deformation and fracture makes the DEM a very popular technique particularly in geo-technical engineering [Cundall and Strack, 1979, Scholtès and Donzé, 2012, Sinaie et al., 2018a].

The first comparative study of SPH and the MPM for hypervelocity impact problems was reported by Ma et al. [2009b,a]. For the simulations considered, they showed that the MPM is faster and more accurate than SPH (precisely SPH in LS-DYNA). This can be explained by the fact that, the critical time step size in the MPM depends on the cell size of the background grid, rather than the particle space as in SPH, so that the time step used in the MPM is much larger than that of SPH. Furthermore, there is no neighboring particles search which is very time consuming. Note, however, that this conclusion is rather superficial as it was based on only a few numerical tests. It is certain that a working MPM simulation requires less *tricky ad hoc* parameters than SPH. A typical MPM simulation requires only data such as mesh spacing and time steps in the same manner with the FEM.

Interestingly, explicit MPM is faster than LS-DYNA explicit FEM for the well known Taylor impact problem, see Ma et al. [2009a]. This superior efficiency was also reported in Leroch et al. [2018] for micro-milling simulations. The reason is the same: the critical time step size in the MPM is very much larger than the one in the FEM as the FEM mesh deforms and the element sizes become very small.

Within the context of implicit dynamics, Iaconeta et al. [2017] carried out a comparative study of the standard MPM and the so-called Galerkin meshfree method (GMM) proposed in Espluga [2014]. GMM is very similar to the OTM of Li et al. [2010], except that a forward Euler is used for time integration. The authors find that the GMM using the max-ent and MLS lack robustness for very large deformation problems. This lack of robustness is due to the meshfree nature of these shape functions i.e., one has to ensure that there are always sufficient nodes surrounding a given material point. A stabilized OTM was presented in Weißenfels and Wriggers [2018].

Remark 4 *Recent works combine the MPM and SPH. For example, Raymond et al. [2018] coupled SPH to an MPM. They used SPH in the bulk and MPM around the surfaces for an easy enforcement of boundary conditions. He et al. [2019] developed the so-called SMPM where the particle velocities and stresses are smoothed using the SPH functions. The SMPM is shown, for impact simulations, to perform better than the traditional MPM and SPH and more efficient than the latter. See also He and Chen [2019] for an application of the SMPM to strain localization.*

For granular flow, some studies on the performance of DEM and MPM have been reported in Coetzee et al. [2007], Coetzee [2003], Ceccato et al. [2018], Gracia et al. [2019]. Basically, MPM can capture DEM accuracy if a proper constitutive model is adopted. One might argue that MPM should be used for large scale simulations using constitutive models derived from DEM simulations. In Dunatunga and Kamrin [2015], a constitutive framework for granular media was presented that is able to simulate in one setting a wide range of granular behaviors spanning several phases: solid-like static behavior, plastic flow (up to very large strains), as well as separation and reconsolidation of the material.

1.1.10 Coupling the MPM with other methods

Since no method is without shortcomings, it is natural to want to couple two (or even more than two) methods together to take advantages of their best features. Herein, we review works that present the coupling of the MPM with other numerical

methods be it the FEM or molecular dynamics. Section 1.2.2 discusses the coupling of the MPM with other methods for fluid-structure interaction problems.

It is reasonable to employ the efficient and accurate FEM in regions where the deformation is moderate and a particle method such as SPH or the MPM in domains featuring high deformation. Zhang et al. [2006] developed an explicit material point finite element method (MPFEM) to take advantages of both FEM and the MPM. The basic ideas is that the solid is discretized by a mesh of finite elements, and a computational grid is additionally predefined in the potential large deformation zone. The nodes covered by the grid are treated as MPM particles, and the remaining nodes are treated as FE nodes. The MPFEM was improved later by Lian et al. [2011a, 2012], Chen et al. [2015]. These algorithms are not presented here, and we refer to the textbook of Zhang et al. [2016] for details. Results reported in these papers show that the coupled MPFEM is indeed faster than the pure MPM. These work provided another respective for the CPDI, see Appendix C for details.

The MPM has been coupled with molecular dynamics to achieve a multiscale atomistic-to-continuum method [Ayton et al., 2001, Lu et al., 2006, Chen et al., 2012, Liu et al., 2013]. It is the particle nature of MPM that makes the coupling straightforward. As we lack experiences in this area, our discussion is merely for completeness. We refer the reader to the textbook of Zhang et al. [2016] for details.

1.2 Applications

The MPM has been applied to a wide range of problems involving solids, fluids and gases undergoing very large deformation. This section attempts to present an extensive overview of MPM applications. The idea is to give an outline of the types of simulation that can be done with the MPM. By no means it is exhaustive. Therefore, we apologize to those authors whose works were not mentioned here due to our negligence.

This section is organized as follows. Section 1.2.1 presents works done in the geo-technical engineering field. Fluid-structure interactions are given in Section 1.2.2. Image-based simulations are discussed in Section 1.2.3. Section 1.2.4 covers the works done by computer graphics researchers. Finally, Section 1.2.5 summarizes important works from other fields. As contact and fracture problems were detailed earlier (see Sections 1.1.6 and 1.1.7), they are not covered specifically in this section.

1.2.1 Large strain geo-technical engineering

The MPM has been adopted in large strain geo-technical engineering problems including landslide [Andersen and Andersen, 2010b, Llano-Serna et al., 2016, Soga et al., 2015, Yerro et al., 2018], silo discharging [Wieçkowski et al., 1999, Wieçkowski, 2004, Mühlhaus et al., 2001], anchor pull-out [Coetzee et al., 2005], excavator bucket filling [Coetzee et al., 2007], pile driving [Lim et al., 2013, Nguyen et al., 2016, Galavi et al., 2017], problem of subsidence of landfill covers [Zhou et al., 1999], shaped-charge jet penetration in wellbore completion [Burghardt et al., 2010, Homel et al., 2015] and installation of geosynthetics materials [Hamad et al., 2015]. The MPM was also used to model large deformation of saturated porous media, see for instance Zhang et al. [2009], Beuth et al. [2011], Jassim et al. [2013], Zheng et al. [2013], Abe et al. [2014], Ma et al. [2014], Yerro et al. [2015], Pinyol et al. [2017]. More recently, the MPM has been successfully used in the field of terramechanics [Agarwal et al., 2019] to simulate the interaction of rigid wheels with dry granular media. We refer to the book of Fern et al. [2019] for a more complete coverage of MPM applications in geo-engineering.

All the above works adopted the conventional theory of continuum mechanics to the exception of Mühlhaus et al. [2001] who used the Cosserat continuum framework in the MPM. The Cosserat theory is different as it also involves rotational velocities and the balance of angular momentum results in a non-symmetrical Cauchy stress tensor.

Remark 5 *People who are familiar with FLAC (Fast Lagrangian Analysis of Continua)[3] developed their own MPM formulations. For example Konagai and Johansson [2001] made the Lagrangian Particle finite difference method (LPFDM) for geomechanics. Differences with the standard MPM lie in the fact that stresses are smeared over the background grid i.e., internal forces are computed using the averaged stress tensor[4] stored at the element center.*

1.2.2 Fluid-Structure Interaction

Fluid-structure interaction (FSI) is the interaction of a rigid or deformable structure with an internal or surrounding fluid flow. Fluid-structure interactions are a crucial consideration in the design of many engineering systems, e.g. aircraft, spacecraft,

[3]FLAC is a two-dimensional explicit finite difference program for engineering mechanics computation, and a product of Itasca: http://www.itascacg.com/software/flac.

[4]Averaged means a volume weighted sum of particle stresses.

engines and bridges. Two main approaches exist for the simulation of fluid–structure interaction problems: (1) the monolithic approach where the equations governing the flow and the displacement of the structure are solved simultaneously – with a single solver – and (2) the partitioned approach where the equations governing the flow and the displacement of the structure are solved separately – with two distinct solvers. We refer to Bazilevs and Takizawa [2017] for a comprehensive account on this very challenging topic. The following of this section reviews works on FSI using the MPM.

Monolithic FSI MPM. The first FSI using the MPM was presented in York et al. [1999, 2000]. The structures studied in this work are two dimensional membranes. In this formulation, a membrane is represented by a set of material points and the fluids/gases by a another set of particles, see Fig. 7a. Both types of particles interact with each other via the background grid. The constitutive model for the membranes is such that they only sustains axial stress. The algorithm was applied to airbag impact simulations and the results were found to be in good agreement with experiments.

This model was improved later by other researchers e.g., Gan et al. [2011], Lian et al. [2011c, 2014], Nguyen et al. [2017]. Particularly, Lian et al. [2011c] treat the membrane (actually reinforcement bars) as 1D two-node bar elements. They now connect the membrane particles together, see Fig. 7b. By doing this, the number of particles necessary to discretize the membrane is significantly reduced [Nguyen et al., 2017]. In this model, the membrane is discretized by a set of two-node bar finite elements. For these elements, their mass and their internal forces are calculated the FEM way. Then, instead of solving the movement of these elements' nodes the FEM way, the mass and internal forces are projected onto the background grid. Then, the motion of the membrane particles and the fluid particles are updated the standard MPM way. We refer to Appendix C for details.

Based on the work of Lian et al. [2011c], Hamad et al. [2015] developed a new 3D solid-membrane coupling method. It is essentially a coupling of an MPM for the solid and a FEM (three-node triangular elements) for the membrane. The method was applied to simulate the installation process and the behavior of geosynthetics systems for geotechnical applications. In the computer graphics community, similar work has been done. For example Guo et al. [2018] presented an MPM for thin shells with frictional contact where the shells are represented by Catmull-Clark subdivision surfaces of which control points are treated as particles in an MPM method. In a related work, but for bird strike simulations, Wu et al. [2018] presented a coupling of shell finite elements with the MPM (to model the large deformation of the poor birds).

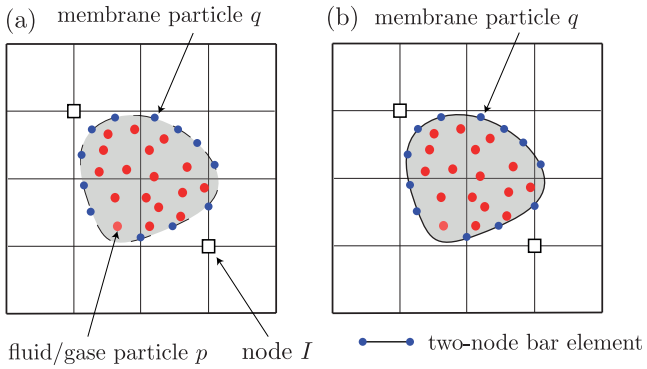


Figure 7: Monolithic FSI solver using the MPM. If the solid part is a structure, it can be discretized by just a set of particles as in (a) or it is modeled as a set of finite elements (bar elements in 2D and membrane elements in 3D).

The algorithm of York et al. [1999, 2000] was applied to fluid-structure interaction problems in Hu et al. [2009], Mao [2013], Yang et al. [2018], Su et al. [2019], Sun et al. [2019]. In Su et al. [2019], temperature effects were considered. Sun et al. [2019] presented a set of benchmark tests for FSI problems and verified MPM results against experiments and other numerical methods. All these works only consider no-slip condition on the fluid-structure interface, to the exception of Hu et al. [2009, 2011] who considered slip boundary conditions. Hu and coworkers' works introduced many techniques for a robust MPM-based fluid-structure interaction simulator: (i) interface material points to track the fluid- structure interface, (ii) fluid particle regularization (or redistribution) to alleviate large particle distortion which is typical of fluid motion, (iii) adaptive mesh refinement, using GIMP, to reduce computational cost that is inherent in traditional uniform grids.

Partitioned FSI MPM. There also exist hybrid approaches or partitioned approaches where a fluid flow solver is coupled with an MPM solid solver for FSI problems e.g., Guilkey et al. [2007], Gilmanov and Acharya [2008b], Sun et al. [2010]. This was motivated by the fact that the MPM is not the best solver for fluids.

In Gilmanov and Acharya [2008b,a] the hybrid immersed boundary method (HIBM) for fluids is combined with the MPM for solids and is presented as an effective strategy for solving 3D fluid-structure interaction problems. The idea is based on the immersed boundary method of Peskin [2002] where the fluid is treated using a Cartesian grid (with finite difference

solver) and the fluid-structure interface is immersed in this grid. In Gilmanov and Acharya [2008b], the structure is a 3D soft membrane (for example a capsule moving in a blood) which is discretized by an unstructured mesh made of three-node triangular finite elements. The deformation of the membrane was, however, not treated by using the FEM but the MPM. That is, the membrane mass and internal forces (calculated the FEM way) are projected to an MPM grid. Therefore, there are the grids in this method: one Cartesian grid for the fluid, one unstructured grid for the membrane, and one Cartesian background grid to solve for the membrane deformation. As one can see, the work of Lian et al. [2011c], Hamad et al. [2015] reinvented this algorithm (without knowing its existence).

1.2.3 Image-based simulations

The MPM (or any meshfree methods) is better suited for image-based simulations involving large deformations than the FEM. This is because the MPM requires only a set of particles, not a body-fitted mesh. It is easier to transform an image into a set of points (particles in the MPM) rather than a finite element mesh. The algorithm used for this is as simple as reading the image voxel by voxel, converting each voxel into a material point. This idea has been exploited in a couple of works such as Bardenhagen et al. [2005], Brydon et al. [2005], Lelong and Rochais [2019] for the simulation of foam microstructures, Nairn [2006] for woods, Lee and Huang [2010] for low-density snow, and Xue et al. [2006b,a] for highly filled composites and nanoparticle-polymer composite membranes.

Image-based simulations find natural applications in biomechanics, see for instance Guilkey et al. [2006], Liu et al. [2015b]. A recent work in this direction is that of Liu and Sun [2019] in which a shift boundary method to apply boundary conditions on surfaces not aligned with the grid nodes is introduced .

There are two points warrant further discussions on this. First, all meshless methods, not just the MPM, are suitable for image-based numerical simulations. Second, when images of very high resolution are involved, the resulting numerical model can be very large. Therefore converting each voxel to a particle might not be the most efficient way to do image-based simulations. In spite of that, as of today and to the best of our knowledge, nobody has published more efficient algorithms.

Remark 6 *It should be noted that there are many contact events in the simulations of foam densification reported in Bardenhagen et al. [2005], Brydon et al. [2005] but they are just no-slip contacts. If needed, frictional contacts of these complex foam materials can be treated using the self-contact method of Homel and Herbold [2017]. But this has yet to be done.*

1.2.4 Computer graphics

Why discussing works done by the computer graphics community? Because innovative MPM algorithms have been developed by this community. Their computer science knowledge helped them to write very efficient MPM codes that can run in real-time [Hu et al., 2019]. We, engineers, can definitely benefit from their contributions. Being aware of the advances made by the computer graphics community would prevent people from reinventing the wheel.

The MPM is now very popular in computer graphics. It is integrated into the production framework of Walt Disney Animation Studios and has been used in featured animations including Frozen, Big Hero 6 and Zootopia [Jiang et al., 2016]. The power of the MPM has been demonstrated in a number of papers for simulating various materials including elastic objects, sand, cloth, hair, snow, lava, and viscoelastic fluids [Daviet and Bertails-Descoubes, 2016, Fu et al., 2017, Jiang et al., 2017, Hu et al., 2018, Wolper et al., 2019, Han et al., 2019]. It all started with the pioneering work of Stomakhin et al. [2014a] who developed an semi-implicit MPM technique and constitutive model to animate the unique behavior of snow. The snow is treated as a continuum avoiding the need to model every snow grain. Jiang et al. [2015b] subsequently extended this MPM to incorporate phase changes due to heat flow. Yue et al. [2015] presented a method to simulate dense foams that exhibit nonlinear, viscoplastic behaviors using the highly flexible Herschel-Bulkley constitutive model. To robustly treat large shearing effects characteristic of dense foams, the authors developed a particle resampling technique, based on the Poisson disk sampling, for the MPM to prevent the formation of nonphysical voids.

The computer graphics community focuses on the efficiency and visual effects of simulators, but not so much on the physics. So, if engineers want to use advances made by this community, they need to make sure that the physics relevant for the considered problems are well modeled.

1.2.5 Other applications

In the context of geophysical simulation, Moresi et al. [2003, 2007] reinterpreted the MPM as a type of FEM with the particles as integration points. They coined the method FEM with Lagrange integration points (FEMLIP). They computed quadrature weights such that affine functions can be exactly re-constructed, giving a second-order accurate reconstruction. Their entire algorithm does, however, not achieve second-order accuracy, due to other low-order approximations as in the

standard MPM. Later, they proposed a multiscale MPM in the sense of FE^2 methods (reviews of which are given in Geers et al. [2010], Nguyen et al. [2012]) using computational homogenization to obtain on-the-fly micromechanically derived constitutive behaviors.

The MPM was used in sea ice models for climate simulation [Sulsky et al., 2007], snow avalanche [Gaume et al., 2018] as well as more traditional explosive-related simulations e.g., explosive welding [Wang et al., 2011], high explosive explosion [Ma et al., 2009a], explosively driven metals [Lian et al., 2011b], cutting processes [Ambati et al., 2012], high melting explosive with cavities [Pan et al., 2008], blast and fragmentation [Banerjee, 2004, Hu and Chen, 2006], and wear [Mishra et al., 2019]. Different low and high velocity impact problems were studied in Li et al. [2011], Zhou et al. [2013], Ye et al. [2018].

1.3 Mathematical analysis

Through the aforementioned literature review one can see that the MPM is a computationally effective method for solving complex solid/fluid mechanics problems involving large deformations, contacts and fractures which are sometimes problematic for other numerical methods. For all that, it has been shown that even for simple one dimensional problems, the convergence is not optimal (for instance, lower than the FEM). And when the mesh is very fine, the method does not converge at all, cf. [Sulsky and Gong, 2016] and Section 6.2. Like most meshfree/particle methods, analysis of the method is not an easy task. This difficulty stems from the fact that there are more than one type of error in the MPM: interpolation error, temporal error, quadrature error, particle to grid mapping error etc. which are all inter-related. The lack of an analysis framework for the MPM, as found in FE methods, makes it difficult to explain unexpected numerical artifacts often found in its employment. Only few works were published on this difficult but important topic e.g., Wallstedt and Guilkey [2008], Tran et al. [2010], Steffen et al. [2008c,b, 2010], Gritton and Berzins [2017], Berzins [2018]. A very useful technique is the method of manufactured solutions (see Knupp and Salari [2003] and Appendix D) to verify MPM implementations for large deformation solid mechanics problems where analytical solutions do not exist.

1.4 MPM resources

Even though the implementation of the MPM is relatively simple compared with other meshfree methods (in fact it can be considered as an updated Lagrangian FEM), there are still many pitfalls. Usually, these pitfalls (e.g., boundary conditions, small nodal masses etc.) are not well documented in the peer reviewed literature. However, they are often detailed in doctoral dissertations. Some good theses on the MPM are given in Section 1.4.1. Section 1.4.2 lists open source and commercial MPM codes that we are aware of in a hope that they can serve as a starting point for junior researchers in their journey with the MPM.

We think that beginners to the field should implement the MPM themselves using a high-level language such as Matlab, Python, or Julia; as it is the only way to fully understand the method. To that end, the note of Nguyen [2014b] and the Julia implementation[5] in Sinaie et al. [2017] can be consulted. On the other hand, to solve large scale problems, one would need a more efficient MPM code, usually implemented in a low-level language such as Fortran or C++. This need can probably be covered by existing efficient open source or commercial MPM codes which are presented in Section 1.4.2.

1.4.1 Doctoral dissertations

There exist excellent doctoral dissertations on the MPM which provide details on many features of the method which are not present in articles. For example, York [1997] detailed the first contact-release algorithm, the first weakly compressible fluid MPM and the first FSI model. Steffen [2009] analyzed the errors of the MPM. For the MPM formulations specifically developed for geo-technical engineering problems, the doctoral dissertations of Coetzee [2003], Andersen [2009], Beuth [2012], Mast [2013], Al-Kafaji [2013], Hamad [2014] are worth being consulted. In computer graphics, Jiang [2015] is the first PhD thesis on the topic. For geo-technical community, the website of the Anura3D MPM Research Community at `http://www.mpm-dredge.eu/home` provides many useful information, including an updated publication lists on MPM for geo-technical engineering.

Remark 7 *People new to this method might find the textbook of Zhang et al. [2016] and the lecture note of Jiang et al. [2016] useful.*

[5] Julia is a high-level programming language designed for high-performance numerical analysis and computational science, similar to Matlab but much faster.

1.4.2 Open source and commercial MPM codes

For engineers and scientists who just want to use existing MPM packages for their research, open source MPM codes are as useful as commercial FE packages such as `Abaqus`. To researchers whose work involves developing new MPM techniques, these codes are also useful as it is not an easy task to develop an efficient, scalable and extensible MPM code from scratch. For both of these audiences, we herein list available, to the best of our knowledge, MPM codes. The aim is to help the readers to find the MPM implementation that suits their needs best.

Some open source MPM implementations are

- `Uintah`, http://www.uintah.utah.edu
- `Mechsys`, http://mechsys.nongnu.org/index.shtml
- `MPM-GIMP`, http://sourceforge.net/p/mpmgimp/home/Home/
- `NairnMPM`, http://osupdocs.forestry.oregonstate.edu/index.php/Main_Page
- `CB-Geo`, https://www.cb-geo.com/research/mpm/
- `MPM3D`, http://comdyn.hy.tsinghua.edu.cn/english/mpm3d

All are written in C++. `CB-Geo` is an MPM code for geomechanics problems. `Uintah` implements probably the most efficient MPM to date: it uses blocked structured adaptive mesh refinement (BSAMR) grids running on hundreds to thousands of processors. It is not easy for people new to this code to modify it though.

A commercial MPM package is `MPMsim`, found at http://www.mpmsim.com. The core is written in C++ and it has a user friendly interface written in Matlab. It can convert CAD files to material points.

1.5 Layout

The previous discussion has painted an overall picture about the MPM. The remaining of the paper is to fill in the details. Eleven ready-to-code algorithms are provided such that everyone can implement the MPM and carry out the simulations given in Fig. 8.

The remaining of the paper is organized as follows. Section 2 describes the general basic MPM for solids. By general, we mean that no specification of the shape functions is given. And by basic, we mean that collisions/contacts are not discussed. Section 3 presents different MPM versions that basically adopt different shape functions (e.g., hat functions, B-splines, GIMP and CPDI). Section 4 discusses some implementation details such as particle generation, initial and boundary conditions, MPM with unstructured grids, visualization of MPM results and `Karamelo`. Some advanced topics such as contacts, high order MPM, adaptivity, volumetric locking and fluid modeling are given in Section 5. Section 6 is devoted to an extensive set of MPM simulations carried out by the authors using our in-house codes. We have three sets of examples: (i) implementation tests containing simple one and two dimensional examples to verify MPM implementations; (ii) convergence tests that use the method of manufactured solutions to determine the convergence rate and stability of MPMs, and (iii) practical engineering problems.

The paper also contains some appendices. Appendix A briefly presents the method of manufactured solutions. Common material models are given in Appendix B. The relation between CPDI and the projection of Lagrangian mesh quantities to a Eulerian grid is discussed in Appendix C for a better understanding of CPDI. Appendix D discusses the moving least square approximations used in the iMPM. Some useful utilities are given in Appendix E such as how to use a computer algebra system to derive CPDI functions. Appendix F gives a short but practical presentation of updated and total Lagrangian FEM for nonlinear solid mechanics. This is useful to see the difference and similarities of the MPM and the FEM. Axi-symmetric MPM formulations are briefly introduced in Appendix G.

1.6 Notations

Three notations are adopted in this work namely indicial notation, tensor notation and matrix notation (also known as engineering notation). The squared magnitude of a three-dimensional vector expressed in these three notations is given in the following equation

$$r^2 = \underbrace{x_i x_i}_{\text{indicial notation}} = \underbrace{\mathbf{x}\cdot\mathbf{x}}_{\text{tensor notation}} = \underbrace{\mathbf{x}^{\mathrm{T}}\mathbf{x}}_{\text{matrix notation}} \tag{1.1}$$

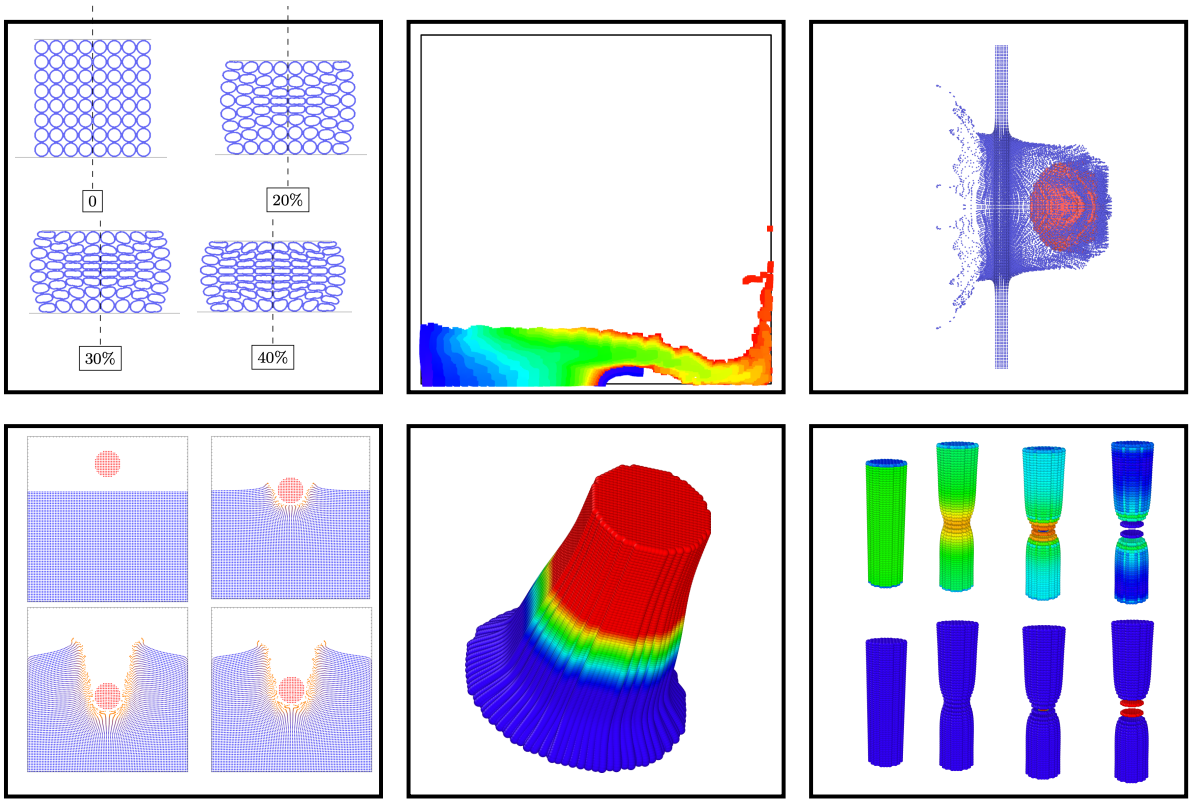


Figure 8: Representative simulations that can be done using MPM algorithms presented in this paper. We refer to Section 6 for details. These simulations involve very large deformation, contacts and fracture. And one simulation is a fluid-structure interaction problem.

Any vectors and tensors in this paper is defined in a rectangular Cartesian coordinate system. Therefore a vector $\mathbf{x}$ can be written as

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad \text{or} \quad \mathbf{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{1.2}$$

In indicial notation, the components of tensors are explicitly specified e.g., a vector in indicial notation is hence given by x_i in which the index i ranges from one to the number of spatial dimensions. Indices which are repeated twice in a term are summed, a rule known as the Einstein summation, an example of which is the squared magnitude of a vector:

$$r^2 = x_i x_i = x_1^2 + x_2^2 + x_3^2 \tag{1.3}$$

Scalars are written using a normal font whereas tensors of order one or greater are expressed in boldface. Table 2 provides a list of important notations used in this contribution. And Table 3 presents abbreviations.

2 A general MPM for solid mechanics

This section presents a general formulation of the MPM for solid mechanics. This formulation applies to all existing MPM variants such as GIMP, CPDI and BSMPM. Here, even though only explicit dynamics MPM is presented in great details, we also touch briefly on implicit dynamics and quasi-static MPM. Moreover, both updated and total Lagrangian MPM are treated.

We start with a short review of continuum mechanics in Section 2.1. Next, governing equations using the updated Lagrangian description written in a strong form are stated in Section 2.2. The weak form corresponding to the equation of motion is given in Section 2.3. Also, the MPM spatial discretization procedure of this weak form is treated. The obtained semi-discrete equations can also be derived from the updated Lagrangian FEM weak form discretization considering the finite elements' quadrature points as material points as shown in Section 2.4. Time discretization of the semi-discrete equations is

Variable	Type	Meaning
$\mathbf{x}_p$	Vector	Particle position (time-dependent)
$\mathbf{X}_p$	Vector	Particle initial position
$\mathbf{v}_p$	Vector	Particle velocity
m_p	Scalar	Particle mass
V_p	Scalar	Particle volume
ρ_p	Scalar	Particle density
T_p	Scalar	Particle temperature
$\boldsymbol{\sigma}_p$	Tensor/Matrix	Particle Cauchy stress
$\mathbf{P}_p$	Tensor/Matrix	Particle 1$^{\text{st}}$ Piola-Kirchoff stress
$\mathbf{F}_p$	Tensor/Matrix	Particle deformation gradient
$\mathbf{L}_p$	Tensor/Matrix	Particle velocity gradient
$\mathbf{D}_p$ or $\dot{\boldsymbol{\epsilon}}_p$	Tensor/Matrix	Particle rate of deformation
$\mathbf{v}_I$	Vector	Node velocity
$\tilde{\mathbf{v}}_I^{t+\Delta t}$	Vector	Temporary updated node velocity
$\mathbf{v}_I^{t+\Delta t}$	Vector	Final updated node velocity
m_I	Scalar	Node mass
$\Phi_I(\mathbf{x}_p)$ or Φ_{Ip}	Scalar	Weighting function of node I evaluated at particle p
$\nabla\Phi_I(\mathbf{x}_p)$ or $\nabla\Phi_{Ip}$	Vector	Gradient of weighting function of node I evaluated at particle p
$\nabla_0\Phi_I(\mathbf{x}_p)$ or $\nabla_0\Phi_{Ip}$	Vector	Gradient (w.r.t $\mathbf{X}$) of weighting function of node I evaluated at p

Table 2: Important notations used in this contribution. Subscript I denotes the value of grid nodes, and subscript p denotes the value of particles.

Abbreviation	Full name
MPM	Material Point Method
ULMPM	Updated Lagrangian Material Point Method
TLMPM	Total Lagrangian Material Point Method
PIC	Particle in Cell
FLIP	Fluid Implicit Particle method
FEM	Finite Element Method
ULFEM	Updated Lagrangian Finite Element Method
TLFEM	Total Lagrangian Finite Element Method
CPDI	Convected Particle Domain Integrator
GIMP	Generalized Interpolation Material Point
BSMPM	B-splines Material Point Method
DDMPM	Dual Domain Material Point Method
SPH	Smoothed Particle Hydrodynamics
iMPM	improved Material Point Method

Table 3: A list of abbreviations used in this paper.

presented in Section 2.5. In this section, various MPM algorithms such as Updated Stress Last (USL) and Modified Updated Stress Last (MUSL) are also discussed. Finally, in Section 2.6, the algorithm of the total Lagrangian MPM is provided.

2.1 Basic concepts of continuum mechanics

Continuum mechanics is a theory that models solids and fluids at a macroscopic scale which ignores inhomogeneities such as molecular, granular, or crystal structures. Therefore, within this theory, behavior of solids and fluids can be characterized by smooth functions of spatial variables. The subject of continuum mechanics comprises the following basic topics: (1) the study of motion and deformation without considering the causes (kinematics), (2) the study of internal forces (kinetics), (3) the conservation equations or balance principles that state that there are certain important physical properties e.g., mass,

momentum and energies that must be conserved, and (4) constitutive models that furnish the relationship between kinematics and kinetics variables. It is via a constitutive model that, in continuum mechanics, one differentiate a solid from a fluid, a rubber from a rock, etc.

This section reviews the key concepts and equations of continuum mechanics. Good knowledge of continuum mechanics is essential for the understanding of the MPM as it is a continuum-based numerical method. Derivations are not presented, but relevant literature is given. For more details, we refer to standard textbooks such as Malvern [1969], Gurtin [1981], Marsden and Hughes [1983], Ogden [1984], Holzapfel [2000].

2.1.1 Motion and deformation

In continuum mechanics, a body $\mathcal{B}$ is considered as being formed by an infinite set of material points, which are endowed with certain mechanical properties. The position vector of a material point in the initial, undeformed configuration of the body is denoted $\mathbf{X}$ relative to some coordinate basis. $\mathbf{X}$ is named the *material or Lagrangian coordinate*. The position of the same material point, in the deformed configuration, is designated by $\mathbf{x}$ called *spatial or Eulerian coordinate*.

The motion (deformation) of a solid is described by a function $\boldsymbol{\phi}(\mathbf{X}, t)$. A relation between spatial coordinates and material coordinates can be established as follows

$$\mathbf{x} = \boldsymbol{\phi}(\mathbf{X}, t) \tag{2.1}$$

The displacement, velocity and acceleration fields of a body are the primary kinematical fields in describing the motion of the body. The displacement of a material point $\mathbf{X}$, denoted by $\mathbf{u}(\mathbf{X}, t)$, is the difference between its current position $\phi(\mathbf{X}, t)$ and its initial position $\phi(\mathbf{X}, 0)$. So,

$$\mathbf{u}(\mathbf{X}, t) := \boldsymbol{\phi}(\mathbf{X}, t) - \boldsymbol{\phi}(\mathbf{X}, 0) = \mathbf{x} - \mathbf{X} \tag{2.2}$$

The velocity of a material point $\mathbf{X}$, denoted by $\mathbf{v}(\mathbf{X}, t)$, is defined as the rate of change of position of this material point, that is

$$\mathbf{v}(\mathbf{X}, t) := \frac{\partial \boldsymbol{\phi}(\mathbf{X}, t)}{\partial t} \tag{2.3}$$

This is the Lagrangian velocity field. There exists a Eulerian form for this velocity field but as the MPM adopts a Lagrangian description, it is not discussed here.

The acceleration of a material point $\mathbf{X}$ is the rate of change of its velocity, i.e., the material time derivative of the velocity,

$$\mathbf{a}(\mathbf{X}, t) := \frac{\partial \mathbf{v}(\mathbf{X}, t)}{\partial t} = \frac{\partial^2 \boldsymbol{\phi}(\mathbf{X}, t)}{\partial t^2} \tag{2.4}$$

The deformation gradient tensor $\mathbf{F}$ is a key quantity in finite deformation continuum mechanics as all deformation quantities are derived from it. It is a linear mapping operator which maps each infinitesimal linear element $d\mathbf{X}$ in the reference configuration into an infinitesimal linear element $d\mathbf{x}$ in the current configuration. It is defined as:

$$\mathbf{F} := \frac{\partial \boldsymbol{\phi}}{\partial \mathbf{X}} = \frac{\partial \mathbf{x}}{\partial \mathbf{X}} \quad \text{or} \quad F_{ij} = \frac{\partial x_i}{\partial X_j} \tag{2.5}$$

Next, the concept of *material time derivative* is introduced. To understand this important concept, consider the following situation. Assume we have a certain field φ (scalar, vectorial or tensorial) defined over the body for which we want to know the rate of change, at a given material point $\mathbf{X}$. This is known as the material time derivative of φ. There are two definition of this concept, corresponding to material and spatial descriptions, respectively:

1. Lagrangian description. In the Lagrangian description, the independent variables are the material coordinates $\mathbf{X}$ and time t. So all we have to do is taking the partial derivative of the given field φ with respect to time. For a material field $\varphi(\mathbf{X}, t)$, its material time derivative is
$$\frac{D\varphi(\mathbf{X}, t)}{Dt} \equiv \dot{\varphi} = \frac{\partial \varphi(\mathbf{X}, t)}{\partial t} \tag{2.6}$$
where the first two equations indicate standard notation for the material time derivative. As the MPM adopts a Lagrangian description, the material time derivative is very simple.

2. Eulerian description. The considered field is $\varphi(\mathbf{x}, t)$. This case is much more complicated since not only time changes but also the spatial position $\mathbf{x}$ of the considered particle . We must calculate the partial derivative with respect to time of the material description of $\varphi(\mathbf{x}, t)$, keeping $\mathbf{X}$ fixed.

$$\frac{D\varphi(\mathbf{x},t)}{Dt} \equiv \dot{\varphi} := \lim_{\Delta t\to 0} \frac{\varphi(\boldsymbol{\phi}(\mathbf{X},t+\Delta t),t+\Delta t)-\varphi(\boldsymbol{\phi}(\mathbf{X},t),t)}{\Delta t} \tag{2.7}$$

Using the chain rule, we obtain the important formula of the material time derivative for a Eulerian scalar field:

$$\frac{D\varphi(\mathbf{x},t)}{Dt} = \frac{\partial\varphi(\mathbf{x},t)}{\partial t} + \nabla\varphi(\mathbf{x},t)\cdot\mathbf{v}(\mathbf{x},t) \tag{2.8}$$

The term $\partial\varphi/\partial t$ is called the *spatial time derivative*, and the term $\varphi_{,j}v_j$ is the convective term, which is also called the transport term.

2.1.2 Strain measures

There are many measures of strain and strain rate in nonlinear continuum mechanics. A strain measure must vanish for any rigid body motion, and in particular for rigid body rotation. Herein, we review some strain measures commonly adopted in nonlinear continuum mechanics. They are the right Cauchy-Green deformation tensor, the Green strain tensor, and the rate of deformation tensor.

The right Cauchy-Green deformation tensor is written as

$$\mathbf{C} := \mathbf{F}^{\mathrm{T}}\cdot\mathbf{F}; \quad C_{ij} := F_{ki}F_{kj} \tag{2.9}$$

where the superscript T denotes the transpose operator. The Green strain tensor is given by

$$\mathbf{E} := \frac{1}{2}(\mathbf{C}-\mathbf{I}) = \frac{1}{2}(\mathbf{F}^{\mathrm{T}}\cdot\mathbf{F}-\mathbf{I}), \quad E_{ij} = \frac{1}{2}(F_{ki}F_{kj}-\delta_{ij}) \tag{2.10}$$

This strain tensor measures the difference of the square of the length of $d\mathbf{x}$ and $d\mathbf{X}$.

The spatial gradient of velocity or *velocity gradient tensor* $\mathbf{L}$ is defined as the spatial gradient of the velocity, that is

$$\mathbf{L}(\mathbf{x},t) := \frac{\partial\mathbf{v}}{\partial\mathbf{x}}, \quad \text{or} \quad L_{ij} = \frac{\partial v_i}{\partial x_j} \tag{2.11}$$

The velocity gradient $\mathbf{L}$ allows the material time derivative of the deformation gradient $\mathbf{F}$ to be written as

$$\dot{\mathbf{F}} = \frac{\partial}{\partial t}\left(\frac{\partial\boldsymbol{\phi}(\mathbf{X},t)}{\partial\mathbf{X}}\right) = \frac{\partial\mathbf{v}}{\partial\mathbf{X}} = \frac{\partial\mathbf{v}}{\partial\mathbf{x}}\cdot\frac{\partial\mathbf{x}}{\partial\mathbf{X}} = \mathbf{L}\cdot\mathbf{F}, \quad \Rightarrow \mathbf{L} = \dot{\mathbf{F}}\cdot\mathbf{F}^{-1} \tag{2.12}$$

where in the second equality, we have used the fact that material time derivative of Lagrangian fields commute with material gradient. Noting that, this fact does not hold generally for Eulerian fields.

The velocity gradient tensor can be decomposed into symmetric and skew-symmetric parts by

$$\mathbf{L} = \frac{1}{2}(\mathbf{L}+\mathbf{L}^{\mathrm{T}}) + \frac{1}{2}(\mathbf{L}-\mathbf{L}^{\mathrm{T}}) \tag{2.13}$$

which is a standard decomposition of a second-order tensor. The rate of deformation tensor $\mathbf{D}$ is defined as the symmetric part of $\mathbf{L}$, and the spin tensor as the skew-symmetric part. Using these definitions, one can write

$$\mathbf{D} := \frac{1}{2}(\mathbf{L}+\mathbf{L}^{\mathrm{T}}), \quad \mathbf{W} := \frac{1}{2}(\mathbf{L}-\mathbf{L}^{\mathrm{T}}) \tag{2.14}$$

where $\mathbf{D}$ describes the rate of stretching and shearing.

2.1.3 Stress measures

Since there are different strain measures there also exist different stress measures which are work conjugated with them. The most commonly used stress tensors are (1) Cauchy stress, (2) Kirchhoff stress, (3) first Piola-Kirchhoff stress (1^{st} PK) and (4) second Piola-Kirchhoff stress (2^{nd} PK). Relation between these stress tensors is given in Table 4. The Cauchy stress is the true stress and work conjugate with the rate of deformation $\mathbf{D}$ with respect to the deformed volume, cf. Equation (2.19). The Kirchhoff stress–also referred to as the weighted Cauchy stress–is work conjugate with the rate of deformation tensor with respect to the initial volume. The 1^{st} PK stress, which is not symmetric, is work conjugate to the rate of deformation gradient The 2^{nd} Piola-Kirchhoff stress, a totally material symmetric stress tensor, is work conjugate to the Green strain rate tensor. Note that, some authors e.g., Belytschko et al. [2000], prefer to work with the nominal stress, the transpose of which is the 1^{st} PK stress.

	Cauchy stress $\boldsymbol{\sigma}$	Kirchhoff stress $\boldsymbol{\tau}$	1st PK $\mathbf{P}$	2nd PK $\mathbf{S}$
$\boldsymbol{\sigma}$	-	$\boldsymbol{\tau} J^{-1}$	$J^{-1}\mathbf{P}\mathbf{F}^{\mathrm{T}}$	$J^{-1}\mathbf{F}\mathbf{S}\mathbf{F}^{\mathrm{T}}$
$\boldsymbol{\tau}$	$J\boldsymbol{\sigma}$	-	$\mathbf{P}\mathbf{F}^{\mathrm{T}}$	$\mathbf{F}\mathbf{S}\mathbf{F}^{\mathrm{T}}$
$\mathbf{P}$	$J\boldsymbol{\sigma}\mathbf{F}^{-\mathrm{T}}$	$\boldsymbol{\tau}\mathbf{F}^{-\mathrm{T}}$	-	$\mathbf{F}\mathbf{S}$
$\mathbf{S}$	$J\mathbf{F}^{-1}\boldsymbol{\sigma}\mathbf{F}^{-\mathrm{T}}$	$\mathbf{F}^{-1}\boldsymbol{\tau}\mathbf{F}^{-\mathrm{T}}$	$\mathbf{F}^{-1}\mathbf{P}$	-

Table 4: Relation between different stress measures.

2.1.4 Objective stress rates

Constitutive equations are often written in a rate form i.e., a relation between a stress rate and a deformation rate. Under large rotations simply using the material derivatives of the stress tensors e.g., the rate of Cauchy stress $D\boldsymbol{\sigma}/Dt$ is wrong since it does not transform properly as a tensor under a superposed rigid body motion. We discuss three objective stress rates: the Jaumman rate, the Truesdell rate and the Green-Naghdi rate which are frequently used in practice. A constitutive model can be formulated in terms of any one of these objective stress rates, and changing from one rate to another requires that the constitutive model be reformulated.

They are collectively given in Equation (2.15).

$$\begin{aligned}
\boldsymbol{\sigma}^{\nabla T} &= \frac{D\boldsymbol{\sigma}}{Dt} + \mathrm{div}(\mathbf{v})\boldsymbol{\sigma} - \mathbf{L}\cdot\boldsymbol{\sigma} - \boldsymbol{\sigma}\cdot\mathbf{L}^{\mathrm{T}} && \text{Truesdel rate} \\
\boldsymbol{\sigma}^{\nabla J} &= \frac{D\boldsymbol{\sigma}}{Dt} - \mathbf{W}\cdot\boldsymbol{\sigma} - \boldsymbol{\sigma}\cdot\mathbf{W}^{\mathrm{T}} && \text{Jaumman rate} \\
\boldsymbol{\sigma}^{\nabla G} &= \frac{D\boldsymbol{\sigma}}{Dt} - \boldsymbol{\Omega}\cdot\boldsymbol{\sigma} - \boldsymbol{\sigma}\cdot\boldsymbol{\Omega}^{\mathrm{T}} && \text{Green-Naghdi rate}
\end{aligned} \tag{2.15}$$

A discussion on which objective stress rate to be used was given in Benson [1992].

2.1.5 Conservation equations

An important set of equations in continuum mechanics are the conservation equations or balance equations. For thermomechanical systems, the conservation laws include

1. Conservation of mass
2. Conservation of linear momentum
3. Conservation of angular momentum, and
4. Conservation of energy

Conservation of mass. The law of conservation of mass is described by the *equation of mass conservation*, or often called *equation of continuity* written as:

$$\frac{D\rho}{Dt} + \rho\nabla\cdot\mathbf{v} \quad \text{or} \quad \dot{\rho} + \rho v_{i,i} = 0 \tag{2.16}$$

If the density does not change, i.e., the material is incompressible, hence the material time derivative of the density vanishes, and the continuity equation becomes $v_{i,i} = 0$ which is the well known incompressibility condition. For Lagrangian description, a simpler algebraic equation for the mass conservation is given by

$$\rho J = \rho_0 \tag{2.17}$$

Conservation of linear momentum. This law demands that the change of linear momentum in time is equal to the sum of all external forces (volume and surface forces) acting on the body. It is described by the so-called *the momentum equation*:

$$\rho\frac{D\mathbf{v}}{Dt} = \nabla\cdot\boldsymbol{\sigma} + \rho\mathbf{b} \quad \text{or} \quad \rho\dot{v}_i = \sigma_{ji,j} + \rho b_i \tag{2.18}$$

Conservation of angular momentum. This laws requires that the Cauchy stress be a symmetric tensor.

Conservation of energy. This law states that the rate of change of the total energy in the body (consisting of the internal energy and kinetic energy) is equal to the rate of work done by the external forces plus the rate of work provided by heat flux $\mathbf{q}$ and energy sources.

$$\rho \frac{De}{Dt} = \mathbf{D} : \boldsymbol{\sigma} - \nabla \cdot \mathbf{q} + \rho s \tag{2.19}$$

where e is the specific internal energy; ρs indicates a heat source per unit volume.

2.1.6 Constitutive models

All the equations given previously are material independent: they are valid for both solids and fluids. To model material behavior a *constitutive equation* or *constitutive relation* – a relation between kinetic quantities (e.g., stresses) as related to kinematic quantities (e.g., strains) – is needed. It is through constitutive equations that one can differentiate fluids from solids, concretes from rubbers etc. The first constitutive equation (constitutive law) was developed by Robert Hooke and is known as Hooke's law. It deals with the case of linear elastic materials. Since then, a plethora of constitutive models has been developed to characterize a diverse range of natural and engineering materials [Gurtin, 1981, Marsden and Hughes, 1983, Ogden, 1984]. For the sake of completeness, we summarize in Appendix B some commonly used constitutive models for elastic and plastic solids. It is worth noting that numerical simulations can only be as accurate as the utilized material models.

2.2 Strong form

For a continuum body under purely mechanical loading (neglect heat exchange), governing differential equations in an updated Lagrangian description include balance laws, constitutive equation, kinematics equation and boundary/initial conditions, which are collectively given as

$$\begin{aligned}
&\frac{D\rho}{Dt} + \rho \nabla \cdot \mathbf{v} = 0 && \text{(conservation of mass)} \\
&\rho \frac{D\mathbf{v}}{Dt} = \nabla \cdot \boldsymbol{\sigma} + \rho \mathbf{b} && \text{(conservation of linear momentum)} \\
&\rho \frac{De}{Dt} = \mathbf{D} : \boldsymbol{\sigma} && \text{(conservation of energy)} \\
&\mathbf{D} = \text{sym}(\nabla \mathbf{v}) && \text{(strain measure)} \\
&\boldsymbol{\sigma}^{\nabla} = S_t^{\sigma D}(\mathbf{D}, \boldsymbol{\sigma}) && \text{(constitutive equation)} \\
&\mathbf{v}(\mathbf{x}, t = 0) = \mathbf{v}_0, \ \boldsymbol{\sigma}(\mathbf{x}, t = 0) = \boldsymbol{\sigma}_0 && \text{(initial conditions)} \\
&\mathbf{u} = \bar{\mathbf{u}} \text{ on } \Gamma_u && \text{(Dirichlet boundary conditions)} \\
&\mathbf{t} = \bar{\mathbf{t}} \text{ on } \Gamma_t && \text{(Neumann boundary conditions)}
\end{aligned} \tag{2.20}$$

where $\rho(\mathbf{X}, t)$ is the density, $\mathbf{v}(\mathbf{X}, t)$ denotes the velocity, $\boldsymbol{\sigma}(\mathbf{X}, t)$ is the Cauchy stress tensor, $\mathbf{b}$ is the specific body force and ∇ is the gradient operator with respect to the current configuration. The rate of deformation tensor is represented by $\mathbf{D}(\mathbf{X}, t)$, $\boldsymbol{\sigma}^{\nabla}$ denotes some objective stress rates which are necessary for large rotations.

The conservation of energy equation is used to update the internal energy for the equation of state and to check the energy conservation. As a Lagrangian description is used in MPM, conservation of mass, first equation in Equation (2.20) is not solved. The formulation of the basic MPM is isothermal and thus, the energy equation is not solved either. We postpone the treatment of temperature effects to Section 5.3. Known quantities include prescribed displacements $\bar{\mathbf{u}}$ (or equivalently prescribed velocities) on the Dirichlet boundary Γ_u, prescribed tractions $\bar{\mathbf{t}}$ on the traction boundary Γ_t, initial velocities $\mathbf{v}_0$ and initial stresses $\boldsymbol{\sigma}_0$. Recall that in the updated Lagrangian formulation, the material time derivative is simply a partial derivative with respect to time $\frac{D\varphi(\mathbf{X},t)}{Dt} \equiv \dot{\varphi} = \frac{\partial \varphi(\mathbf{X},t)}{\partial t}$. This is much simpler than the Eulerian formulation of the material time derivative.

As mentioned above, the independent variables in Lagrangian formulation are the material coordinate $\mathbf{X}$ and time t. The dependent variables include (i) mass density $\rho(\mathbf{X}, t)$ (one unknown), (ii) velocity field[6] $\mathbf{v}$ (3 unknowns in 3D), (iii) stress field $\boldsymbol{\sigma}$ (6 unknowns as Cauchy stress tensor is symmetric) and (iv) the deformation rate tensor $\mathbf{D}$ (6 unknowns). Therefore, in total, we have 16 unknowns in three dimensions. The governing equations include (i) conservation of mass (1 equation), (ii) conservation of momenta (3 equations), (iii) conservation of energy (1 equation), (iv) constitutive equations (6 equations that relate the six stress components to the six components of the deformation rate tensor) and (v) strain-displacement equations

[6]Having the velocity one can obtain the displacement and acceleration fields.

(6 equations). In total, we have 17 equations. However, since we are interested in non-adiabatic, non-isothermal processes, the conservation of energy is not a PDE, so finally we have 16 equations for 16 unknowns. Note that the conservation of energy is needed in the so-called equation of state that relates pressure, volume and specific energy.

2.3 Weak form and spatial discretization

The MPM, which can be considered as an updated Lagrangian FEM, also employs a weak formulation. The weak form of the momentum equation, the second equation in Equation (2.20), is given by see e.g., Belytschko et al. [2000]

$$\int_{\Omega} \rho \delta u_i a_i \mathrm{d}\Omega + \int_{\Omega} \rho \frac{\partial \delta u_i}{\partial x_j} \sigma^s_{ij} \mathrm{d}\Omega = \int_{\Omega} \rho \delta u_i b_i \mathrm{d}\Omega + \int_{\Gamma_t} \rho \delta u_i \bar{t}^s_i \mathrm{d}\Gamma \tag{2.21}$$

where Ω denotes the current configuration, σ^s_{ij} is the specific Cauchy stresses i.e., $\sigma^s_{ij} = \sigma_{ij}/\rho$; subscripts $i, j = 1, 2, 3$ used to denote components of vectors and tensors; $\delta\mathbf{u}$ is the virtual displacement field or the test functions; $\mathbf{a}$ the acceleration field. The specific traction vector is denoted by $\bar{t}^s_i$. Derivation of this weak form is standard and therefore skipped here. Note that the above was written in indicial notation which will facilitate the derivation of the discrete equations. From this weak form, one can proceed as Sulsky et al. [1994] in what follows to obtain the semi-discrete equations for the ULMPM or one can derive these equations from the ones of the ULFEM by considering the particles as quadrature points, see Section 2.4.

The whole material domain Ω is discretized by a set of material sub-domains Ω_p, and it is assumed that the whole mass of a material sub-domain is concentrated at the corresponding material point, which means that the mass density field is expressed as

$$\rho(\mathbf{x}, t) = \sum_{p=1}^{n_p} m_p \delta(\mathbf{x} - \mathbf{x}_p) \tag{2.22}$$

where δ is the Dirac delta function with dimension of the inverse of volume. Note that m_p denotes the mass of particle p. Substitution of Equation (2.22) into Equation (2.21) results in

$$\sum_{p=1}^{n_p} m_p \delta u_i(\mathbf{x}_p) a_i(\mathbf{x}_p) + \sum_{p=1}^{n_p} m_p \left.\frac{\partial \delta u_i}{\partial x_j}\right|_{(\mathbf{x}_p)} \sigma^s_{ij}(\mathbf{x}_p) = \sum_{p=1}^{n_p} m_p \delta u_i(\mathbf{x}_p) b_i(\mathbf{x}_p) + \sum_{p=1}^{n_p} m_p \delta u_i(\mathbf{x}_p) \bar{t}^s_i(\mathbf{x}_p) h^{-1} \tag{2.23}$$

where use was made of the identity $\int f(\mathbf{x})\delta(\mathbf{x} - \mathbf{x}_p) = f(\mathbf{x}_p)$. As particles constitute a volume, one needs to introduce a boundary layer thickness h in the calculation of the external force due to the traction [Zhang et al., 2016] .

Next, the space is discretized by a finite element mesh (or a grid) so that any spatially varying field can be approximated. The mesh consists of n_n nodes with shape functions ϕ_I associated with each node I; $x_{iI}(t)$ is the i component of the position vector of node I. In 3D, one writes $\mathbf{x}_I = (x_{1I}, x_{2I}, x_{3I}) = (x_I, y_I, z_I)$. Subscript I denotes the value of grid nodes, and subscript p denotes the value of particles. Thanks to the use of a FE mesh, evaluation of shape functions and derivatives are standard (thus efficient) and does not involve neighbor search as in other meshfree methods such as SPH or EFG.

The FE approximation of the motion is given by

$$x_i(\mathbf{X}, t) = \sum_{I=1}^{n_n} \phi_I(\mathbf{X}) x_{iI}(t) \tag{2.24}$$

As can be seen, the shape functions are expressed in terms of the Lagrangian coordinates $\mathbf{X}$ not the Eulerian coordinates $\mathbf{x}$, similar to Lagrangian finite elements [Belytschko et al., 2000].

Using Equation (2.24) for the initial configuration one writes

$$X_i = \sum_{I=1}^{n_n} \phi_I(\mathbf{X}) X_{iI} \tag{2.25}$$

with X_{iI} is the i component of $\mathbf{X}_I$–the coordinates of node I in the initial configuration.

The displacement is thus approximated as

$$u_i = x_i - X_i = \sum_{I=1}^{n_n} \phi_I(\mathbf{X})(x_{iI} - X_{iI}) = \sum_{I=1}^{n_n} \phi_I(\mathbf{X}) u_{iI}(t) \tag{2.26}$$

where u_{iI} designates the i component of the displacement vector of node I.

The velocity and acceleration fields are thus given by

$$v_i(\mathbf{X},t) = \sum_{I=1}^{n_n} \phi_I(\mathbf{X}) v_{iI}(t) \tag{2.27}$$

and

$$a_i(\mathbf{X},t) = \sum_{I=1}^{n_n} \phi_I(\mathbf{X}) a_{iI}(t) \tag{2.28}$$

where v_{iI}, a_{iI} are the i component of the velocity and acceleration vectors of node I, respectively. Note that Equation (2.27) is not needed in the derivation of the semi-discrete equations given in this section, it will be used later, for example to compute the velocity gradient and update the particle position.

Using the Bubnov-Galerkin method, the virtual displacement field is approximated as

$$\delta u_i(\mathbf{X}) = \sum_{I=1}^{n_n} \phi_I(\mathbf{X}) \delta u_{iI} \tag{2.29}$$

i.e., the virtual displacement field is approximated using the same shape functions. The spatial derivatives of δu_i is thus given by

$$\frac{\partial \delta u_i}{\partial x_j} = \sum_{I=1}^{n_n} \frac{\partial \phi_I}{\partial x_j} \delta u_{iI} \tag{2.30}$$

Substituting the FE approximations of δu_i, a_i and $\frac{\partial \delta u_i}{\partial x_j}$ evaluated at the particles using Equations (2.28) to (2.30) into Equation (2.23) leads to

$$\sum_{p=1}^{n_p} m_p \left[\sum_{I=1}^{n_n} \phi_I(\mathbf{x}_p)\delta u_{iI}\right]\left[\sum_{J=1}^{n_n} \phi_J(\mathbf{x}_p) a_{iJ}\right] + \sum_{p=1}^{n_p} m_p \left[\sum_{I=1}^{n_n} \left.\frac{\partial \phi_I}{\partial x_j}\right|_{(\mathbf{x}_p)} \delta u_{iI}\right] \sigma^s_{ij}(\mathbf{x}_p) = \sum_{p=1}^{n_p} m_p \left[\sum_{I=1}^{n_n} \phi_I(\mathbf{x}_p)\delta u_{iI}\right] b_i(\mathbf{x}_p) + \sum_{p=1}^{n_p} m_p \left[\sum_{I=1}^{n_n} \phi_I(\mathbf{x}_p)\delta u_{iI}\right] \bar{t}^s_i(\mathbf{x}_p) h^{-1} \tag{2.31}$$

Since δu_{iI} are arbitrary[7], we obtain the following set of equations (3 equations for each node I, $I = 1, \ldots, n_n$)

$$\sum_{p=1}^{n_p} m_p \phi_I(\mathbf{x}_p)\left(\sum_{J=1}^{n_n} \phi_J(\mathbf{x}_p) a_{iJ}\right) + \sum_{p=1}^{n_p} m_p \left.\frac{\partial \phi_I}{\partial x_j}\right|_{(\mathbf{x}_p)} \sigma^s_{ij}(\mathbf{x}_p) = \sum_{p=1}^{n_p} m_p \phi_I(\mathbf{x}_p) b_i(\mathbf{x}_p) + \sum_{p=1}^{n_p} m_p \phi_I(\mathbf{x}_p) \bar{t}^s_i(\mathbf{x}_p) h^{-1} \tag{2.32}$$

which can be written in the following compact form

$$m_{IJ}\mathbf{a}_J = \mathbf{f}^{\text{ext}}_I + \mathbf{f}^{\text{int}}_I, \quad I = 1, 2, \ldots, n_n \tag{2.33}$$

where m_{IJ}, $\mathbf{f}^{\text{ext}}_I$, $\mathbf{f}^{\text{int}}_I$ are IJ component of the consistent mass matrix, the external force vector and the internal force vector, respectively. This equation is exactly identical to the FEM. Equation (2.33) is usually referred to as the semi-discrete equation as just space was discretized.

The IJ component of the consistent mass matrix is given by

$$m_{IJ} = \sum_{p=1}^{n_p} m_p \phi_I(\mathbf{x}_p)\phi_J(\mathbf{x}_p) \tag{2.34}$$

Note that the mass matrix is not constant as in the FEM but changes in time because the material points move while the grid nodes are reset after a time step.

[7] Boundary conditions will be imposed on the discrete equations later.

The external force vector is written as

$$\mathbf{f}_I^{\text{ext}} = \sum_{p=1}^{n_p} m_p \phi_I(\mathbf{x}_p)\mathbf{b}(\mathbf{x}_p) + \sum_{p=1}^{n_p} m_p \phi_I(\mathbf{x}_p)\bar{\mathbf{t}}^s(\mathbf{x}_p)h^{-1} \tag{2.35}$$

and the internal force vector as

$$\mathbf{f}_I^{\text{int}} = -\sum_{p=1}^{n_p} m_p/\rho_p \boldsymbol{\sigma}_p \nabla\phi_I(\mathbf{x}_p) = -\sum_{p=1}^{n_p} V_p \boldsymbol{\sigma}_p \nabla\phi_I(\mathbf{x}_p) \tag{2.36}$$

where $\nabla\phi_I = \left(\frac{\partial\phi_I}{\partial x_1}, \frac{\partial\phi_I}{\partial x_2}, \frac{\partial\phi_I}{\partial x_3}\right)^{\mathrm{T}}$ denotes the gradient of the shape function; V_p is the volume of particle p; $\boldsymbol{\sigma}_p$ is the 3×3 Cauchy stress matrix of particle p. Recall that the particle density is defined as the ratio of the particle mass to particle volume. Note that the definition of the nodal internal force is slightly different from the one in the nonlinear FE literature–there is no minus sign in FEM formulation. We decided to be consistent with the MPM notation so that confusions would not occur for beginners.

Remark 8 *Since the Neumann boundary Γ_t where the traction is prescribed is not accurately defined in MPM, it is difficult to compute the external force due to non-zero traction. And this difficulty applies to the enforcement of non-zero heat flux in thermal and thermo-mechanical analysis. We refer to Section 4.2.2 for a discussion on this topic.*

We have presented the derivation of Equation (2.33), the semi-discrete equation following the MPM way as Sulsky et al. [1994] did. In the next section, another derivation is provided to see the link to the updated Lagrangian finite element method.

2.4 MPM as FEM with particles as integration points

In what follows, we show that the MPM semi-discrete equation can be obtained directly from the updated Lagrangian FEM equations by considering the particles as the integration points. This derivation usually seems to be the most straightforward for readers with experiences in the FEM. More importantly, it shows one major drawback of the standard MPM: the quadrature approximation nature of the method. Note that this way of deriving the MPM equations is not suitable to obtain the GIMP and CPDI formulations.

The UL finite element mass matrix, internal force and external force vectors are given by [Belytschko et al., 2000]

$$\begin{aligned} m_{IJ} &= \int_\Omega \rho\phi_I\phi_J \,\mathrm{d}\Omega \\ \mathbf{f}_I^{\text{int}} &= -\int_\Omega \nabla\phi_I \boldsymbol{\sigma} \,\mathrm{d}\Omega \\ \mathbf{f}_I^{\text{ext}} &= \int_\Omega \rho\phi_I \mathbf{b} \,\mathrm{d}\Omega \end{aligned} \tag{2.37}$$

where we have skipped the traction terms in the external force for simplicity.

Integrals in Equation (2.37) are computed using *numerical integration* or *numerial quadrature* rules. The integrand is evaluated at a finite set of points called *integration points* and a weighted sum of these values is used to approximate the integral. For a function f, one writes

$$\int_\Omega f \,\mathrm{d}\Omega = \sum_g f(\mathbf{x}_g) w_g \tag{2.38}$$

in which w_g denotes the weight of integration point g and $\mathbf{x}_g$ is the position of this point. It can be shown that if the material points are taken as the integration points with the weights being its volumes then the FEM equations reduce to the MPM ones. For examples,

$$\mathbf{f}_I^{\text{int}} = -\int_\Omega \nabla\phi_I \boldsymbol{\sigma} \,\mathrm{d}\Omega = -\sum_p \nabla\phi_I(\mathbf{x}_p)\boldsymbol{\sigma}_p V_p \tag{2.39}$$

which is exactly Equation (2.36).

As only the particle positions and volumes are updated, the deformed domain cannot be tiled without gaps as in the FEM, cf. Fig. 9. Thus, the integration measure is not exactly preserved in the MPM. In other words, the sum of the particle volumes

is not exactly equal to the volume of the deformed domain. Therefore, the error due to numerical quadrature in the MPM (more precisely the standard MPM formulation) is significant. And this is just the first source of quadrature error in MPM. Understanding this issue is crucial to reduce this error in developing methods. This is discussed in subsequent sections of this paper.

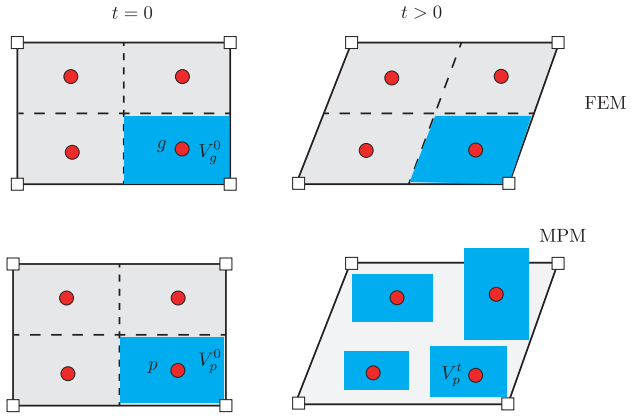


Figure 9: Numerical quadratures in the FEM and MPM: quadrature measure error in MPM.

The second source of quadrature error emerges from the fact that particles are independently located with respect to the background grid. Therefore, the particle based quadrature does not respect the continuity of the grid functions or the support of these functions. This is similar to other Galerkin meshfree method, see e.g., Dolbow and Belytschko [1999]. In the context of the MPM, Steffen et al. [2008b] presented a detailed analysis of this quadrature error.

Solutions to the quadrature error of the MPM include the utilization of smooth grid basis functions such as GIMP, B-spline, CPDI, or the adoption of finite element quadrature rules (commonly used in the geotechnical engineering community with unstructured background grids) and the FEMLIP of Moresi et al. [2003, 2007] where the quadrature weights are calculated on the fly such that affine functions can be exactly re-constructed. Among these techniques, CPDI is the most accurate in terms of quadrature.

2.5 Temporal discretization and resulting MPM algorithms

Up to this point, we have obtained the semi-discrete Equation 2.33. The full discrete equation is obtained by performing a time integration of this equation as described therein. This section also presents the corresponding MPM algorithms.

2.5.1 Lumped mass matrix

It is obvious from Equation (2.33) that, to obtain the acceleration one needs to solve a system of linear equations at every time step. The size of this system is $3n_n \times 3n_n$ in 3D where n_n – the number of nodes of the grid – can be a very large number. To avoid this, a lumped mass matrix is adopted. This lumped mass matrix is a diagonal matrix of which the diagonal terms are given by

$$m_I = \sum_{J=1}^{n_n} m_{IJ} = \sum_{J=1}^{n_n} \sum_{p=1}^{n_p} m_p \phi_I(\mathbf{x}_p)\phi_J(\mathbf{x}_p) = \sum_{p=1}^{n_p} m_p \phi_I(\mathbf{x}_p) \tag{2.40}$$

where Equation (2.34) was used in the second equality and the partition of unity (PU) property of FE shape functions, $\sum_J \phi_J(\mathbf{x}) = 1, \ \forall \mathbf{x}$, was used in the third equality.

Remark 9 *A lumped mass matrix is more than just a computational simplification; it also gives better results for impulsive loadings [Benson, 1992]. Note also that the use of a lumped mass matrix results in energy dissipation, see e.g., Zienkiewicz and Taylor [2006] (section 16.2.4).*

Remark 10 *Since the momentum equation is resolved at the grid nodes and not the material points, the conservation of mass must be satisfied at the nodes. By using Equation (2.40) one can write*

$$\sum_I m_I = \sum_I \left(\sum_{p=1}^{n_p} m_p \phi_I(\mathbf{x}_p) \right) = \sum_{p=1}^{n_p} m_p \left(\sum_I \phi_I(\mathbf{x}_p) \right) = \sum_p m_p \tag{2.41}$$

which proved that mass is conserved at grid nodes as well.

With this lumped mass matrix, one gets the following system of ordinary differential equations (ODE) in time:

$$m_I \mathbf{a}_I(t) = \mathbf{f}_I(t) := \mathbf{f}_I^{\text{ext}}(t) + \mathbf{f}_I^{\text{int}}(t) \tag{2.42a}$$

$$\mathbf{a}_I = \frac{d\mathbf{v}_I(t)}{dt} \tag{2.42b}$$

for all the nodes I.

For time discretization, the temporal domain $0 \leq t \leq t_f$ is divided into n_T time steps with time increment $\Delta t = t_f / n_T$. Therefore, Equation (2.42) has to be satisfied for every time instances $t_k = k\Delta t$ with $k = 0, 1, \ldots, n_T$. The most straightforward method to advance the solution in time i.e., solving the semi-discrete equations, is an explicit formulation in which the solution is advanced in time from t (i.e., t_k) to $t + \Delta t$ (or t_{k+1}) without solving a system of linear algebra equations. The forward Euler method is such an scheme and will be discussed in Section 2.5.3. Note that explicit schemes demand the use of quite small time steps Δt for stability.

A word about notation is in order. We use the superscript t to denote quantities at time instant t which are known and superscript $t + \Delta t$ for unknown quantities at the next time instant.

In contrary to the FEM where the velocity field is stored at the nodes, the grid velocities are, in the MPM, nullified after every step when the grid is reset. Therefore, at the beginning of a time step t, one needs to project the particle velocities to the grid nodes to serve as the starting point for the time advancement. This crucial step is discussed in Section 2.5.2. Section 2.5.3 presents the first complete MPM algorithm named USL (update stress last) which is probably the most popular MPM algorithm. A slight modification of the USL and dubbed MUSL (modified USL) is given in Section 2.5.4 to enhance the stability of MPM simulations.

2.5.2 Calculation of nodal velocities (momenta)

At the beginning of every time step, the particle velocities need to be mapped to the nodes. This step is not present in the FEM and is necessary since the grid is reset at the end of a time step and the nodal velocities for that step are lost. More precisely, the nodal momenta are mapped from the particles using the shape functions [Burgess et al., 1992]

$$(m\mathbf{v})_I^t = \sum_p \phi_I(\mathbf{x}_p^t)(m\mathbf{v})_p^t \tag{2.43}$$

or the particle momenta are projected to the grid nodes.

In what follows, we prove that a weighted least square approximation is needed for the momenta projection (Equation (2.43)) as there are more particles than nodes. For the sake of presentation, let us consider a one dimensional grid made of one cell with two nodes of which velocities are denoted by v_i. Moreover, within this cell, there are two particles at x_1 and x_2.

The idea is to minimize the following function

$$J = \frac{1}{2}\left[m_1 (V_1 - (\phi_1(x_1)v_1 + \phi_2(x_1)v_2))^2 + m_2 (V_2 - (\phi_1(x_2)v_1 + \phi_2(x_2)v_2))^2 \right] \tag{2.44}$$

which is a weighted least squares where the weights are the particle mass. In the above equation, V_i are the particle velocities (not volumes) and m_i are the particle masses. Differentiating J with respect to v_1 and v_2 and making them zeros one obtains

$$\begin{aligned} [m_1 V_1 \phi_{11} + m_2 V_2 \phi_{12}] &= (\phi_{11}v_1 + \phi_{21}v_2) m_1 \phi_{11} + (\phi_{12}v_1 + \phi_{22}v_2) m_2 \phi_{12} \\ [m_1 V_1 \phi_{21} + m_2 V_2 \phi_{22}] &= (\phi_{11}v_1 + \phi_{21}v_2) m_1 \phi_{21} + (\phi_{12}v_1 + \phi_{22}m_2) m_2 \phi_{22} \end{aligned} \tag{2.45}$$

where use was made of the short notation $\phi_{ij} = \phi_i(x_j)$, the above equation can be rewritten as follows

$$\begin{bmatrix} \phi_{11}\phi_{11}m_1 + \phi_{12}\phi_{12}m_2 & \phi_{11}\phi_{21}m_1 + \phi_{22}\phi_{12}m_2 \\ \phi_{11}\phi_{21}m_1 + \phi_{12}\phi_{22}m_2 & \phi_{21}\phi_{21}m_1 + \phi_{22}\phi_{22}m_2 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} m_1 V_1 \phi_1(x_1) + m_2 V_2 \phi_1(x_2) \\ m_1 V_1 \phi_2(x_1) + m_2 V_2 \phi_2(x_2) \end{bmatrix} \tag{2.46}$$

which is a system of linear algebra equations to solve for v_1 and v_2. It can be observed that the coefficient matrix of this linear system is exactly the consistent mass matrix. To avoid the solution of such a system, a row sum lumping technique is used which results in the following

$$\begin{bmatrix} \phi_1(x_1)m_1 + \phi_1(x_2)m_2 & 0 \\ 0 & \phi_2(x_1)m_1 + \phi_2(x_2)m_2 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} m_1 V_1 \phi_1(x_1) + m_2 V_2 \phi_1(x_2) \\ m_1 V_1 \phi_2(x_1) + m_2 V_2 \phi_2(x_2) \end{bmatrix} \tag{2.47}$$

which can be generalized to obtain Equation (2.43) which concludes the proof.

Remark 11 *We are checking whether linear momentum is conserved with Equation (2.43). By using Equation (2.43) one can write*

$$\sum_I (m\mathbf{v})_I = \sum_I \left(\sum_{p=1}^{n_p} (m\mathbf{v})_p \phi_I(\mathbf{x}_p) \right) = \sum_{p=1}^{n_p} (m\mathbf{v})_p \left(\sum_I \phi_I(\mathbf{x}_p) \right) = \sum_p (m\mathbf{v})_p \tag{2.48}$$

which proved that linear momentum is conserved at grid nodes as well (as long as ϕ_I makes a partition of unity).

Remark 12 *Even though this projection of particle velocity has been used nearly in all MPM simulations, we will see later in Section 5.5.1 that it is not able to provide an exact projection of a linear velocity field for arbitrary particle positions.*

2.5.3 Standard formulation (USL)

Equation (2.42a) is used to obtain the nodal accelerations $\mathbf{a}_I^t = \mathbf{f}_I^t / m_I^t$, and then the nodal velocity field is updated using the explicit Euler forward method as follows

$$\mathbf{v}_I^{t+\Delta t} = \mathbf{v}_I^t + \Delta t \mathbf{a}_I^t \tag{2.49}$$

where $\mathbf{v}_I^t$ denotes the nodal velocity at time t, which is known; Δt is the time increment. Theoretically, the nodes are moved to the new positions given by

$$\mathbf{x}_I^{t+\Delta t} = \mathbf{x}_I^t + \Delta t \mathbf{v}_I^{t+\Delta t} \tag{2.50}$$

Note that this nodal position update is rarely realized as the grid would be reset in the beginning of the next time step anyway as done in most MPM implementations. However, it should be emphasized that grid resetting is not a requirement. Grid nodes can be updated using Equation (2.50) until the grid is distorted or the grid can be replaced with any other suitable grid.

Once the grid has been updated, the grid velocities are used to update the particle state including position, velocity, volume, deformation gradient, stresses etc. We discuss the update of the particle positions and velocities first as there are different options which might confuse new comers to the field. In the PIC way, the particle velocity is obtained using the total grid velocity. On the other hand, according to the FLIP way, the particle velocity is obtained using the grid velocity increment. These are summarized in the following equations

$$\text{PIC:} \quad \mathbf{v}_p^{t+\Delta t} = \sum_I \phi_I(\mathbf{x}_p^t) \mathbf{v}_I^{t+\Delta t} \tag{2.51}$$

$$\text{FLIP:} \quad \mathbf{v}_p^{t+\Delta t} = \mathbf{v}_p^t + \sum_I \phi_I(\mathbf{x}_p^t) \left[\mathbf{v}_I^{t+\Delta t} - \mathbf{v}_I^t \right] \tag{2.52}$$

Following Stomakhin et al. [2013], Leroch et al. [2018], a mix of PIC and FLIP is used here for the particle velocity

$$\mathbf{v}_p^{t+\Delta t} = \alpha \left(\mathbf{v}_p^t + \sum_I \phi_I(\mathbf{x}_p^t) \left[\mathbf{v}_I^{t+\Delta t} - \mathbf{v}_I^t \right] \right) + (1-\alpha) \sum_I \phi_I(\mathbf{x}_p^t) \mathbf{v}_I^{t+\Delta t} \tag{2.53}$$

$$\mathbf{x}_p^{t+\Delta t} = \mathbf{x}_p^t + \Delta t \sum_I \phi_I(\mathbf{x}_p^t) \mathbf{v}_I^{t+\Delta t} \tag{2.54}$$

where $\alpha = 1$ corresponds to the FLIP of Brackbill and Ruppel [1986] that overcomes the numerical dissipation of the PIC ($\alpha = 0$) . Note that Leroch et al. [2018] updated the particle position as $\mathbf{x}_p^{t+\Delta t} = \mathbf{x}_p^t + \Delta t \mathbf{v}_p^{t+\Delta t}$ which we found to give similar results to the standard way. We refer to Section 6.1.1 for illustrations on the influence of α.

Finally, particle stresses are updated. This is known as the update stress last (USL) formulation in the MPM literature. There exists other formulations as discussed in Remark 17. Depending on the constitutive models being used, one might need to compute the gradient deformation $\mathbf{F}$, the velocity gradient $\mathbf{L}$ and the rate of deformation $\mathbf{D}$ etc. For example, one may need to compute the particle velocity gradients, defined in Equation (2.11), then compute the gradient deformation using the relation $\dot{\mathbf{F}} = \mathbf{L}\mathbf{F}$, and finally using the continuity equation $\rho J = \rho_0$ to determine the updated volume. This is typically for hyperelastic solids. They are given by

$$\mathbf{L}_p^{t+\Delta t} \equiv \nabla \mathbf{v}_p^{t+\Delta t} = \sum_I \nabla \phi_I(\mathbf{x}_p^t)\mathbf{v}_I^{t+\Delta t} \tag{2.55a}$$

$$\frac{\mathbf{F}^{t+\Delta t} - \mathbf{F}^t}{\Delta t} = \mathbf{L}^{t+\Delta t}\mathbf{F}^t, \quad \Rightarrow \mathbf{F}_p^{t+\Delta t} = (\mathbf{I} + \mathbf{L}_p^{t+\Delta t}\Delta t)\mathbf{F}_p^t \tag{2.55b}$$

$$V_p^{t+\Delta t} = J V_p^0, \quad J = \det \mathbf{F}_p^{t+\Delta t} \tag{2.55c}$$

$$\rho_p^{t+\Delta t} = \rho_0 / J \tag{2.55d}$$

where $\mathbf{L}_p$ is a 3×3 matrix of which components are $L_{ij} = v_{i,j}$ (in three dimensions). In the above equation, $\mathbf{I}$ is the identity matrix, and $\mathbf{F}_p$ is a 3×3 matrix with the initial matrix being $\mathbf{I}$ i.e., $\mathbf{F}_p^0 = \mathbf{I}$. We have added the equation to update the particle density, which is for example needed to calculate the equations of state.

For a hypoelastic constitutive model, one needs to compute the strain increment $\Delta \mathbf{e}_p = \left(\text{sym}\mathbf{L}_p^{t+\Delta t}\right)\Delta t$ and using it to compute the stress increment $\Delta \boldsymbol{\sigma}_p$. The updated particle stresses are given by

$$\boldsymbol{\sigma}_p^{t+\Delta t} = \boldsymbol{\sigma}_p^t + \Delta \boldsymbol{\sigma}_p \tag{2.56}$$

For complex constitutive models, it might be required to calculate other quantities to update the particle stresses. We refer to Appendix B for some common material models for elastic, hyperelastic and elasto-plastic solids and Section 5.4 for fluids and gaseous. As the material points are Lagrangian, existing stress update algorithms developed mainly by the FEM community can be readily reused in the MPM. We refer to the textbooks of Simo and Hughes [1998], de Souza Neto et al. [2011] for detail. And this brings us to the first complete explicit ULMPM algorithm given in Algorithm 1. As can be seen, it is a very simple algorithm, which can be coded straightforwardly. And yet, it has been used to solve many challenging solid mechanics problems.

Remark 13 *The velocity gradient* $\mathbf{L}$ *is actually computed as*

$$\mathbf{L} = \begin{bmatrix} \sum_I \phi_{I,x} v_{xI} & \sum_I \phi_{I,y} v_{xI} & \sum_I \phi_{I,z} v_{xI} \\ \sum_I \phi_{I,x} v_{yI} & \sum_I \phi_{I,y} v_{yI} & \sum_I \phi_{I,z} v_{yI} \\ \sum_I \phi_{I,x} v_{zI} & \sum_I \phi_{I,y} v_{xI} & \sum_I \phi_{I,z} v_{zI} \end{bmatrix} = \sum_I \begin{bmatrix} v_{xI} \\ v_{yI} \\ v_{zI} \end{bmatrix} \begin{bmatrix} \phi_{I,x} & \phi_{I,y} & \phi_{I,z} \end{bmatrix} \tag{2.57}$$

for 3D problems. Simplifications for 1D and plane strain/stress 2D problems are straightforward. For axi-symmetric problems, see Appendix G.

Algorithm 1 Solution procedure of explicit MPM (USL, cut-off).

Initialization
  Set up the Cartesian grid, set time $t = 0$
  Set up particle data: $\mathbf{x}_p^0, \mathbf{v}_p^0, \boldsymbol{\sigma}_p^0, \mathbf{F}_p^0, V_p^0, m_p, \rho_p^0$
end
while $t < t_f$ **do**
  Reset grid quantities: $m_I^t = 0, (m\mathbf{v})_I^t = \mathbf{0}, \mathbf{f}_I^{\text{ext},t} = \mathbf{0}, \mathbf{f}_I^{\text{int},t} = \mathbf{0}$
  Mapping from particles to nodes (P2G)
    Compute nodal mass $m_I^t = \sum_p \phi_I(\mathbf{x}_p^t) m_p$
    Compute nodal momentum $(m\mathbf{v})_I^t = \sum_p \phi_I(\mathbf{x}_p^t)(m\mathbf{v})_p^t$
    Compute external force $\mathbf{f}_I^{\text{ext},t} = \sum_p \phi_I(\mathbf{x}_p) m_p \mathbf{b}(\mathbf{x}_p)$
    Compute internal force $\mathbf{f}_I^{\text{int},t} = -\sum_p V_p^t \boldsymbol{\sigma}_p^t \nabla\phi_I(\mathbf{x}_p^t)$
    Compute nodal force $\mathbf{f}_I^t = \mathbf{f}_I^{\text{ext},t} + \mathbf{f}_I^{\text{int},t}$
  end
  Update the momenta $(m\mathbf{v})_I^{t+\Delta t} = (m\mathbf{v})_I^t + \mathbf{f}_I^t \Delta t$
  Fix Dirichlet nodes I e.g., $(m\mathbf{v})_I^t = \mathbf{0}$ and $(m\mathbf{v})_I^{t+\Delta t} = \mathbf{0}$
  Update particles (G2P)
    Get nodal velocities $\mathbf{v}_I^t = (m\mathbf{v})_I^t / m_I^t$ and $\mathbf{v}_I^{t+\Delta t} = (m\mathbf{v})_I^{t+\Delta t} / m_I^t$
    Update particle velocities $\mathbf{v}_p^{t+\Delta t} = \alpha\left(\mathbf{v}_p^t + \sum_I \phi_I(\mathbf{x}_p^t)\left[\mathbf{v}_I^{t+\Delta t} - \mathbf{v}_I^t\right]\right) + (1-\alpha)\sum_I \phi_I(\mathbf{x}_p^t)\mathbf{v}_I^{t+\Delta t}$
    Update particle positions $\mathbf{x}_p^{t+\Delta t} = \mathbf{x}_p^t + \Delta t \sum_I \phi_I(\mathbf{x}_p^t)\mathbf{v}_I^{t+\Delta t}$
    Compute gradient velocity $\mathbf{L}_p^{t+\Delta t} = \sum_I \nabla\phi_I(\mathbf{x}_p^t)\mathbf{v}_I^{t+\Delta t}$
    Updated gradient deformation tensor $\mathbf{F}_p^{t+\Delta t} = (\mathbf{I} + \mathbf{L}_p^{t+\Delta t}\Delta t)\mathbf{F}_p^t$
    Update volume $V_p^{t+\Delta t} = \det \mathbf{F}_p^{t+\Delta t} V_p^0$
    Compute the rate of deformation matrix $\mathbf{D}_p^{t+\Delta t} = 0.5(\mathbf{L}_p^{t+\Delta t} + (\mathbf{L}_p^{t+\Delta t})^{\mathrm{T}})$
    Compute the strain increment $\Delta\boldsymbol{\epsilon}_p = \Delta t \mathbf{D}_p^{t+\Delta t}$
    Update stresses: $\boldsymbol{\sigma}_p^{t+\Delta t} = \boldsymbol{\sigma}_p^t + \Delta\boldsymbol{\sigma}_p(\Delta\boldsymbol{\epsilon}_p)$, or $\boldsymbol{\sigma}_p^{t+\Delta t} = \boldsymbol{\sigma}_p^{t+\Delta t}(\mathbf{F}_p^{t+\Delta t})$
  end
  Advance time $t = t + \Delta t$
  Error calculation: if needed (e.g., for convergence tests)
end while

Remark 14 *There are some remarks on Algorithm 1. First, we have omitted the contribution to the external force due to non-zero traction. This is because it is more difficult to deal with than with the FEM as discussed in Section 4.2.2. Second, we have assumed that a constant time step $\Delta t = const$ was used. In practice, varying time steps are often adopted, see Section 2.7 for details. Finally, we presented the so-called momentum formulation. Note that this momentum formulation is very common but it does not improve the stability of the MPM.*

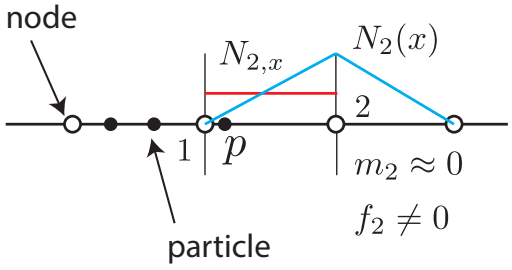


Figure 10: Troubled nodes with nearly zero mass resulting in infinite acceleration (node 2).

Small mass issue. There is a numerical issue in this formulation: the division operator in $\mathbf{a}_I^t = \mathbf{f}_I^t / m_I^t$ would yield infinite acceleration if the mass m_I^t is small. In turn, the velocity gradient in Equation (2.55a) would be infinite as well and this would spoil the particle stresses [Sulsky et al., 1995]. This happens when a particle is very close to a node which has only one particle within its support, cf. Fig. 10. This usually happens with nodes located close to the material interface. For multiple

bodies simulations, those interface nodes are often in contact and therefore contact algorithms have to carefully address the small nodal mass issue.

In order to identify the trouble more clearly, we turn to the example given in Fig. 10. We assume that there is only one element and one single particle, with mass denoted by m_p, located very close to node 1. Furthermore, we assume the old nodal velocities are zero i.e., $v_1^t = v_2^t = 0$.

The nodal masses are given by

$$\begin{aligned} m_1 &= m_p N_1 \\ m_2 &= m_p N_2 \quad \text{(small value)} \end{aligned} \tag{2.58}$$

And the nodal accelerations are thus

$$\begin{aligned} a_1 &= f_1/m_1 \\ a_2 &= f_2/m_2 \quad \text{(very large value)} \end{aligned} \tag{2.59}$$

where f_1 and f_2 are the nodal forces. The updated nodal velocities are given by using Equation (2.49)

$$\begin{aligned} v_1^{t+\Delta t} &= \Delta t f_1/m_1 \\ v_2^{t+\Delta t} &= \Delta t f_2/m_2 \quad \text{(very large value)} \end{aligned} \tag{2.60}$$

The updated particle position and velocity are given by according to Equations (2.53) and (2.54) with $\alpha = 1$

$$\begin{aligned} v_p^{t+\Delta t} &= v_p^t + \Delta t(N_1 f_1/m_1 + N_2 f_2/m_2) = v_p^t + \Delta t(f_1/m_p + f_2/m_p) \\ x_p^{t+\Delta t} &= x_p^t + \Delta t(N_1 \Delta t f_1/m_1 + N_2 \Delta t f_2/m_2) = x_p^t + \Delta t^2(f_1/m_p + f_2/m_p) \end{aligned} \tag{2.61}$$

where Equation (2.58) was used for the nodal masses m_1 and m_2. Note that the updated particle position and velocity are normal as they are smoothed out by the shape functions.

Next, one computes the velocity gradient needed for stress updating

$$L_p^{t+\Delta t} = -\frac{1}{h} v_1^{t+\Delta t} + \frac{1}{h} v_2^{t+\Delta t} \quad \text{(very large value)} \tag{2.62}$$

In conclusion, the problem lies in the use of $v_2^{t+\Delta t}$ in computing the velocity gradient but not in updating the particle velocity and position. A technique to solve this issue, proposed by [Sulsky et al., 1995], will be presented in Section 2.5.4.

Cutoff technique. In this technique, a small positive threshold is introduced to cure the small mass issue. Accordingly, the nodal velocities are computed as

$$\mathbf{v}_I^{t+\Delta t} = \begin{cases} \dfrac{(m\mathbf{v})_I^{t+\Delta t}}{m_I^t} & \text{if } m_I^t > tol \\ 0 & \text{otherwise} \end{cases} \tag{2.63}$$

This algorithm requires an extra parameter (a cutoff value). Yet how to chose it is not clear. Even if a good cutoff value can be chosen, it produces an undesirable constraint which should not be in the system. More advanced techniques are presented in what follows.

Remark 15 *Wallstedt and Guilkey [2008] have studied a number of families of time integration schemes for use with GIMP including Runge Kutta, Runge-Kutta-Nystrom, Adams-Bashforth-Moulton (ABM), and Predictor-Corrector Newmark methods. They reported that few of these methods have been able to achieve their formal orders of accuracy. Not only is the MPM used for highly discontinuous and nonlinear problems but the spatial error of the method tend to overwhelm any improvement that a temporally high order method might offer. They also showed that the central difference scheme, commonly used in nonlinear finite element codes [Belytschko et al., 2000], is exactly the same as USL but for one crucial difference: initialization of particle velocity to a negative half time step.*

Remark 16 *As can be seen from Algorithm 1, the MPM algorithm is very similar to the ULFEM, cf. Algorithm 14 in Appendix F.1. There are just a few differences. First, the nodal mass and velocity have to be re-calculated at the beginning of every time step. This is natural as the grid does not store permanent information. Second, one needs to update the particle's position and velocity. In the ULFEM, the integration points' position are fixed and one does not need to calculate their velocity. Based on this observation, per time step, the MPM is about 2 to 3 times slower than the ULFEM. Yet, for very large deformation problems, Ma et al. [2009b] showed that their in-house MPM code is much faster than the commercial LS-DYNA FEM.*

2.5.4 Modified update stress last (MUSL)

In the MUSL of [Sulsky et al., 1995], the updated particle velocities are mapped back to the nodes to get the nodal velocities using

$$(m\mathbf{v})_I^{t+\Delta t} = \sum_p \phi_I(\mathbf{x}_p)(m\mathbf{v})_p^{t+\Delta t} \tag{2.64}$$

and thus

$$\mathbf{v}_I^{t+\Delta t} = \frac{(m\mathbf{v})_I^{t+\Delta t}}{m_I^t} = \frac{\sum_p \phi_I(\mathbf{x}_p)(m\mathbf{v})_p^{t+\Delta t}}{\sum_p \phi_I(\mathbf{x}_p) m_p} = \frac{\sum_p \phi_I(\mathbf{x}_p)(m\mathbf{v})_p^{t+\Delta t}}{m_I^t} \tag{2.65}$$

In the second equality, the appearance of the shape functions in both numerator and denominator cancels out its role and the numerical problem regarding very large velocity gradient is thus cured. Applying this to the example given in Fig. 10, the updated velocity at the troubled node is now given by

$$v_2^{t+\Delta t} = (1/m_2)\left[\Delta t\left(\frac{f_1}{m_p} + \frac{f_2}{m_p}\right) m_p\right] N_2 = \Delta t\left(\frac{f_1}{m_p} + \frac{f_2}{m_p}\right) \tag{2.66}$$

where the first of Equation (2.61) was used with a simplification that $v_p^t = 0$. Apparently $v_2^{t+\Delta t}$ is not infinite and can be safely used for computing the gradient velocity. The resulting algorithm, dubbed MUSL, is given in Algorithm 2. Other name for this algorithm is the double mapping USL as the particle momenta are extrapolated to the nodes twice–at the beginning of the step and after updating the nodal momenta. We use a tilde $\tilde{\square}$ to denote the temporary grid velocities (Line 14). The differences of MUSL compared with USL are in Lines 16 to 21.

Remark 17 *It was Bardenhagen [2002] who introduced the term Update Stress Last (USL) and presented an Update Stress First (USF) algorithm where stresses are updated in the P2G step. He found that while USL is dissipative (i.e., suffers from numerical dissipation), USF conserves energies well. Nairn [2003] analyzed the USF and MUSL for a 2D elastic vibration problem. He found that USL is very unstable, that the MUSL approach slowly dissipated energy while the USF approach slowly increased energy. Wallstedt and Guilkey [2008] used the method of manufactured solutions to test temporal and spatial convergence of GIMP with USF and USL. Their results show that USL is superior in terms of stability and convergence.*

Remark 18 *We have so far presented just explicit dynamics MPM. For implicit dynamics and quasi-static MPM formulations, please refer to the discussion in Section 1.1.4. It is quite straightforward to develop these algorithms as the MPM is very similar to the updated Lagrangian FEM. That is expressions for the geometric and material tangent matrices developed for ULFEM can be readily reused, see Belytschko et al. [2000]. A very clear presentation of an implicit dynamics MPM is given in Iaconeta et al. [2017].*

Remark 19 *An explicit MPM code can also be used for quasi-static problems. Even though using an explicit code for simple static simulations is not efficient (due to many time steps for a long simulation period), an explicit code is the only option for challenging static simulations where implicit solvers would crash (e.g., the solver does not converge). Global and local damping can be added to mitigate the effects of stress waves for static simulations [Al-Kafaji, 2013].*

Algorithm 2 Solution procedure of explicit MPM (MUSL).

Initialization
  Set up the Cartesian grid, set time $t = 0$
  Set up particle data: $\mathbf{x}_p^0, \mathbf{v}_p^0, \boldsymbol{\sigma}_p^0, \mathbf{F}_p^0, V_p^0, m_p, \rho_p^0$
end
while $t < t_f$ **do**
  Reset grid quantities: $m_I^t = 0, (m\mathbf{v})_I^t = \mathbf{0}, \mathbf{f}_I^{\text{ext},t} = \mathbf{0}, \mathbf{f}_I^{\text{int},t} = \mathbf{0}$
  Mapping from particles to nodes (P2G)
    Compute nodal mass $m_I^t = \sum_p \phi_I(\mathbf{x}_p^t) m_p$
    Compute nodal momentum $(m\mathbf{v})_I^t = \sum_p \phi_I(\mathbf{x}_p^t)(m\mathbf{v})_p^t$
    Compute external force $\mathbf{f}_I^{\text{ext},t} = \sum_p \phi_I(\mathbf{x}_p) m_p \mathbf{b}(\mathbf{x}_p)$
    Compute internal force $\mathbf{f}_I^{\text{int},t} = -\sum_p V_p^t \boldsymbol{\sigma}_p^t \nabla \phi_I(\mathbf{x}_p^t)$
    Compute nodal force $\mathbf{f}_I^t = \mathbf{f}_I^{\text{ext},t} + \mathbf{f}_I^{\text{int},t}$
  end
  Update the momenta $(m\tilde{\mathbf{v}})_I^{t+\Delta t} = (m\mathbf{v})_I^t + \mathbf{f}_I^t \Delta t$
  Fix Dirichlet nodes I e.g., $(m\mathbf{v})_I^t = \mathbf{0}$ and $(m\tilde{\mathbf{v}})_I^{t+\Delta t} = \mathbf{0}$
  Update particle velocities and grid velocities (double mapping)
    Get nodal velocities $\tilde{\mathbf{v}}_I^{t+\Delta t} = (m\tilde{\mathbf{v}})_I^{t+\Delta t} / m_I^t$
    Update particle positions $\mathbf{x}_p^{t+\Delta t} = \mathbf{x}_p^t + \Delta t \sum_I \phi_I(\mathbf{x}_p^t) \tilde{\mathbf{v}}_I^{t+\Delta t}$
    Update particle velocities $\mathbf{v}_p^{t+\Delta t} = \alpha\left(\mathbf{v}_p^t + \sum_I \phi_I(\mathbf{x}_p^t)\left[\tilde{\mathbf{v}}_I^{t+\Delta t} - \mathbf{v}_I^t\right]\right) + (1-\alpha)\sum_I \phi_I(\mathbf{x}_p^t)\tilde{\mathbf{v}}_I^{t+\Delta t}$
    Update grid velocities $(m\mathbf{v}_I)^{t+\Delta t} = \sum_p \phi_I(\mathbf{x}_p^t)(m\mathbf{v})_p^{t+\Delta t}$
    Fix Dirichlet nodes $(m\mathbf{v})_I^{t+\Delta t} = \mathbf{0}$
  end
  Update particles (G2P)
    Get nodal velocities $\mathbf{v}_I^{t+\Delta t} = (m\mathbf{v})_I^{t+\Delta t} / m_I^t$
    Compute gradient velocity $\mathbf{L}_p^{t+\Delta t} = \sum_I \nabla \phi_I(\mathbf{x}_p^t) \mathbf{v}_I^{t+\Delta t}$
    Updated gradient deformation tensor $\mathbf{F}_p^{t+\Delta t} = (\mathbf{I} + \mathbf{L}_p^{t+\Delta t} \Delta t)\mathbf{F}_p^t$
    Update volume $V_p^{t+\Delta t} = \det \mathbf{F}_p^{t+\Delta t} V_p^0$
    Update stresses $\boldsymbol{\sigma}_p^{t+\Delta t} = \boldsymbol{\sigma}_p^t + \Delta \boldsymbol{\sigma}_p$
  end
  Advance time $t = t + \Delta t$
  Error calculation: if needed (e.g., for convergence tests)
end while

2.6 Total Lagrangian MPM

A total Lagrangian MPM (TLMPM) was presented in de Vaucorbeil et al. [2019] where the stress and strain are Lagrangian, i.e., they are defined with respect to the reference configuration (for example, the 1st PK stress is employed), the spatial derivatives are computed with respect to the material coordinates. The corresponding weak form therefore involves integrals over the reference configuration. The result is a very efficient and easy to implement MPM that does not suffer cell-crossing error and numerical fracture. Furthermore, the TLMPM has a better quadrature approximation since the particles are always located at the optimal quadrature points. For all that, the inherent no-slip no-penetration contact capability in the ULMPM ceases to exist for the TLMPM.

The algorithm is nearly identical to the standard MPM, see Algorithm 3. The differences lie in (1) the 1st PK stress is employed in the internal force, (2) the spatial derivatives are computed with respect to $\mathbf{X}_p$ (rather than $\mathbf{x}_p^t$), and (3) the deformation gradient and the velocity gradient are calculated differently. Furthermore, the nodal mass, the weighting functions, and gradients are computed once. And for some consitutive models adopting the Cauchy stress, one might need to convert it to the 1st PK stress.

The deformation gradient can be computed as follows

$$\dot{\mathbf{F}}_p = \frac{\mathbf{F}_p^{t+\Delta t} - \mathbf{F}^t}{\Delta t} \Rightarrow \mathbf{F}_p^{t+\Delta t} = \mathbf{F}_p^t + \Delta t \dot{\mathbf{F}}_p, \quad \dot{\mathbf{F}}_p = \sum_I \nabla_0 \phi_I(\mathbf{X}_p) \mathbf{v}_I^{t+\Delta t} \tag{2.67}$$

or alternatively, $\mathbf{F}$ can be computed using the relation $\mathbf{F} = \mathbf{I} + \frac{\partial \mathbf{u}}{\partial \mathbf{X}}$:

$$\mathbf{F}_p^{t+\Delta t} = \mathbf{I} + \sum_I \nabla_0 \phi_I(\mathbf{X}_p)(\mathbf{x}_I^{t+\Delta t} - \mathbf{X}_I) \tag{2.68}$$

Our experiences show that the two ways yield identical results.

The velocity gradient $\mathbf{L}$ is then computed as

$$\mathbf{L} := \frac{\partial \mathbf{v}}{\partial \mathbf{x}} = \frac{\partial \mathbf{v}}{\partial \mathbf{X}} \frac{\partial \mathbf{X}}{\partial \mathbf{x}} = \dot{\mathbf{F}} \mathbf{F}^{-1} \tag{2.69}$$

From which one can compute the strain rate $\mathbf{D}$.

Remark 20 *The rate of the deformation gradient $\dot{\mathbf{F}}$ is actually computed as*

$$\dot{\mathbf{F}} = \begin{bmatrix} \sum_I \phi_{I,X} v_{xI} & \sum_I \phi_{I,Y} v_{xI} & \sum_I \phi_{I,Z} v_{xI} \\ \sum_I \phi_{I,X} v_{yI} & \sum_I \phi_{I,Y} v_{yI} & \sum_I \phi_{I,Z} v_{yI} \\ \sum_I \phi_{I,X} v_{zI} & \sum_I \phi_{I,Y} v_{xI} & \sum_I \phi_{I,Z} v_{zI} \end{bmatrix} = \sum_I \begin{bmatrix} v_{xI} \\ v_{yI} \\ v_{zI} \end{bmatrix} \begin{bmatrix} \phi_{I,X} & \phi_{I,Y} & \phi_{I,Z} \end{bmatrix} \tag{2.70}$$

for 3D problems.

Remark 21 *Similar to the ULMPM, the TLMPM can be derived following two ways. In the first way, one can start from the strong form of the governing equations in the TL form and develop the corresponding weak form. In the second way, one can directly use the TLFEM semi-discrete equations and use the particles as the integration points. Since we have done these steps for the ULMPM, we do not repeat them for the TLMPM.*

Remark 22 *Even though the TLMPM is very similar to the TLFEM, there are subtle differences. First, the TLMPM does not need a mesh conforming to the solid under consideration. Second, modeling contact can be done in the spirit of particle methods. Third, the TLMPM provides an ideal testbed for developing high order MPM algorithms as it eliminates many issues of the ULMPM.*

Algorithm 3 Solution procedure of explicit TLMPM (MUSL).

Initialization
  Set up particle data: $\mathbf{X}_p, \mathbf{v}_p^0, \boldsymbol{\sigma}_p^0, \mathbf{F}_p^0, V_p^0, m_p, \rho_p^0$
  Compute nodal mass $m_I = \sum_p \phi_I(\mathbf{X}_p) m_p$
  Compute and store weighting and gradient $\phi_I(\mathbf{X}_p)$ and $\nabla_0 \phi_I(\mathbf{X}_p)$
end
while $t < t_f$ **do**
  Reset grid quantities: $m_I^t = 0, (m\mathbf{v})_I^t = \mathbf{0}, \mathbf{f}_I^{\text{ext},t} = \mathbf{0}, \mathbf{f}_I^{\text{int},t} = \mathbf{0}$
  Mapping from particles to nodes (P2G)
    Compute nodal momentum $(m\mathbf{v})_I^t = \sum_p \phi_I(\mathbf{X}_p)(m\mathbf{v})_p^t$
    Compute external force $\mathbf{f}_I^{\text{ext},t}$
    Compute internal force $\mathbf{f}_I^{\text{int},t} = -\sum_{p=1}^{n_p} V_p^0 \boldsymbol{P}_p^t \nabla_0 \phi_I(\mathbf{X}_p)$
    Compute nodal force $\mathbf{f}_I^t = \mathbf{f}_I^{\text{ext},t} + \mathbf{f}_I^{\text{int},t}$
  end
  Update the momenta $(m\tilde{\mathbf{v}})_I^{t+\Delta t} = (m\mathbf{v})_I^t + \mathbf{f}_I^t \Delta t$
  Fix Dirichlet nodes I e.g., $(m\tilde{\mathbf{v}})_I^{t+\Delta t} = \mathbf{0}$ and $(m\mathbf{v})_I^t = \mathbf{0}$
  Update particle velocities and grid velocities (double mapping)
    Get nodal velocities $\tilde{\mathbf{v}}_I^{t+\Delta t} = (m\tilde{\mathbf{v}})_I^{t+\Delta t} / m_I^t$
    Update particle velocities $\mathbf{v}_p^{t+\Delta t} = \alpha\left(\mathbf{v}_p^t + \sum_I \phi_I(\mathbf{X}_p)\left[\tilde{\mathbf{v}}_I^{t+\Delta t} - \mathbf{v}_I^t\right]\right) + (1-\alpha)\sum_I \phi_I(\mathbf{X}_p)\tilde{\mathbf{v}}_I^{t+\Delta t}$
    Update grid velocities $(m\mathbf{v}_I)^{t+\Delta t} = \sum_p \phi_I(\mathbf{X}_p)(m\mathbf{v})_p^{t+\Delta t}$
    Fix Dirichlet nodes $(m\mathbf{v})_I^{t+\Delta t} = \mathbf{0}$
  end
  Update particle (G2P)
    Compute $\dot{\mathbf{F}}_p^{t+\Delta t} = \sum_I \nabla_0 \phi_I(\mathbf{X}_p)\mathbf{v}_I^{t+\Delta t}$
    Updated gradient deformation tensor $\mathbf{F}_p^{t+\Delta t} = \mathbf{F}_p^t + \Delta t \dot{\mathbf{F}}_p^{t+\Delta t}$
    Velocity gradient $\mathbf{L}_p^{t+\Delta t} = \dot{\mathbf{F}}_p^{t+\Delta t}(\mathbf{F}_p^{t+\Delta t})^{-1}$
    Update stresses $\boldsymbol{\sigma}_p^{t+\Delta t} = \boldsymbol{\sigma}_p^t + \Delta \boldsymbol{\sigma}_p$
    Covert stresses to 1st PK stresses $\mathbf{P}_p^{t+\Delta t} = g(\boldsymbol{\sigma}_p^{t+\Delta t})$ using Table 4
    Update particle positions (for visualization) $\mathbf{x}_p^{t+\Delta t} = \mathbf{x}_p^t + \Delta t \sum_I \phi_I(\mathbf{X}_p)\mathbf{v}_I^{t+\Delta t}$
  end
end while

2.7 Adaptive time step

As explicit time integrations are only conditionally stable, explicit dynamics MPM must employ a time step smaller than a critical value so that errors will not be so amplified from time step to time step that the error will quickly swamp the solution. In typical explicit MPM simulations, an adaptive time step is employed i.e., the time step is adjusted according to the particle velocities instead of being fixed. One first computes the dilatational wave speed c_{dil}:

$$c_{\text{dil}} = \sqrt{\frac{\lambda + 2\mu}{\rho}} = \sqrt{\frac{K + \frac{4}{3}G}{\rho}} \tag{2.71}$$

where λ, μ are the Lamé constants and K is the bulk modulus and $G = \mu$ denotes the shear modulus.

Next, one computes the maximum wave speed using the following equation [Anderson Jr, 1987]

$$\mathbf{c} = (\max_p(c_{\text{dil}} + |v_{xp}|), \max_p(c_{\text{dil}} + |v_{yp}|), \max_p(c_{\text{dil}} + |v_{zp}|)) \tag{2.72}$$

where v_{xp} is the x component of particle p's velocity. For hyper-velocity impact problems, the above equation, where the particle velocity is taken into account, is very much needed.

The time step Δt is then chosen as follows

$$\Delta t = \alpha \min\left(\frac{h_x}{c_x}, \frac{h_y}{c_y}, \frac{h_z}{c_z}\right) \tag{2.73}$$

where (h_x, h_y, h_z) are the cell spacings and α is a time step multiplier ranging from 0 to 1.This factor is needed as the stability analysis was done for linear problems. The above formulation was implemented in the Uintah MPM code.

Remark 23 *When a model contains a few very stiff elements, the efficiency of explicit time integration is severely compromised. This is because the time step of the entire model is decided by these very stiff elements. Sub-cycling is a technique where the problem is divided into a number of sub-domains and each sub-domain is integrated in time with its own stable time steps, see [Belytschko et al., 2000]. This technique has just recently been taken in the computer graphics and they introduced the so-called asynchronous material point method [Hu and Fang, 2017].*

Remark 24 *The formulation presented so far can be used for either 1D, plane stress/strain 2D or 3D problems. For axisymmetric problems, slight modifications are needed and we refer to Appendix G for details.*

We have presented the basic MPM formulation for explicit solid dynamics. This MPM formulation, as simple as it is, can simulation collision of solids involving large deformation where the contact is no-slip, see Fig. 8. Frictional contact can be incorporated into this model quite straightforwardly (see Section 5.1). However, no details about $\phi_I(\mathbf{x})$ and $\nabla\phi_I(\mathbf{x})$ are specified yet. The next section is devoted to this topic.

3 Various MPM formulations

A general framework for the MPM has been presented in Section 2 in which the shape functions have not yet been specified. In this section, various shape functions ranging from the standard hat functions (or linear Lagrange functions), see Section 3.2, generalized interpolation MPM (Section 3.3), B-splines (Section 3.4), Bernstein functions (Section 3.5), convected particle domain integrator (Section 3.6) are discussed.

A note on terminology of $\phi_I(\mathbf{x})$ and $\nabla\phi_I(\mathbf{x})$ is worthy here. As will be seen in Section 3.3, $\phi_I(\mathbf{x})$ is constructed as a convolution of the linear/bilinear/trilinear FE shape functions $N_I(\mathbf{x})$ with the particle characteristic function. Therefore, it is not rigorous to call $\phi_I(\mathbf{x})$ grid basis functions. Therefore, in the remaining of this manuscript, $\phi_I(\mathbf{x})$ is referred to as weighting function and $\nabla\phi_I(\mathbf{x})$ the weighting gradient.

Remark 25 *We restrict, for now, our discussion to MPM formulations using a Cartesian grid. There exists MPM variants that adopt unstructured grids. See Section 4.4 for a discussion on this topic.*

3.1 Properties of weighting functions

The weighting functions $\phi_I(\mathbf{x})$ should satisfy all the following properties in addition to being smooth and continuous across the cell boundaries

Partition of Unity (PU) $\sum_I \phi_I(\mathbf{x}) = 1$ for all $\mathbf{x}$.

Compact support $\phi_I(\mathbf{x}) \neq 0$ for just points $\mathbf{x}$ close to node I.

Non-negativity $\phi_I(\mathbf{x}) \geq 0$ for all $\mathbf{x}$.

Kronecker delta property $\phi_I(\mathbf{x}_J) = \delta_{IJ}$.

The defines completeness (i.e., the ability to represent rigid motions and constant strains) which is required for convergence [Hughes, 2000]. Compact support is for efficiency and non-negativity ensures positive nodal mass when a lumped mass is used. The use of second-order finite elements implies the use of shape functions having negative values in their domain. This may lead to negative mass at some of the grid points, possibly causing instability of the solution scheme [Andersen and Andersen, 2010a]. The Kronecker delta property should be satisfied at least at the solid boundaries so that enforcement of Dirichlet boundary conditions is straightforward.

3.2 Standard linear basis functions

Although any grid can be used in the MPM, a Cartesian grid is usually chosen for computational convenience reasons. In order to avoid finding the natural coordinates of material points, if shape functions are defined in the parameter space, in MPM shape functions are conveniently defined in the global coordinate system. In 1D, the shape functions are defined as

$$N_I^x(x) = \begin{cases} 1 - |x - x_I|/h_x & \text{if } |x - x_I| \leq h_x \\ 0 & \text{else} \end{cases} \tag{3.1}$$

where h_x denotes the nodal spacing or element size in the x direction. Its derivatives are given by

$$N_{I,x}^x(x) \equiv \frac{dN_I^x(x)}{dx} = \begin{cases} -\text{sign}(x - x_I)/h_x & \text{if } |x - x_I| \leq h_x \\ 0 & \text{else} \end{cases} \tag{3.2}$$

where $\text{sign}(x)$ is the signum function.

For 2D, the shape functions are simply the tensor-product of the two shape functions along the x and y directions

$$N_I(x, y) = N_I^x(x) N_I^y(y) \tag{3.3}$$

In the first MPM, the weighting functions are simply this hat function i.e., $\phi_I(\mathbf{x}) = N_I(\mathbf{x})$. This MPM is refered to as the standard MPM in this text. An illustration of the hat shape functions is given in Fig. 11 for a series of three elements in 1D.

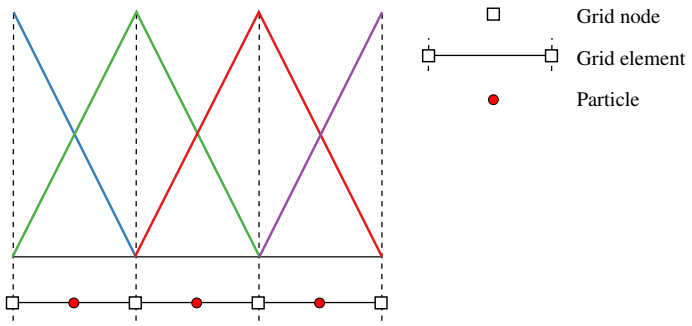


Figure 11: Hat shape functions for a series of three elements in 1D.

Cell crossing instability. The original MPM with C^0 shape functions suffers from the so-called cell crossing instability. To illustrate this phenomenon, consider a 1D MPM discretization shown in Fig. 12 in which all particles have the same stress (i.e., uniform stress state), the same volume and a uniform element size. Each element has two particles, Fig. 12a. The internal force at node 2 is identically zero in the absence of body force. When a particle has just moved to a new cell, Fig. 12b, the internal force at node 2 is non-zero resulting in non-equilibrium. In order to demonstrate this issue, we consider a simple one dimensional MPM simulation shown in Fig. 13a where two particles are moving with a constant velocity towards each other. The grid consists of 6 cells with cell size is 1/6. While the particle stresses are identically zero, before collision, and thus causing no issue as the particles travel, after collision they become non-zero. And right after the particles cross the cells, the stress field is spoiled and there is a sudden increase of the strain energy, cf Fig. 13b/c.

The cell-crossing issue is more severe in static analyses where there is no inertia force. For structure and material failure modeling where the onset of failure is most often based on stresses, cell-crossing issue must be completely avoided. Fine meshes are preferred for accuracy, but they are more prone to cell crossing. A simple but inefficient way to prevent this is to adopt more material points and smaller time steps, cf. Fig. 14 where three particles per cell are used. Another option is to not to reset the grid at the end of every steps as done in Guilkey et al. [2006]: particles never move out of the deformed grid. The resulting algorithm is very close to the ULFEM but the extent of mesh distortion is much lower as the initial mesh contains elements with right angle corners.

Better methods to mitigate this error include the use of high order B-splines basis functions, the generalized interpolation material point (GIMP) method (and its variants such as CPDI), the use of modified gradient of shape functions in the dual domain MPM [Zhang et al., 2011] and the total Lagrangian MPM (Section 2.6). All these methods also improve the quadrature of the MPM.

For quasi-static problems, Beuth et al. [2011] proposed to use Gauss integration to remove the cell-crossing error. The idea is that Gauss points never leave the elements. In their formulation, also used in later work of Jassim et al. [2013], one

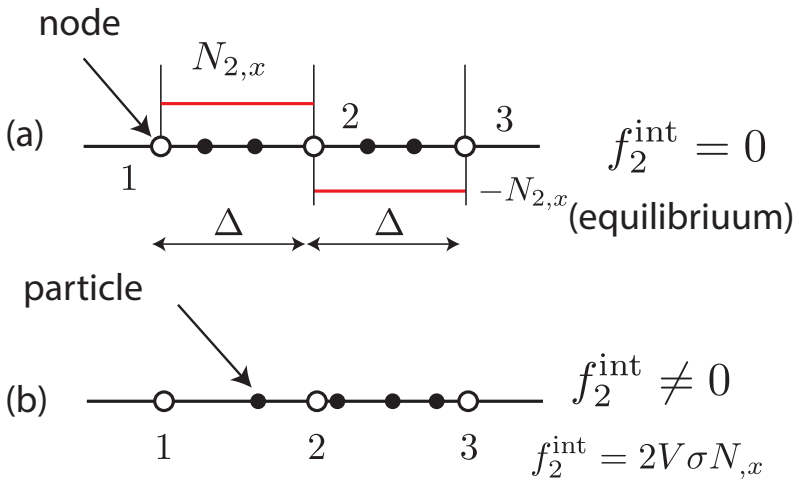
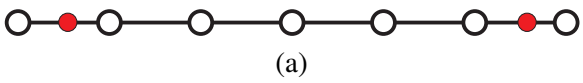
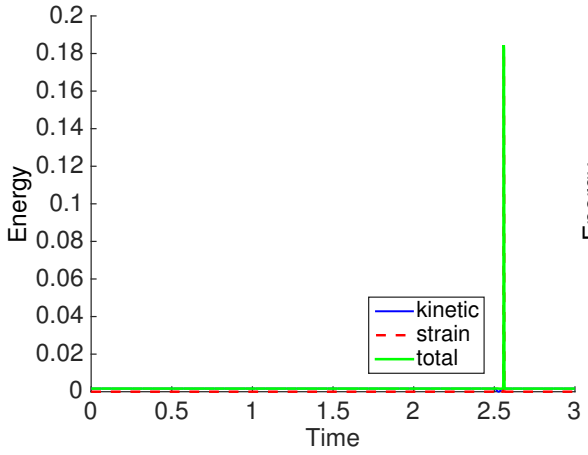


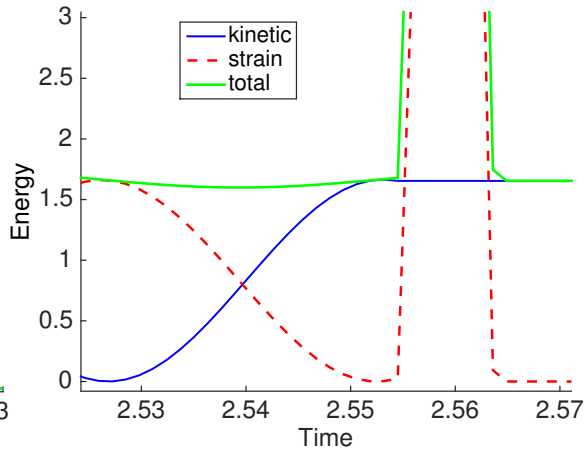
Figure 12: Cell crossing issue in MPM. Since the shape functions are linear, the derivatives of the shape functions which are used to form the divergence are piecewise constant over the elements. More important is that the derivatives are discontinuous across the cell boundary (change sign).



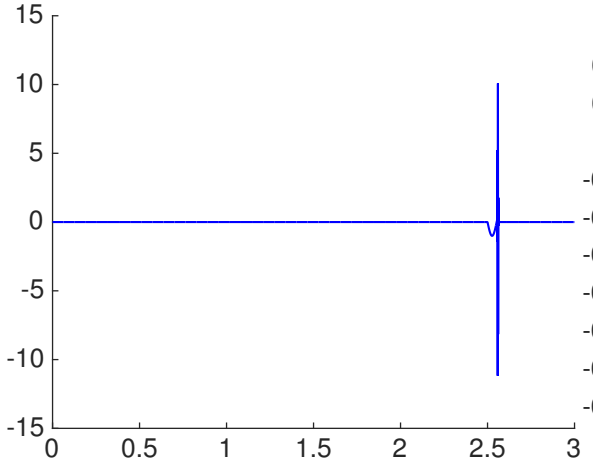
(a)



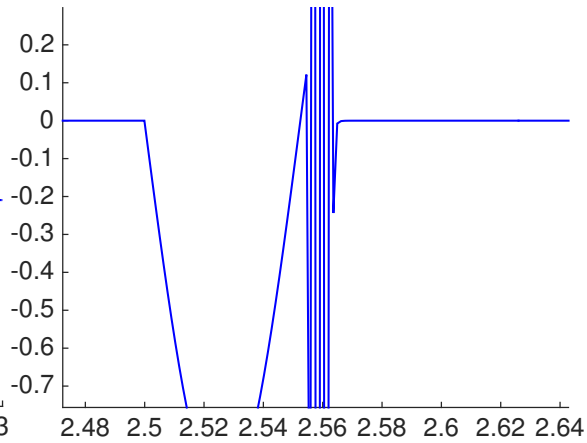
(b) energies ($\times 10^3$)



(c) energies, zoom-in



(d) stress

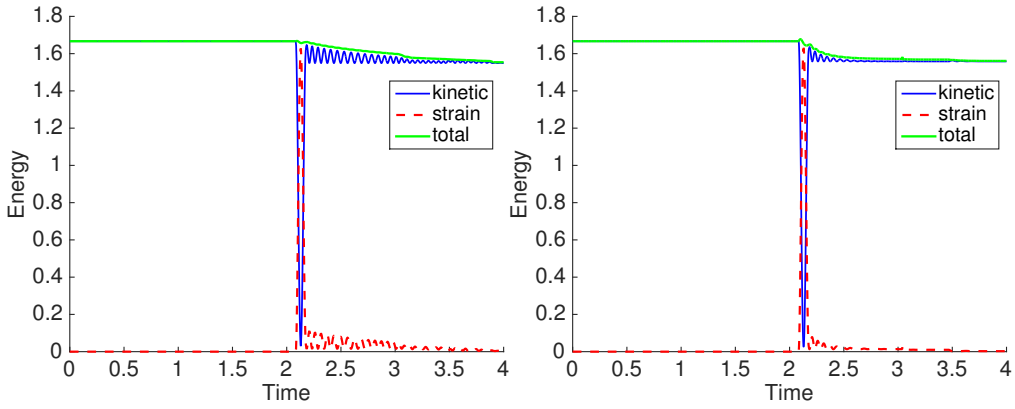


(e) stress, zoom-in

Figure 13: Simple example to demonstrate the cell-crossing issue of MPM: (a) problem setup, (b) plot of strain and kinetic energies showing the erroneous strain energy and (c) zoom-in plot. The stress of the left particle is depicted in (c,d).

uses Gauss points not MPs, whose data are interpolated from the MPs, to integrate the internal force vector (which is the cause of the problem) for fully filled elements. In Alonso and Zabala [2011] a simple procedure that can be used to reduce

this type of instability is to consider a constant stress at each cell equal to the stress average of the particles that are currently within the cell. In this case, the internal forces are obtained in the same way as in the FEM when one point of integration is used, using the gradient of the shape functions evaluated at the cell center. This idea is used later in the improved MPM of Sulsky and Gong [2016], see Section 5.5.



(a) energies ($\times 10^3$), $\Delta t = 0.0005$ (b) energies ($\times 10^3$), $\Delta t = 0.0001$

Figure 14: Using more particles per cell can reduce the impact of the cell-crossing issue.

3.3 Generalized interpolation material point (GIMP)

There are different ways to develop GIMP and Bardenhagen and Kober [2004] derived the formulation using the Petrov-Galerkin method. We adopt a simpler view taken by Steffen et al. [2008c]–the projection from particles to nodes for a scalar field g can be written as

$$g_I = \sum_p g_p N_I(\mathbf{x}_p) = \sum_p g_p \frac{\int_\Omega N_I(\mathbf{x})\delta(\mathbf{x}-\mathbf{x}_p)\mathrm{d}\Omega}{\int_\Omega \delta(\mathbf{x}-\mathbf{x}_p)\mathrm{d}\Omega} \tag{3.4}$$

where δ is the Dirac delta and N_I are the standard grid functions, presented in Section 3.2. In GIMP, one replaces the Dirac delta with a general function $\chi(\mathbf{x})$ called the *particle characteristic function* and the resulting projection is given by

$$g_I = \sum_p g_p \phi_I(\mathbf{x}_p) \tag{3.5}$$

where $\phi_I(\mathbf{x}_p)$ is given by

$$\phi_{Ip} \equiv \phi_I(\mathbf{x}_p) = \frac{1}{V_p}\int_{\Omega_p} \chi(\mathbf{x}-\mathbf{x}_p) N_I(\mathbf{x})\mathrm{d}\Omega \tag{3.6}$$

and the short notation ϕ_{Ip} was introduced to represent $\phi_I(\mathbf{x}_p)$; Ω_p denotes the particle domain. Fig. 15 illustrates the concept of particle domains in the MPM and in meshfree approximations. GIMP weighting functions, as defined by the above equation, are the convolution of the characteristic function and the grid basis function normalized by the particle volume.

The particle characteristic functions must satisfy the partition of unity property in the reference undeformed configuration [Bardenhagen and Kober, 2004]

$$\sum_p \chi_p(\mathbf{x}, t=0) = 1 \quad \forall \mathbf{x} \tag{3.7}$$

Note that Bardenhagen and Kober [2004] placed no such constraint on the characteristic functions in the deformed configuration due to the potential existence of gaps between the different particles' domains. However, the CPDI of Sadeghirad et al. [2011] ensures the PU of $\chi_p(\mathbf{x}, t)$ in the deformed configuration as it closely tracks the particle domains.

Typically piece-wise constant particle characteristic functions that satisfy the PU given in Equation (3.7) are used

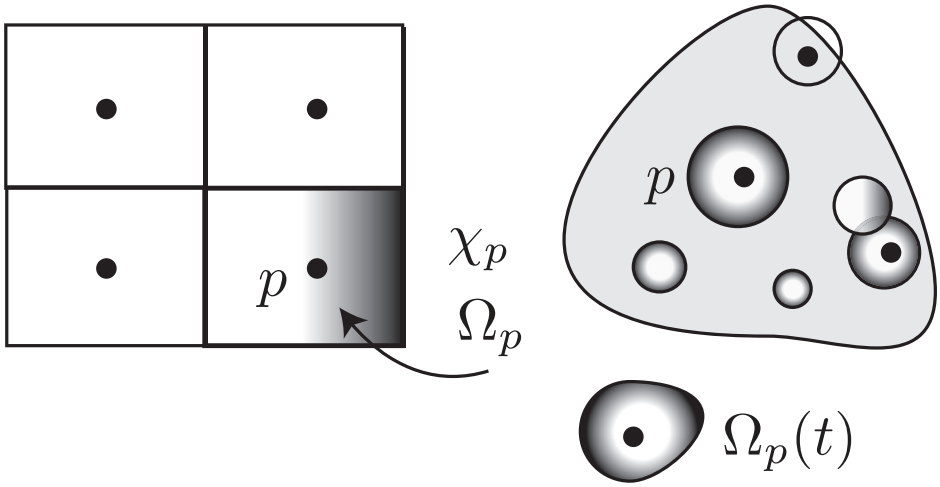


Figure 15: Particle domain and particle characteristic function in GIMP: rectangular particle domains can fill the initial material domain without overlapping (left) and overlapping circular particle domains commonly used in meshfree methods (right). Note also that particle domain is evolving in time as the material deforms.

$$\chi_p(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \in \Omega_p \\ 0 & \text{otherwise} \end{cases} \tag{3.8}$$

which implies that the material domain is partitioned into non-overlapping rectangular particle domains (in 2D) as shown in Fig. 15. This characteristic function is known as the "top-hat" function. This particular particle characteristic function results in the following GIMP weighting functions (simplification of Equation (3.6))

$$\phi_{Ip} = \frac{1}{V_p} \int_{\Omega_p} N_I(\mathbf{x}) \mathrm{d}\Omega \tag{3.9}$$

Since the GIMP functions depend on the particle domain Ω_p which in turn evolves in time, the GIMP functions are particle specific and time-dependent. Differences in how the integral in Equation (3.9) is evaluated and how the particle domains are defined/updated result in various GIMP formulations. Fig. 16 illustrates existing GIMP methods to be discussed in what follows.

In the approach commonly referred to as uGIMP (unchanged GIMP) Ω_p is kept fixed and the integral in Equation (3.9) can be exactly integrated resulting in analytical expressions for ϕ_{Ip}. Therefore, uGIMP is more effective than GIMP formulation. However, as material deforms, the unstretched particle domains cannot fill the material space. A more complicated approach, known as cpGIMP (contiguous particle GIMP), updates the particle domain using the deformation gradient $\mathbf{F}$ without taking shear deformation into account. Analytical expression for the weighting functions and its derivatives are available. Nonetheless, the updated particle domain is a axis-aligned rectangle in 2D and space cannot be tiled particularly for shearing. Andersen [2009] used Gaussian quadrature to numerically evaluate the GIMP basis functions on the fully updated particle domain (the particle domain is a parallelogram in 2D). But, it is very computationally expensive and thus should not be employed. Convected Particle Domain Interpolation (CPDI) [Sadeghirad et al., 2011, 2013] is the next logical development of GIMP where particles are given parallelogram-shaped domains that are constantly updated using the deformation gradient evaluated at the particle location. The novelty in CPDI is that the integrals in Equation (3.9) are also analytically evaluated thanks to the use of alternative basis functions. CPDI will be presented in Section 3.6.

3.3.1 uGIMP

In uGIMP and cpGIMP, the following one dimensional particle characteristic function is employed

$$\chi_p(x) = \begin{cases} 1 & \text{if } |x - x_p| \leq l_p/2 \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

Here l_p is the current particle size. The initial particle size is determined by dividing the cell spacing h_x by the number of particles per cell. Equation (3.9) is reduced to

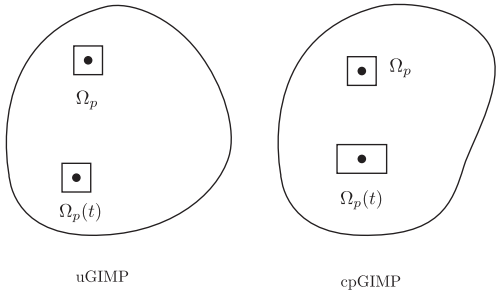


Figure 16: Tracking particle domain in GIMP: space cannot be tiled in a general multi-dimension domain using rectilinear Ω_p.

$$\phi_{Ip} = \frac{1}{l_p}\int_{x_p-l_p/2}^{x_p+l_p/2} N_I(x)\mathrm{d}x \tag{3.11}$$

and after substitution of the standard FE hat function $N_I(x)$, cf. Equation (3.1), into the above, one obtains the uGIMP function [Steffen et al., 2008c]

$$\phi(x) = \begin{cases} 1-(4x^2+l_p^2)/(4h_x l_p) & \text{if } |x| < 0.5l_p \\ 1-|x|/h & \text{if } 0.5l_p \le |x| \le h_x - 0.5l_p \\ (h+l_p/2-|x|)^2/(2h_x l_p) & \text{if } h_x - 0.5l_p \le |x| < h_x + 0.5l_p \\ 0 & \text{otherwise} \end{cases} \tag{3.12}$$

of which an example is given in Fig. 17[8]. Note that $\phi_I = \phi(x - x_I)$. Obviously, the GIMP functions are C^1 across the cell boundaries, have support in adjacent cells and their next nearest neighbors. Note also that there are four non-zero basis functions within one cell. But, for a given particle in a cell there are only three non-zero functions. As can be seen, if there are many particles per element (l_p is getting smaller), the GIMP functions resemble the standard FE hat functions.

The first derivative of the GIMP weighting function is given by

$$\phi_{,x} = \begin{cases} -8x/(4h_x l_p) & \text{if } |x| < 0.5l_p \\ -(1/h_x)\mathrm{sign}(x) & \text{if } 0.5l_p \le |x| \le h_x - 0.5l_p \\ -\mathrm{sign}(x)(h_x+l_p/2-|x|)/(h_x l_p) & \text{if } h_x - 0.5l_p \le |x| < h_x + 0.5l_p \\ 0 & \text{otherwise} \end{cases} \tag{3.13}$$

where $\mathrm{sign}(x)$ is the *signum* function.

In 2D the particle domain is a rectangle defined as $l_p^x \times l_p^y$ and the GIMP functions are the tensor product of the 1D functions i.e., $\phi(x,y) = \phi(x,l_p^x)\phi(y,l_p^y)$. An example of 2D uGIMP functions is given in Fig. 18. In 3D, the particle domain is a cube and the weighting function is defined similarly. For a given particle p there are 9/27 non-zero basis functions ϕ_{Ip} for 2D and 3D problems, respectively.

3.3.2 cpGIMP

In the cpGIMP, the particle domain is tracked by updating the particle sizes in the x and y directions

$$\begin{aligned} l_p^x(t+\Delta t) &= l_p^x(t=0)F_{xx}(t+\Delta t) \\ l_p^y(t+\Delta t) &= l_p^y(t=0)F_{yy}(t+\Delta t) \end{aligned} \tag{3.14}$$

i.e., the deformed domain is stretched in orthogonal directions but is never sheared. In other words, particle domains that start as squares (in 2D) or cubes (in 3D) will deform to rectangles (in 2D) or orthogonal boxes (in 3D), respectively. Accordingly, the cpGIMP is limited to problems for which deformation is along the grid directions so that off-diagonal deformation gradient components are negligible.

[8]Matlab scripts used to generate this plot are presented in Appendix E.1

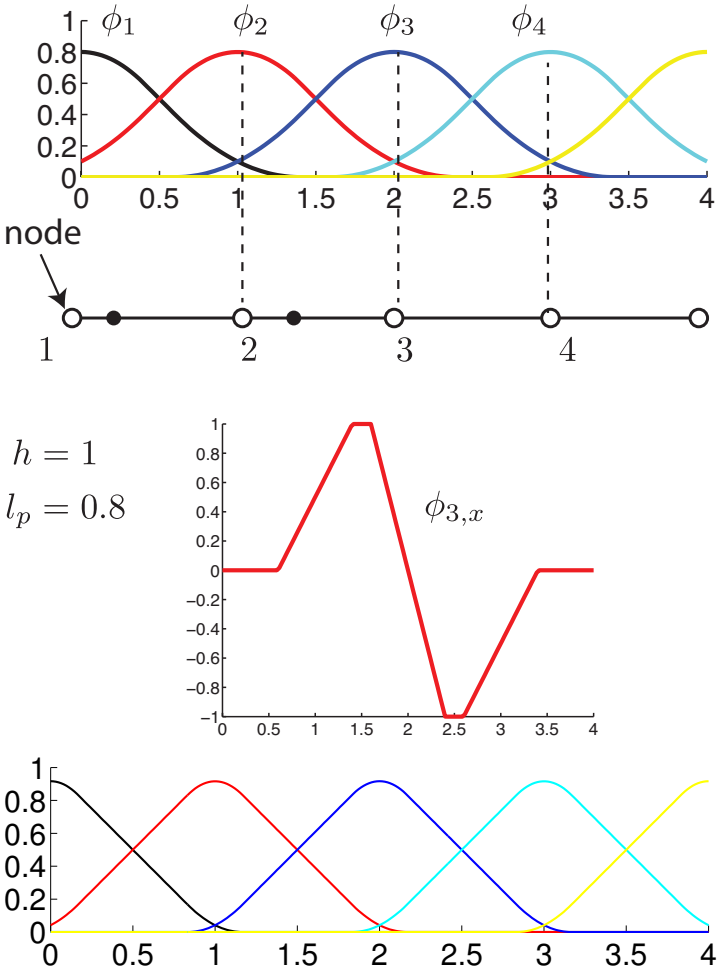


Figure 17: One dimensional GIMP basis functions: basis functions (top row), first derivative (middle row) and GIMP functions are reduced to MPM basis functions as l_p decreases (bottom row).

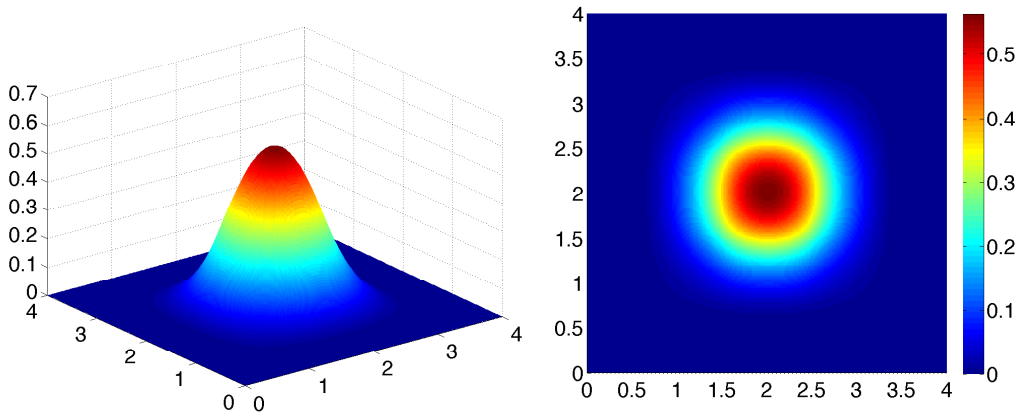


Figure 18: Two dimensional GIMP basis functions: illustrated for node at $(2, 2)$ on a square domain 4×4 discretized by 16 elements of spacing 1.

3.4 B-splines basis functions

Steffen et al. [2008c,b] showed that for simple problems, the use of cubic splines improves the spatial convergence properties of the MPM as grid-crossing errors are reduced. Cubic B-splines were also adopted in the amazing snow animations given in Stomakhin et al. [2013] for the Frozen movie. There are different ways to construct the B-splines basis functions, namely through a recurrence or a convolution concept. The latter was used in Steffen et al. [2008c,b], Stomakhin et al. [2013] and other MPM references. However, we present herein only the B-splines constructed using a recursive formula not only because we are familiar with them based on our work on isogeometric analysis but also because they are general.

Given a knot vector $\Xi^1 = \{\xi_1, \xi_2, \ldots, \xi_{n+k+1}\}$, which is defined as an ordered set of increasing parameter values, the associated set of B-spline basis functions $\{N_{i,k}\}_{i=1}^{n}$ are defined recursively by the Cox-de-Boor formula [Piegl and Tiller, 1996], starting with the zeroth order basis function ($k = 0$)

$$N_{i,0}(\xi) = \begin{cases} 1 & \text{if } \xi_i \leq \xi < \xi_{i+1}, \\ 0 & \text{otherwise,} \end{cases} \tag{3.15}$$

and for a polynomial order $k \geq 1$

$$N_{i,k}(\xi) = \frac{\xi - \xi_i}{\xi_{i+k} - \xi_i} N_{i,k-1}(\xi) + \frac{\xi_{i+k+1} - \xi}{\xi_{i+k+1} - \xi_{i+1}} N_{i+1,k-1}(\xi). \tag{3.16}$$

in which fractions of the form $0/0$ are defined as zero.

High order B-spline basis functions are C^{k-1} not C^0 as high order Lagrange polynomial basis, the connectivity of elements is, therefore, different from standard finite elements. Elements are defined as non-zero knot spans. Note that the B-splines functions are not interpolatory except at the boundaries when open knots[9] are used. Open knots facilitate the imposition of Dirichlet boundary conditions. B-splines elements are used extensively in Isogeometric Analysis [Hughes et al., 2005] which is a computational paradigm that reduces the gap between Computer Aided Design (CAD) and Finite Element Analysis (FEA).

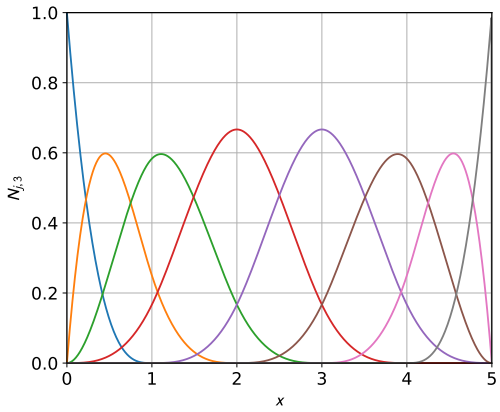


Figure 19: One dimensional cubic ($k = 3$) B-spline basis functions on an open uniform knot $\Xi = \{0, 0, 0, 0, 1, 2, 3, 4, 5, 5, 5, 5\}$. There are 8 nodes (control points in CAD terminology) and 5 elements (or knot spans in CAD). At can be seen from the figure, at any point there are 4 ($= k + 1$) non-zero basis functions. Therefore each element has 4 nodes. The first element's connectivity is $[1, 2, 3, 4]$ *i.e.* particles locate in this element contribute to nodes 1,2,3 and 4. The second element's connectivity is $[2, 3, 4, 5]$ and so on [Hughes et al., 2005].

Fig. 19 shows cubic B-spline basis functions for a uniform knot vector $\Xi = \{0, 0, 0, 0, 1, 2, 3, 4, 5, 5, 5, 5\}$. The knot vector is made of 5 knot spans and there are 8 basis functions. In order to use the knot spans as elements in the manner of Cartesian grid commonly used in MPM, one needs, in this example, 6 shape functions not 8. Therefore, there are two more basis functions than the number of shape functions required. The right number of shape function is obtained by combining the two basis functions (on each side) that do not peak at the junction between two elements to obtain the new one dimensional shape functions, now denoted by $S_{I,\xi}$ where ξ corresponds to any axis x, y or z, plotted on Fig. 20. By doing this, the partition of unity is respected and all elements have the same size. Similar to Steffen et al. [2008b], the two dimensional and

[9] Open knots are those where the first and last knots are repeated $p + 1$ times.

three-dimensional shape functions are obtained as the product of the different one-dimensional shape functions as:

$$\begin{cases} \phi_I(\mathbf{x}_p) & = S_{I,x}\left(\dfrac{x_p - x_I}{h_x}\right) \times S_{I,y}\left(\dfrac{y_p - y_I}{h_y}\right) \text{ in 2D} \\ \phi_I(\mathbf{x}_p) & = S_{I,x}\left(\dfrac{x_p - x_I}{h_x}\right) \times S_{I,y}\left(\dfrac{y_p - y_I}{h_y}\right) \times S_{I,z}\left(\dfrac{z_p - z_I}{h_z}\right) \text{ in 3D} \end{cases} \tag{3.17}$$

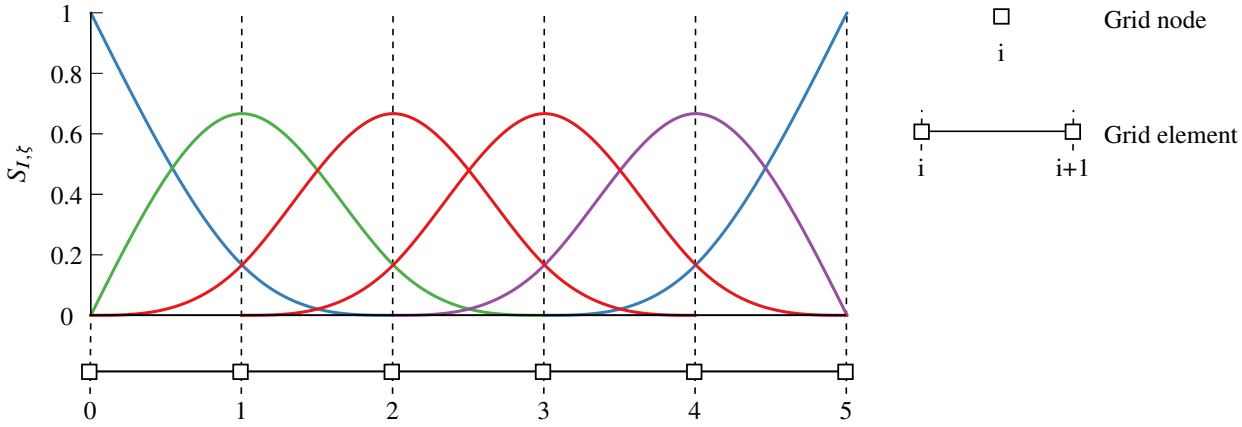


Figure 20: Cubic B-spline shape functions for a series of five elements in 1D. Note that there are now just 6 basis functions.

Because of the presence of boundaries, there are four different types of shape functions $S_{i,\xi}$ which differ by the position of node i with respect to the boundaries. They are represented on Fig. 20 by different colours and their expressions are:

- Shape functions of **type 1** (blue in Fig. 20): the particle i is located at the boundary, *i.e.* $\zeta_i = \zeta_B$, and have the following form:

$$S^1_{I,\xi}(r) = \begin{cases} \frac{1}{6}r^3 + r^2 + 2r + \frac{4}{3}, & -2 \le r \le -1 \\ -\frac{1}{6}r^3 + r + 1, & -1 \le r \le 0 \\ \frac{1}{6}r^3 - r + 1, & 0 \le r \le 1 \\ -\frac{1}{6}r^3 + r^2 - 2r + \frac{4}{3}, & 1 \le r \le 2 \end{cases}, \tag{3.18}$$

  where $r = (\zeta_p - \zeta_i)/h$.

- Shape functions of **type 2**: the particle i is located on the right side of the closest boundary one cell away from it, *i.e.* $\zeta_i = \zeta_B + h$, and have the following form:

$$S^2_{I,\xi}(r) = \begin{cases} -\frac{1}{3}r^3 - r^2 + \frac{2}{3}, & -1 \le r \le 0 \\ \frac{1}{2}r^3 - r^2 + \frac{2}{3}, & 0 \le r \le 1 \\ -\frac{1}{6}r^3 + r^2 - 2r + \frac{4}{3}, & 1 \le r \le 2 \end{cases} \tag{3.19}$$

- Shape functions of **type 3**: the particle i is located at least two cells away from any boundary, *i.e.* $\zeta_i \ge \zeta_B + 2h$, and have the following form:

$$S^3_{I,\xi}(r) = \begin{cases} \frac{1}{6}r^3 + r^2 + 2r + \frac{4}{3}, & -2 \le r \le -1 \\ -\frac{1}{2}r^3 - r^2 + \frac{2}{3}, & -1 \le r \le 0 \\ \frac{1}{2}r^3 - r^2 + \frac{2}{3}, & 0 \le r \le 1 \\ -\frac{1}{6}r^3 + r^2 - 2r + \frac{4}{3}, & 1 \le r \le 2 \end{cases} \tag{3.20}$$

- Shape functions of **type 4**: the particle i is located on the left side of the closest boundary, one cell away from it, *i.e.* $\zeta_i = \zeta_B - h$, and have the following form:

$$S^4_{I,\xi}(r) = \begin{cases} \frac{1}{6}r^3 + r^2 + 2r + \frac{4}{3}, & -2 \le r \le -1 \\ -\frac{1}{2}r^3 - r^2 + \frac{2}{3}, & -1 \le r \le 0 \\ \frac{1}{3}r^3 - r^2 + \frac{2}{3}, & 0 \le r \le 1 \end{cases} \tag{3.21}$$

These four types of one-dimensional shape functions $S_{i,\xi}$ translate into $4^2 = 16$ two dimensional shape functions, and into $4^3 = 64$ three dimensional shape functions: each node I having a different type along the respective axes x, y, and z. Unless otherwise stated, in the following, when cubic B-splines are used, each background cell is populated by 2, 4 and 8 material points in 1D, 2D and 3D, respectively. These particles will be located at positions defined by $\xi_1 = 0.2113$ and $\xi_2 = 0.7887$. And these locations are determined from the Gauss quadrature rule.

Remark 26 *Note that we have redefined the B-splines functions to have exactly $n+1$ nodes (and basis functions) for a mesh of n cells (1D). This is just a matter of implementation as the B-splines grid is now exactly the same as the grid that uses hat functions. Other implementation of B-splines in MPM usually follow the isogeometric analysis one, see e.g. Gan et al. [2018].*

3.5 Bernstein functions

Bernstein polynomials form a basis for the Bézier elements used in isogeometric analysis [Hughes et al., 2005]. These polynomials are used in CAD to construct the so-called Bézier curves/surfaces [Piegl and Tiller, 1996]. The univariate Bernstein basis functions of order k are defined over the biunit interval $[0, 1]$ as:

$$B_{i,k}(\xi) = \binom{k}{i} \xi^i \, (1-\xi)^{k-i} \tag{3.22}$$

where the binomial coefficient $\binom{k}{i} = \frac{k!}{i!(k-i)!}$ for $1 \leq i \leq k+1$. Bernstein polynomials of degree 2 are plotted in Fig. 21. These polynomials form a partition of unity: $\sum_{i=1}^{k} B_{i,k}(\xi) = 1$.

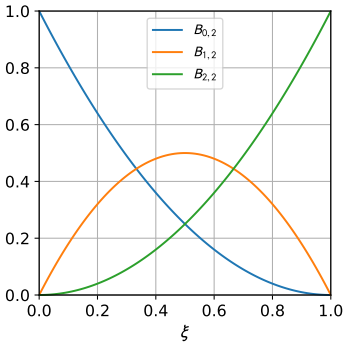


Figure 21: Bernstein basis polynomials of degree 2.

Each cell of the background mesh is made of 3 (in 1D), 9 (in 2D), or 27 (in 3D) nodes. As some of these nodes are common to different cells, we express the shape functions as a function of the normalized distance between a particle p and a node I, *i.e.* $(\mathbf{x}_I - \mathbf{x}_p)/h$:

$$\phi_I(\mathbf{x}_p) = S_{I,x}\left(\frac{x_I - x_p}{h_x}\right) \times S_{I,y}\left(\frac{y_I - y_p}{h_y}\right) \times S_{I,z}\left(\frac{z_I - z_p}{h_z}\right) \tag{3.23}$$

where $S_{I,\xi}$ are the shape functions along the axis ζ (*i.e.* x, y or z). The shape function depends on the position of the nodes in a cell: if it is located on an edge or the cell center along the axis i, they take two different forms: if the node I is located on an edge of a mesh element along the axis ζ:

$$S_{i,\xi}(r) = B_{0,2}(|r|) = \begin{cases} (1-|r|)^2 & \text{if } -1 \leq r \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.24}$$

otherwise, *i.e.* if the node I is located at the center (or inside) an element along the axis ζ:

$$S_{i,\xi}(r) = B_{1,2}\left(|r| + \frac{1}{2}\right) = \begin{cases} \frac{1}{2} - 2r^2 & \text{if } -1/2 \leq r \leq 1/2 \\ 0 & \text{otherwise} \end{cases} \tag{3.25}$$

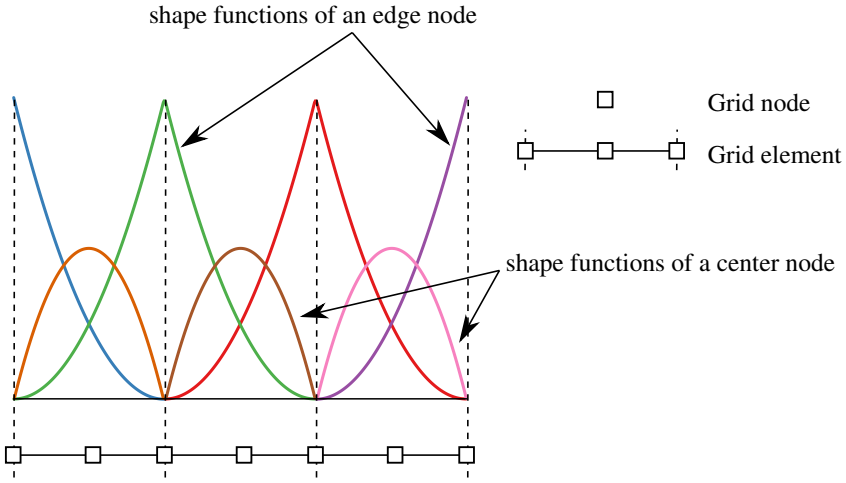


Figure 22: Bernstein quadratic shape functions for a series of three elements in 1D. Note that Bernstein functions are still C^0.

Fig. 22 shows these functions over a grid of three cells. As Bernstein functions are smooth but still C^0 across the cell boundaries, they are not recommended for adoption in any MPM variant except the TLMPM.

In this work, unless otherwise stated, when using Bernstein shape functions, each background cell is populated by 3, 9 and 27 material points in 1D, 2D and 3D, respectively. The particles will be located at positions defined by $\xi_{1,2 \text{ and } 3} = 0.1127$, 0.5, and 0.8873.

3.6 Convected Particle Domain Interpolation

The first method that can fully track particle domains is the Convected Particle Domain Interpolation (CPDI), developed in Sadeghirad et al. [2011] where 2D particle domains are approximated as parallelograms which still induce some gaps. Later on, quadrilateral particle domain was presented in Sadeghirad et al. [2013]. Nguyen et al. [2017] extended CPDI to triangular particle domains, tetrahedral domains, and also to arbitrary polygon/polyhedral domains. All these formulations are presented in this section. We refer to Appendix C for an interpretation of CPDI as a way of projecting quantities defined over a Lagrangian mesh to a Cartesian grid.

3.6.1 One dimensional linear CPDI (CPDI-L2)

In order to have a better understanding of the CPDI shape functions, we herein derive the one dimensional CPDI shape functions. Visualization of 1D CPDI functions were provided in the original reference [Sadeghirad et al., 2011] but without details. If the 1D particle domain is represented as a two-node line element, we have

$$\begin{aligned} \phi_I(x_p) &= 0.5N_I(x_p^1) + 0.5N_I(x_p^2) \\ d\phi_I(x_p) &= \frac{-1}{l_p}N_I(x_p^1) + \frac{1}{l_p}N_I(x_p^2) \end{aligned} \tag{3.26}$$

where x_p^1, x_p^2 are the corners of the particle domain i.e., the nodes of the line element.

For a visualization of these functions, we consider a grid of three cells with four nodes as shown in Fig. 23. There is one particle with $l_p = 1 = h_x$ that moves from the left (node 1) to the right (node 4). By using Equation (3.26), one can compute the CPDI basis functions of all four nodes as plotted in Fig. 23 (bottom figure). Also depicted is the standard FE shape functions–the well known hat functions (middle figure). As can be seen from the figure, the CPDI basis functions form a partition of unity, but they are not interpolator i.e., they do not satisfy the Kronecker property. It should be noted that as we did not use extra cells the CPDI functions do not form a PU for intervals $0 \leq x \leq 0.5$ and $2.5 \leq x \leq 3$.

3.6.2 Convected Particle Domain Interpolation (CPDI-R4)

In the first version of the CPDI family, the particle domain is tracked using the particle deformation gradient $\mathbf{F}$ in the way that the deformed particle domain is a parallelogram as shown in Fig. 24. The domain is defined by (i) the particle position and (ii) the domain vectors. The latter at time $t + \Delta t$ are given by

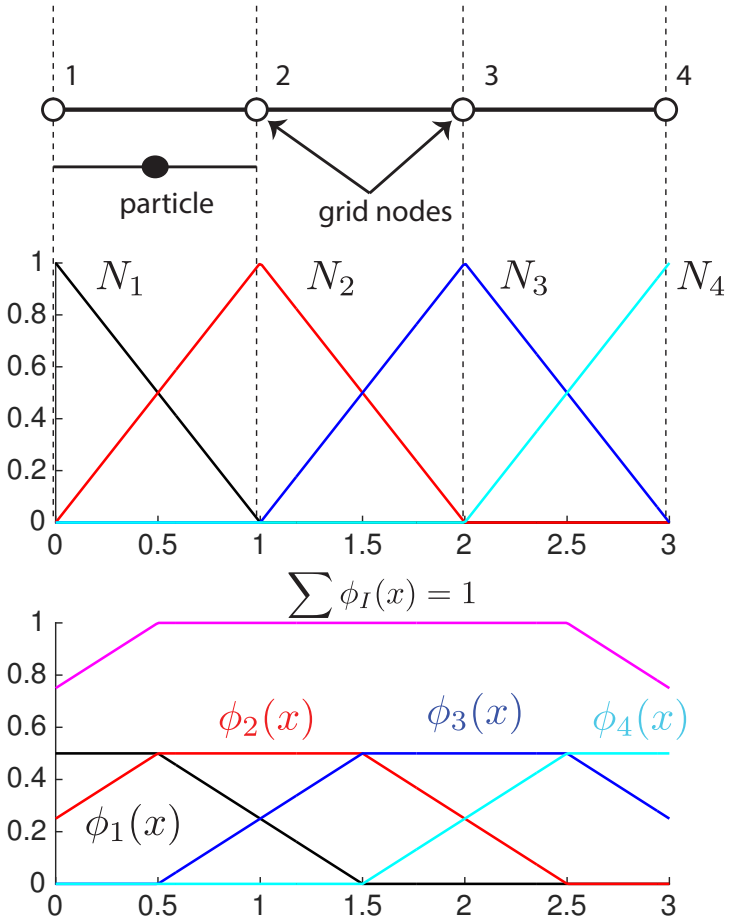


Figure 23: One dimensional CPDI-L2 shape functions (bottom figure). Also shown are the standard FE basis functions (middle figure) and the grid/particle (top figure).

$$\begin{aligned} \mathbf{r}_1^{t+\Delta t} &= \mathbf{F}^{t+\Delta t}\mathbf{r}_1^0 \\ \mathbf{r}_2^{t+\Delta t} &= \mathbf{F}^{t+\Delta t}\mathbf{r}_2^0 \end{aligned} \tag{3.27}$$

where $\mathbf{r}_1^0$ and $\mathbf{r}_2^0$ are the initial domain vectors. Since the initial particle domain is a rectangle (that is why we label it as CPDI-R4 where R stands for rectangles and 4 is the number of nodes of one particle domain), then we have

$$\mathbf{r}_1^0 = \begin{bmatrix} 0.5l_p^{x0} \\ 0 \end{bmatrix}, \quad \mathbf{r}_2^0 = \begin{bmatrix} 0 \\ 0.5l_p^{y0} \end{bmatrix} \tag{3.28}$$

where l_p^{x0} and l_p^{y0} are the initial particle sizes in the x and y direction, respectively.

The main issue in any GIMP methods is how to perform the integral in Equation (3.9) effectively. Particularly when one allows the deformed particle domain to be of arbitrary shape and thus located arbitrarily with respect to the background grid. In CPDI, this is achieved by approximating the grid hat functions $N_I(\mathbf{x})$ over the deformed particle domain Ω_p using yet another basis functions

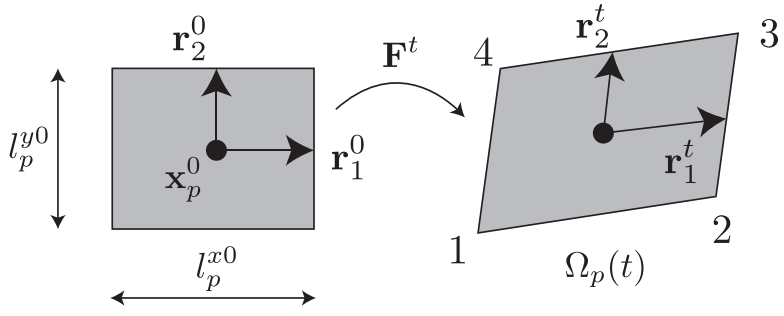


Figure 24: Particle domain as a parallelogram in CPDI is defined by the particle position and the domain vectors.

$$N_I(\mathbf{x}) \approx N_I^{\text{app}}(\mathbf{x}) = \sum_{c=1}^{4} M_c(\mathbf{x}) N_I(\mathbf{x}_c) \tag{3.29}$$

where $N_I(\mathbf{x}_c)$ are the conventional grid functions evaluated at the corner c of the particle domain and $M_c(\mathbf{x})$, or precisely $M_c(\xi,\eta)$ where (ξ,η) are the so-called parent coordinates, are the basis functions of the four-node quadrilateral elements (Q4). The original (N_I) and alternative basis functions ($N_I^{\text{app}}(\mathbf{x})$) differ from each other in the interior of the particle domain. And yet, the alternative basis function identically equals the exact basis function at the particle corners and hence on the particle edges since $M_c(\mathbf{x})$ are interpolation functions. This property makes the CPDI evaluation of nodal internal forces exact in 1D [Sadeghirad et al., 2011].

The GIMP basis functions in Equation (3.9) now becomes

$$\begin{aligned}\phi_{Ip} &= \frac{1}{V_p}\int_{\Omega_p} N_I^{\text{app}}(\mathbf{x})\mathrm{d}\Omega = \frac{1}{V_p}\int_{\Omega_p}\left[\sum_{c=1}^{4} M_c(\mathbf{x}) N_I(\mathbf{x}_c)\right]\mathrm{d}\Omega \\ &= \frac{1}{V_p}\sum_{c=1}^{4}\left[\int_{\Omega_p} M_c(\mathbf{x})\mathrm{d}\Omega\right] N_I(\mathbf{x}_c)\end{aligned} \tag{3.30}$$

and similarly the gradient $\nabla\phi_{Ip}$ is written by

$$\begin{aligned}\nabla\phi_{Ip} &= \frac{1}{V_p}\int_{\Omega_p} \nabla N_I^{\text{app}}(\mathbf{x})\mathrm{d}\Omega = \frac{1}{V_p}\int_{\Omega_p}\left[\sum_{c=1}^{4} \nabla M_c(\mathbf{x}) N_I(\mathbf{x}_c)\right]\mathrm{d}\Omega \\ &= \frac{1}{V_p}\sum_{c=1}^{4}\left[\int_{\Omega_p} \nabla M_c(\mathbf{x})\mathrm{d}\Omega\right] N_I(\mathbf{x}_c)\end{aligned} \tag{3.31}$$

Integrals in Equations (3.30) and (3.31) can be computed exactly and the resulting CPDI basis functions and first derivatives are written as [Sadeghirad et al., 2011]

$$\begin{aligned}\phi_{Ip} &= \frac{1}{4}\sum_{c=1}^{4} N_I(\mathbf{x}_c) \equiv \sum_{c=1}^{4} w_c^f N_I(\mathbf{x}_c), \quad w_c^f = 1/4 \\ \nabla\phi_{Ip} &= \frac{1}{V_p}\left\{(N_I(\mathbf{x}_1) - N_I(\mathbf{x}_3))\begin{bmatrix} r_{1y} - r_{2y} \\ r_{2x} - r_{1x}\end{bmatrix} + (N_I(\mathbf{x}_2) - N_I(\mathbf{x}_4))\begin{bmatrix} r_{1y} + r_{2y} \\ -r_{1x} - r_{2x}\end{bmatrix}\right\} \\ &\equiv \sum_{c=1}^{4} \mathbf{w}_c^g N_I(\mathbf{x}_c)\end{aligned} \tag{3.32}$$

where (r_{1x}, r_{1y}) are the components of $\mathbf{r}_1$; w_c^f and $\mathbf{w}_c^g$ are the so-called function/gradient weights. As can be seen from Equation (3.32), the basis function of node I evaluated at particle p is the sum of the conventional grid functions evaluated at the four corners of the particle domain. The gradient is the weighted sum of the conventional grid functions evaluated at the four corners of the particle domain. Note that the coefficient $1/V_p$ in the gradient is different from the original formula of Sadeghirad et al. [2011] ($1/(2V_p)$) because we adopted different domain vectors.

It can be observed that the function weights sum to unity and the gradient weights sum to zero. These properties are the consequence of the PU of the FE shape functions M_c. For example, one can write

$$\sum_c M_c(\mathbf{x}) = 1 \rightarrow \int_{\Omega_p}\sum_c M_c \mathrm{d}\Omega = V_p \rightarrow \sum_c\left(\frac{1}{V_p}\int_{\Omega_p} M_c \mathrm{d}\Omega\right) = 1 \tag{3.33}$$

This observation is useful for verifying the derivation of CPDI functions. Based on this, it can be straightforwardly shown that the CPDI functions satisfy the partition of unity. We refer to Fig. 23 for a demonstration of this property.

The position of the four corners (they are numbered counter clock wise as shown in Fig. 24) are computed from the particle position and the particle domain vectors

$$\begin{aligned}\mathbf{x}_1 &= \mathbf{x}_p - \mathbf{r}_1 - \mathbf{r}_2\\ \mathbf{x}_2 &= \mathbf{x}_p + \mathbf{r}_1 - \mathbf{r}_2\\ \mathbf{x}_3 &= \mathbf{x}_p + \mathbf{r}_1 + \mathbf{r}_2\\ \mathbf{x}_4 &= \mathbf{x}_p - \mathbf{r}_1 + \mathbf{r}_2\end{aligned} \tag{3.34}$$

and the particle domain volume is

$$V_p = A_p = 4\,||\mathbf{r}_1 \times \mathbf{r}_2|| \tag{3.35}$$

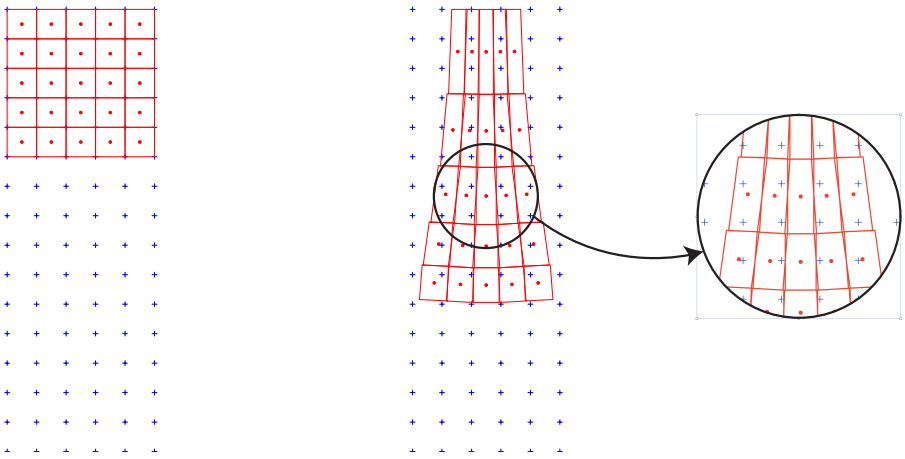


Figure 25: Gaps in CPDI-R4. This is a compliant bar under influence of a large gravity force.

3.6.3 Quadrilateral Convected Particle Domain Interpolation (CPDI-Q4)

It is obvious that parallelograms cannot fill space without gaps, cf. Fig. 25 and thus Sadeghirad et al. [2013] presented an improved CPDI method where particles are represented as quadrilaterals in 2D, cf. Fig. 26. This enhancement was referred to as CPDI2 by the authors. Herein, we label it the CPDI-Q4 to reflect that a particle resembles a Q4 finite element. This minor revision removes overlaps or gaps between particle domains and it also provides flexibility in choosing particle domain shape in the initial configuration.

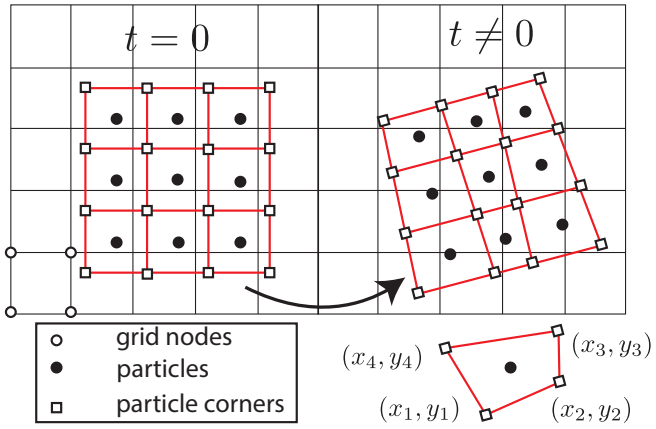


Figure 26: Particle domains as (bilinear) quadrilaterals in CPDI-Q4. Note that the particle domain corners only play a role in defining the basis functions and they do not carry any material quantities.

The CPDI-Q4 weighting function and its derivatives are written by [Sadeghirad et al., 2013]

$$\begin{aligned}\phi_{Ip} = \frac{1}{24V_p}\big[&(6V_p - a - b)N_I(\mathbf{x}_1) + (6V_p - a + b)N_I(\mathbf{x}_2)\\ &+ (6V_p + a + b)N_I(\mathbf{x}_3) + (6V_p + a - b)N_I(\mathbf{x}_4)\big]\end{aligned} \tag{3.36}$$

$$\nabla\phi_{Ip} = \frac{1}{2V_p}\left\{N_I(\mathbf{x}_1)\begin{bmatrix} y_2 - y_4 \\ x_4 - x_2 \end{bmatrix} + N_I(\mathbf{x}_2)\begin{bmatrix} y_3 - y_1 \\ x_1 - x_3 \end{bmatrix} \right.$$
$$\left. N_I(\mathbf{x}_3)\begin{bmatrix} y_4 - y_2 \\ x_2 - x_4 \end{bmatrix} + N_I(\mathbf{x}_4)\begin{bmatrix} y_1 - y_3 \\ x_3 - x_1 \end{bmatrix}\right\} \tag{3.37}$$

where $a = (x_4 - x_1)(y_2 - y_3) - (x_2 - x_3)(y_4 - y_1)$ and $b = (x_3 - x_4)(y_1 - y_2) - (x_1 - x_2)(y_3 - y_4)$. The volume (actually area) of the particle domain is given by $V_p = 0.5[(x_1y_2 - x_2y_1) + (x_2y_3 - x_3y_2) + (x_3y_4 - x_4y_3) + (x_4y_1 - x_1y_4)]$. There was a typo is Sadeghirad et al. [2013] and the above equations are correct. Nguyen et al. [2017] provided a derivation and Appendix E.2 discusses another derivation using a computer algebra system.

The particle corners are updated using the updated grid velocities (as if we did before for the particles)

$$\mathbf{x}_c^{t+\Delta t} = \mathbf{x}_c^t + \Delta t \sum_I N_I(\mathbf{x}_c)\mathbf{v}_I^{t+\Delta t} \tag{3.38}$$

By using the grid velocities to update the particle corners, no gaps between particle domains will be produced. If needed, e.g., for particle visualization, the particle positions can be computed as the centers of the particle domains:

$$\mathbf{x}_o \equiv \frac{1}{V_p}\int_{\Omega_p} \mathbf{x}\mathrm{d}\Omega = \sum_c w_f^c \mathbf{x}_c \tag{3.39}$$

where in the second equality the mapping $\mathbf{x} = M_c\mathbf{x}_c$ was used.

3.6.4 Triangular Convected Particle Domain Interpolation (CPDI-T3)

It is straightforward to extend the CPDI to the case where the particle domains are linear (or three-node) triangles as illustrated in Fig. 27. This is beneficial for geometries where a discretization in terms of triangles is available. The CPDI-T3 weighting functions and first derivatives are given by [Nguyen et al., 2017]

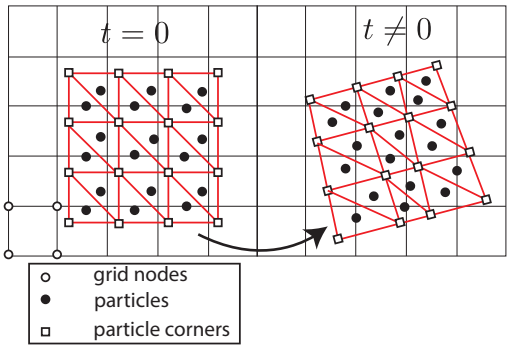


Figure 27: Particle domains as (linear) triangles in CPDI-T3.

$$\phi_{Ip} = \frac{1}{3}\left[N_I(\mathbf{x}_1) + N_I(\mathbf{x}_2) + N_I(\mathbf{x}_3)\right] \tag{3.40}$$

$$\nabla\phi_{Ip} = \frac{1}{2V_p}\left\{N_I(\mathbf{x}_1)\begin{bmatrix} y_2 - y_3 \\ x_3 - x_2 \end{bmatrix} + N_I(\mathbf{x}_2)\begin{bmatrix} y_3 - y_1 \\ x_1 - x_3 \end{bmatrix} + N_I(\mathbf{x}_3)\begin{bmatrix} y_1 - y_2 \\ x_2 - x_1 \end{bmatrix}\right\} \tag{3.41}$$

3.6.5 Three dimensional linear tetrahedron CPDI (CPDI-Tet4)

If the particles are represented by linear tetrahedron elements, the corresponding CPDI-Tet4 weighting and gradient weighting functions are given by [Nguyen et al., 2017]

$$\phi_{Ip} = \frac{1}{4}N_I(\mathbf{x}_1) + \frac{1}{4}N_I(\mathbf{x}_2) + \frac{1}{4}N_I(\mathbf{x}_3) + \frac{1}{4}N_I(\mathbf{x}_4)$$
$$\nabla\phi_{Ip} = \frac{1}{6V_p}\left\{N_I(\mathbf{x}_1)\begin{bmatrix} a_1 \\ b_1 \\ c_1 \end{bmatrix} + N_I(\mathbf{x}_2)\begin{bmatrix} a_2 \\ b_2 \\ c_2 \end{bmatrix} + N_I(\mathbf{x}_3)\begin{bmatrix} a_3 \\ b_3 \\ c_3 \end{bmatrix} + N_I(\mathbf{x}_4)\begin{bmatrix} a_4 \\ b_4 \\ c_4 \end{bmatrix}\right\} \tag{3.42}$$

where

$$
\begin{aligned}
a_1 &= y_{42}z_{32} - y_{32}z_{42}, a_2 = y_{31}z_{43} - y_{34}z_{13}, a_3 = y_{24}z_{14} - y_{14}z_{24}, a_4 = y_{13}z_{21} - y_{12}z_{31} \\
b_1 &= x_{32}z_{42} - x_{42}z_{32}, b_2 = x_{43}z_{31} - x_{13}z_{34}, b_3 = x_{14}z_{24} - x_{24}z_{14}, b_4 = x_{21}z_{13} - x_{31}z_{12} \\
c_1 &= x_{42}y_{32} - x_{32}y_{42}, c_2 = x_{31}y_{43} - x_{34}y_{13}, c_3 = x_{24}y_{14} - x_{14}y_{24}, c_4 = x_{13}y_{21} - x_{12}y_{31}
\end{aligned} \tag{3.43}
$$

with $x_{ij} = x_i - x_j$ and $y_{ij} = y_i - y_j$; $6V_p = x_{21}(y_{23}z_{34} - y_{34}z_{23}) + x_{32}(y_{34}z_{12} - y_{12}z_{34}) + x_{43}(y_{12}z_{23} - y_{23}z_{12})$. Note that V_p is a signed quantity and a proper node numbering was used to have a positive value. This CPDI-Tet4 has been used in Sinaie et al. [2018b] to model thin-walled metallic tubes under impacts and in Leavy et al. [2019] for mesoscale 3D simulations of polycrystalline materials.

3.6.6 Polygonal and polyhedral CPDI

Voronoi diagrams or Voronoi tessellations have widespread applications in computational geometry, city planning, computer graphics, geophysics, and meteorology etc. It is easy to make a simple Voronoi diagram. Just throw a random scattering of points (or seeds) across a plane, connect these sites with lines (linking each point to those which are closest to it), and then bisect each of these lines with a perpendicular, cf. Fig. 28. Each cell in the diagram encloses a particular site, and the surface of the cell contains all the points on the plane that are closer to that site than to any other. The properties of Voronoi diagrams have been studied extensively and we refer the readers to the review paper [Aurenhammer, 1991].

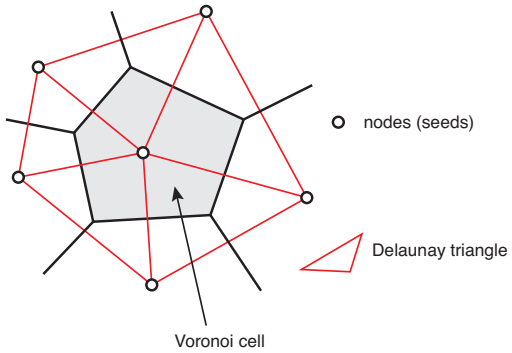


Figure 28: Voronoi diagrams and its dual–the Delaunay triangles.

A centroidal Voronoi tessellation (CVT) is a special type of Voronoi diagrams. A Voronoi tessellation is called centroidal when the generating seed of each Voronoi cell is also its mean–the center of mass with respect to a given density function [Du et al., 1999]. The center of mass is the arithmetic mean of all points weighted by the local density. If a physical object has uniform density, then its center of mass is the same as the centroid of its shape. It can be viewed as an optimal partition corresponding to an optimal distribution of generators.

Nguyen et al. [2017] extended CPDI to arbitrary polyhedron. For sake of simplicity, we present the 2D polygonal CPDI. The idea is simple: the particle polygon of n sides is partitioned into n triangles as shown in Fig. 29. This allows us to rewrite the function ϕ_{Ip} in Equation (3.9):

$$
\phi_{Ip} = \frac{1}{V_p}\sum_{s=1}^{n}\left[\int_{\Omega_p^s} N_I(\mathbf{x})\mathrm{d}\Omega\right] \approx \frac{1}{V_p}\sum_{s=1}^{n}\left[\int_{\Omega_p^s} N_I^{\text{app}}(\mathbf{x})\mathrm{d}\Omega\right] \approx \frac{1}{V_p}\sum_{s=1}^{n}\left[\sum_{c=1}^{3}\bar{w}_f^c N_I(\mathbf{x}_c)\right] \tag{3.44}
$$

where Ω_p^s denotes the sub-triangles and $\bar{w}_f^c$ are the function weights defined previously for the CPDI-T3 case: $\bar{w}_f^c = A_s/3$, $c = 1, 2, 3$, and A_s is the area of sub-triangle s.

Equation (3.44) can be rewritten in the following general form which is equally applicable to any n-sided polygons with $n \geq 4$

$$
\phi_{Ip} = \sum_{c=1}^{n+1} w_f^c N_I(\mathbf{x}_c) \tag{3.45}
$$

where for example $w_f^1 = (A_1/3 + A_2/3)/A$, and A denotes the area of the particle domain i.e., $A = \sum_s A_s$. Note that in our sub-sampling method in addition to the polygon vertices one needs to use the polygon's centroid as well.

In the same manner, the derivatives are given by

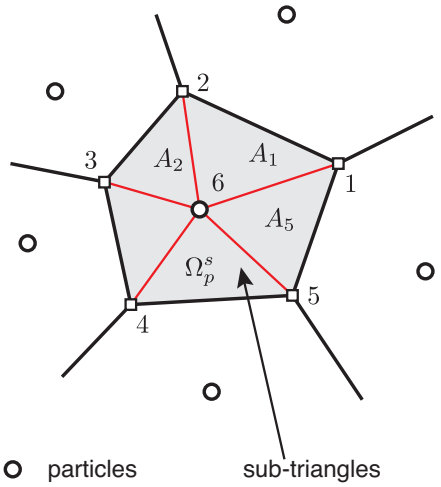


Figure 29: Particle domains as a polygon in polygonal CPDI.

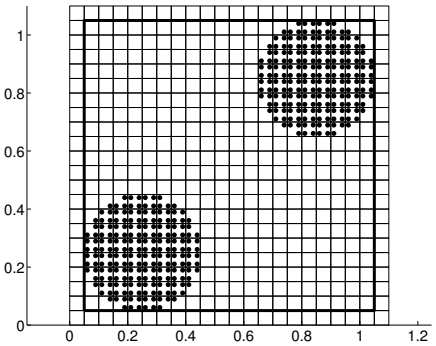


Figure 30: Ghost cells in GIMP/CPDI: due to the larger extent of the GIMP basis functions' support, ghost cells have to be employed. Note that material points never move to ghost cells.

$$\nabla\phi_{Ip} \approx \frac{1}{V_p}\sum_{s=1}^{n}\left[\int_{\Omega_p^s}\nabla N_I^{\text{app}}(\mathbf{x})\mathrm{d}\Omega\right] \approx \frac{1}{V_p}\sum_{s=1}^{n}\left[\sum_{c=1}^{3}\bar{\mathbf{w}}_g^c N_I(\mathbf{x}_c)\right] \approx \sum_{c=1}^{n+1}\mathbf{w}_g^c N_I(\mathbf{x}_c) \tag{3.46}$$

where $\bar{\mathbf{w}}_g^c$ are the unnormalized gradient weights of the sub-triangle under consideration.

The proposed sub-sampling method is easy to be implemented and it is applicable to polygons of arbitrary sides. By numbering the particle corner nodes in a counter clockwise order, one simply loops over the polygon edges, for each edge a sub-triangle is formed and the function weights of this sub-triangle are computed and accumulated to the corresponding weights. The particles are stored as a finite element mesh consisting of elements of different types: quadrilaterals, pentagons, hexagons and heptagons etc.

Remark 27 *The polyhedral CPDI could be the only MPM variant that represent the solid geometry most accurately (including the surfaces) and in the case that remeshing is needed (as the particle domains get distorted), no advection occurs. This is because the particles are the seeds for the Voronoi tesselation. If this is realized, the resulting method is quite similar to the PFEM. Yet this has not yet been implemented as remeshing is against the spirit of meshfree methods.*

3.7 Complications in GIMP/CPDIs

There are some complications associated with GIMP and CPDI. First, for axisymmetric problems, the weighting and gradients must be modified [Nairn and Guilkey, 2015]. Second, ghost cells have to be used and third, there are voids in CPDI. The latter two issues are discussed in this section.

High order C^1 GIMP shape functions and CPDIs requires the introduction of ghost cells (also referred to as extra cells) at the boundaries, cf. Fig. 30, so as that PU is satisfied. The use of ghost cells in GIMP is identical to extra cells in finite difference methods.

Next, we demonstrate holes can appear in the CPDIs[10]. The problem configuration is given in Fig. 31. The blue particle

[10]This was firstly discovered by Dr Brannon at University of Uintah presented in a post on her blog

is moving down with a constant velocity whereas the red particle initial velocity is null. Note that we purposely assign the particle sizes for these two particles as follows. The red particle has a particle size in the horizontal direction *exceeds three grid spans* while the size of the blue particle is slightly smaller than the grid cell. Snapshots of the simulations are shown in Fig. 32a and it clearly demonstrates that CPDIs functions have holes which allow the blue particle goes through the red particle without any deformation. When the particle size is smaller than 3 grid spans (for the red particle) then no hole is created as demonstrated in Fig. 32b. This issue can be solved using particle splitting [Homel et al., 2016].

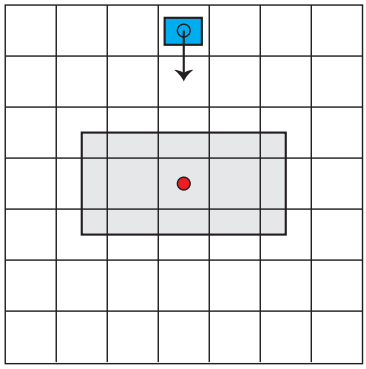
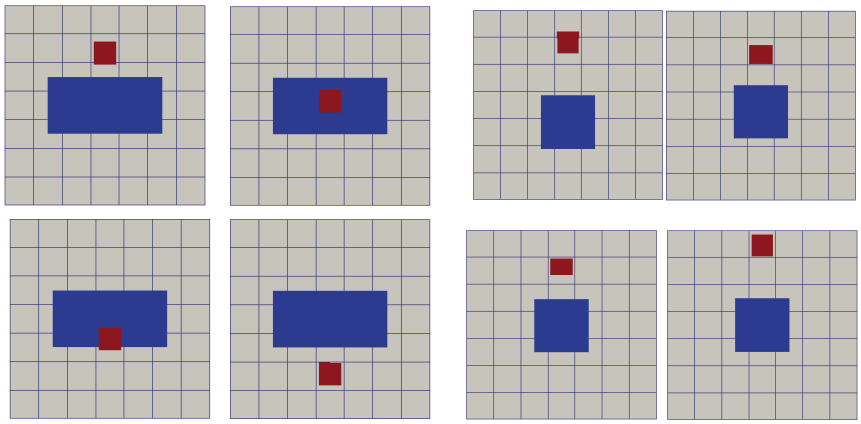


Figure 31: Hole problem configuration: a background grid of 7×7 cells with two particles of which one is moving down with a constant velocity.



(a) (b)

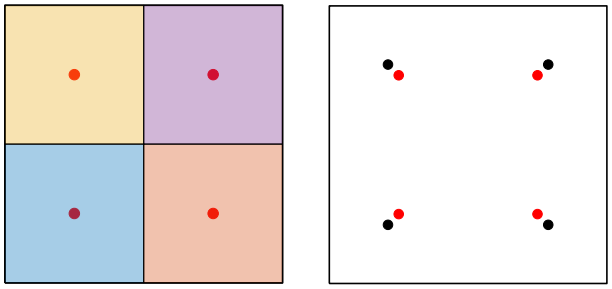
Figure 32: Hole problem result: particle configurations at different time steps.

4 Implementation aspects

The MPM algorithms presented in Section 2 as well as the various weighting functions treated in Section 3 put us in a position nearly ready for coding. There are, nonetheless, some implementation details need to be discussed. First, particle generation is discussed in Section 4.1. Second, application of initial and boundary conditions are given in Section 4.2. Third, as CPDI's implementation is slightly different from other MPM variants, we provide implementation details of CPDI in Section 4.3. Fourth, material point methods adopting an unstructured grid, very common in the geo-technical engineering field, are briefly considered in Section 4.4. Post-processing of the results of MPM simulations is discussed in Section 4.5. A short discussion on `Karamelo` is treated in Section 4.6.

4.1 Initial particle distribution

The MPM requires a grid and material points. As a Cartesian grid is easy to construct in any dimensions, we focus only on particle generation. In the FEM, the solid must be discretized into finite elements using a mesh generator. This meshing step is taking 80% of the total simulation time for complex geometries [Hughes et al., 2005]. In the MPM ,the solid is represented by a cloud of material points thus eliminating this time-consuming meshing step. There are numerous ways to obtain the initial particle distribution which depends on the geometry of the object and/or the available tools.



(a) Regular MPs (b) Gauss points

Figure 33: Initial positions of material points. The black dots are particles placed at Gauss points.

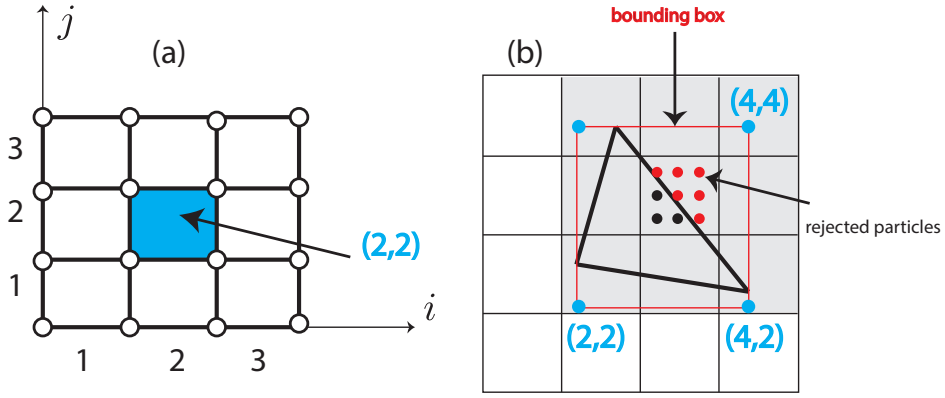


Figure 34: Grid cells are indexed by (i, j) in the integer index space (a), and this is used to quickly identity cells which are closest to a given geometry object based on the bounding box concept (b). Instead of distributing particles in all cells (16) one only does this for 9 cells.

4.1.1 Regular particle distribution

One can distribute the particles in a regular pattern as shown in Fig. 30. The idea is to use a constant number of material points per grid cell and discard the points that fall outside the boundary of the initial material domain. For simple geometries, checking whether a point is outside a domain or not can be done analytically. For complex geometries, the level set method [Sethian, 1999] can be used. This method of particle generation is the most commonly adopted one in MPM simulations. What constitutes a suitable number of particles is something of an open question, but it is typically advisable to use at least two particles in each cell in each direction, i.e. 4 particles per cell (PPC) in 2D and 8 PPC in 3D.

What should be the initial position of the particles relative to the grid cells? For the TLMPM, it is reasonable to place particles at the locations of the Gauss points that are optimal for numerical integration. For the ULMPM, there are different options. One can place the particles at Gauss points (this is coded in `Karamelo`), see Fig. 33b or regularly within the cells, see Fig. 33a. As the particles will, anyway, move out of their original locations, we do not see any difference between the two when a sufficient number of particles is used for each cell.

A fast particle generation algorithm. We present here some techniques to have a fast generation of particles for MPM simulations. For the sake of simplicity, only 2D is considered but the principles are general enough to be easily extended to 3D. If there are n grid cells and m geometrical objects, a naive algorithm by sweeping over the cells and for each cell loop over all the objects would result in an algorithm of order $\mathcal{O}(n \times m)$. This is inefficient if both n and m are large. First an integer cell coordinate i, j is introduced as shown in Fig. 34a. Note that the index is numbered from one. Let's assume that one needs to generate the particles for a triangle given in Fig. 34b. Based on the bounding box of the triangle and the cell indices, we can determine the cells surrounding the triangle. Finally, one loops only over those cells and distribute particles. A simple check whether a particle is within a polygon is used to discard particles outside the triangle.

The index of element with coordinates (i, j) is given by

$$e = i + \texttt{numx} \times (j - 1) \tag{4.1}$$

where `numx` denotes the number of cells along the x direction.

4.1.2 Irregular particle distribution

For solids of complex geometries, one can use a FE mesh generator to build a mesh and take the centers of the elements as particles. Actually, this is the technique often used with the uMPM (see Section 4.4). For example, in order to generate particles for a circular disk, one can use a mesh generator to partition the disk into a set of triangles. The particles can be taken as the centers of these triangles, Fig. 35. Alternatively, integration points of the triangular elements are mapped to the global coordinate system (using Equation (2.25)) and then are used as material points. The area of each triangle can be easily obtained and thus the particle volumes and masses can be determined. This kind of particle distribution is referred to as *irregular distribution*.

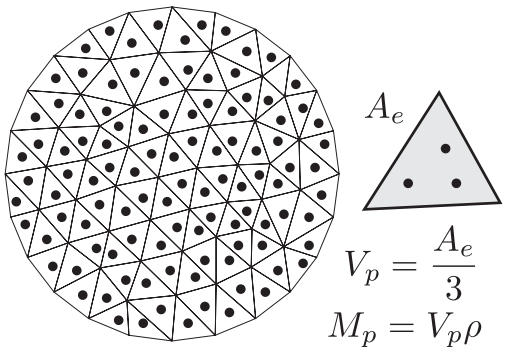


Figure 35: Irregular initial particle distribution: obtained using the available FE mesh generators.

Remark 28 *It would be even better if one can generate particles directly from the CAD files. The idea is to just using a CAD file and a background grid to build the particles. Nonetheless, to the best of our knowledge, such an algorithm has not yet been published.*

4.1.3 Particle distribution from images

Geometries are often available as digital images (e.g., CT scan of the microstructure of the materials). In this case, the MPM is more suitable than the FEM for it allows a rather straightforward pre-processing step from the images to the numerical spatial discretization [Bardenhagen et al., 2005, Guilkey et al., 2006, Nairn, 2007a]. The basic idea is to convert each pixel to a material point locating at the center of the pixel and depending on the intensity of the pixel the particle is tagged to a specific material. The quality of this process depends heavily on the contrast of the image. Fig. 36 depicts an example of converting an image which is a fiber-reinforced composite material to particles.

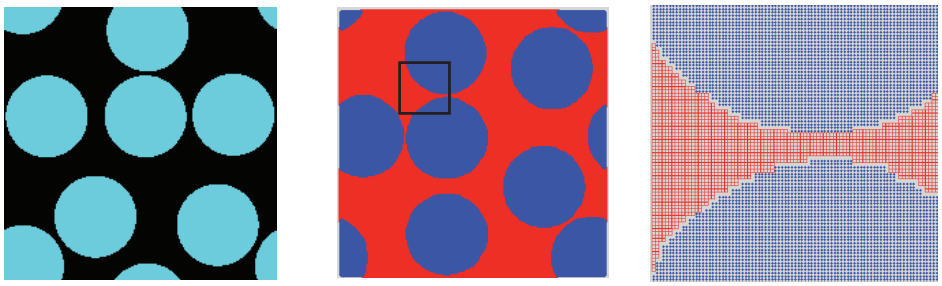


Figure 36: Initial particle distribution: converted from an image. From left to right: RGB image, particles and zoom in.

Remark 29 *It should not be misunderstood that MPM should be the method of choice when the solid geometry is defined as images. There exists excellent tools to convert images into finite element meshes such as* `Simpleware` *(commercially available at* `http://www.simpleware.com`*) or* `OOF` *(freely available at* `http://www.ctcms.nist.gov/oof/oof2/`*). Nonetheless, for highly complex geometries, the automation of the mesh generation process is notoriously difficult and a significant portion of analysis time is spent simply on mesh generation.*

4.2 Initial and boundary conditions

Initial conditions such as initial velocities, temperatures and stresses (residual stresses or equilibrium stresses obtained in a static analysis prior to a dynamic simulation) are imposed on the particles, prior to the time loop starts.

On the other hand, it is quite hard to enforce Dirichlet and Neumann boundary conditions (BCs) in the MPM. Confining the discussion to mechanical problems, there exist Dirichlet BCs of the type $v_{iI} = \bar{v}$ on Γ_u–the so-called Dirichlet boundary, and Neumann BCs of the type $\mathbf{t} := \boldsymbol{\sigma} \cdot \mathbf{n} = \bar{\mathbf{t}}$ on the Neumann boundary Γ_t. In the case that $\bar{\mathbf{t}} = \mathbf{0}$, that is all boundaries are traction-free, one does not have to do anything related to Neumann BCs. Note that if no Dirichlet BC is applied to the boundary nodes, then particles are able to freely move out of the computational domain.

Enforcement of Dirichlet BCs is presented in Section 4.2.1 and of Neumann BCs in in Section 4.2.2. A technique tailored to CPDIs for enforcing Neumann BCs is given in Section 4.2.3.

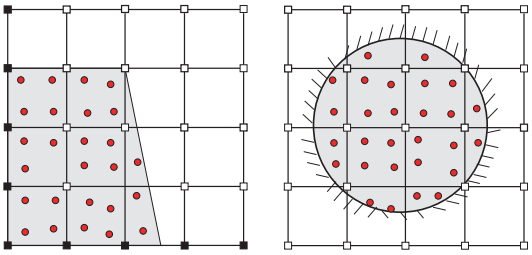


Figure 37: Dirichlet boundary condition treatment in MPM: boundary aligns with the grid (left) and boundary not aligned with the grid (right). Black solid squares are the nodes where the Dirichlet BCs are enforced.

4.2.1 Dirichlet boundary conditions

The Dirichlet boundary Γ_u is most of the case stationary. If Γ_u aligns with the grid, see Fig. 37a, it is straightforward to enforce BCs since the weighting functions satisfy the Kronecker delta property, at least at the boundaries (e.g., B-splines, GIMP). For explicit MPMs, basically one overwrites the calculated grid velocities (v_{iI}^t, $\tilde{v}_{iI}^{t+\Delta t}$ and $v_{iI}^{t+\Delta t}$) with the prescribed values ($\bar{v}$), for nodes I on Γ_u (solid black nodes in Fig. 37). For implicit MPMs, methods used in the FEM can be directly used [Hughes, 2000].

When the Dirichlet boundary is inclined, see Fig. 37b, different options exist. The easiest option is to adopt an unstructured grid that conforms to the Dirichlet boundary as done by geo-technical engineers, see e.g., Wieçkowski [2004]. If a Cartesian grid is being used, and for dynamics problems, the BCs can be enforced using rigid particles as discussed in Section 5.1.5. For implicit MPM (implicit dynamics and quasi-static), it is much harder and there are some solutions [Remmerswaal, 2017, Cortis et al., 2018, Bing et al., 2019, Liu and Sun, 2019]. Bing et al. [2019] presented a B-spline representation of 2D boundaries in the MPM.

Remark 30 *Note that there exists problems where Γ_u is in motion. For example, in the field of geo-technical engineering, a zero pore pressure condition is applied on a moving soil surface. For free surface flows, one also needs to apply a pressure boundary condition (i.e., fluid pressure equals air pressure). It is, therefore, necessary to locate accurately a moving boundary. Remmerswaal [2017] studied many techniques commonly used in fluid mechanics such as VOF (Volume of Fluid), SMM (Surface Marker Method) and LSM (Level Set Method).*

4.2.2 Neumann boundary conditions

Let us recall how the external force due to a traction is computed in the FEM. For simplicity, let's consider a 2D case. The Neumann boundary Γ_t is discretized by a set of 1D elements (they are actually the edges of the solid elements). The nodal force vector is then given by

$$\mathbf{f}_I^{\text{ext}} = \int_{\Gamma_t} N_I \bar{\mathbf{t}} \mathrm{d}\Gamma = \int_{-1}^{1} N_I(\xi) \bar{\mathbf{t}} J \,\mathrm{d}\xi \tag{4.2}$$

As an explicit representation of Γ_t is lacking, and there is no nodes on it as well, the MPM way of computing the force is as follows

$$\mathbf{f}_I^{\text{ext}} = \sum_{p=1}^{n_p} m_p \phi_I(\mathbf{x}_p) \bar{\mathbf{t}}^s(\mathbf{x}_p) h^{-1} = \sum_{p=1}^{n_p} A_p \phi_I(\mathbf{x}_p) \bar{\mathbf{t}}(\mathbf{x}_p) \tag{4.3}$$

where A_p represents the area of particle p. And the sum is over only boundary particles, see Fig. 38. The particle area can be updated using Nanson's formula, see e.g., Belytschko et al. [2000].

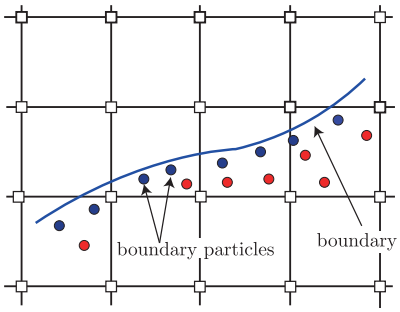


Figure 38: Computation of external force due to a surface traction in the MPM. The traction is applied to the so-called boundary particles.

4.2.3 Neumann boundary conditions with CPDI

In this section computation of surface tractions is presented using procedures extensively used in the FEM. We illustrate the procedure with an example of a cylinder subjected to an inner pressure in Fig. 39. For simplicity the discussion is confined to two dimensions.

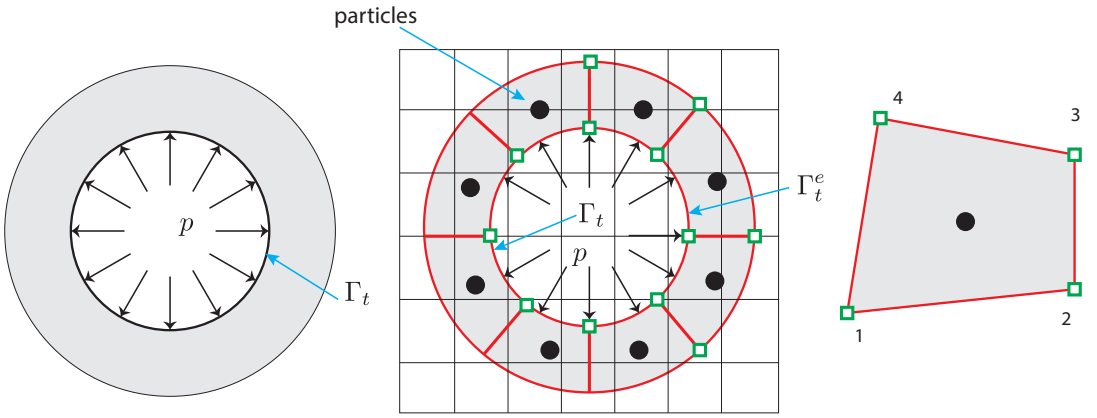


Figure 39: Computation of surface traction in CPDI-MPM. A cylinder subjects to inner pressure (left) and CPDI-MPM (right).

Let us consider the case in which a portion of the traction is applied on edge 1 of a particle domain – edge connecting nodes 1 and 2 – as shown in Fig. 39. The external force is thus given by

$$\begin{aligned} \mathbf{f}_I^{\text{ext}} &= \int_{\Gamma_t} N_I \bar{\mathbf{t}} \mathrm{d}\Gamma \\ &= \int_{\Gamma_t} \left(\sum_{c=1}^{4} M_c(\xi, -1) N_I(\mathbf{x}_c) \right) \bar{\mathbf{t}} \mathrm{d}\Gamma = \int_{-1}^{1} \left(\sum_{c=1}^{4} M_c(\xi, -1) N_I(\mathbf{x}_c) \right) \bar{\mathbf{t}} J \mathrm{d}\xi \end{aligned} \tag{4.4}$$

where in the second equality the idea of CPDI of approximating N_I was used; J is the Jacobian of the transformation that reads

$$J = ||\mathbf{x}'|| = \sqrt{x_{,\xi}^2 + y_{\eta}^2} = \frac{1}{2}\sqrt{x_{21}^2 + y_{21}^2} = 0.5l \tag{4.5}$$

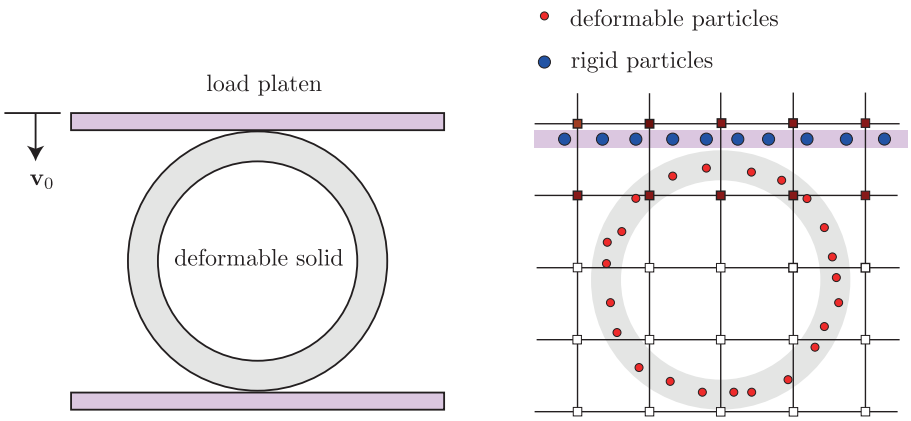


Figure 40: Moving displacement boundary condition: a ring being smashed by a rigid platen (a) and rigid body to model the platen (b). Dark red squares represent nodes which are assigned the moving velocities (if the hat functions are used).

which is constant and l is the length of edge 1. Furthermore, we assume that the traction is uniform over the particle edges. Thus Equation (4.4) becomes

$$\mathbf{f}_I^{\text{ext}} = \sum_{c=1}^{4} \left(\int_{-1}^{1} M_c(\xi, -1) \mathrm{d}\xi \right) N_I(\mathbf{x}_c) \bar{\mathbf{t}} J = N_1(\mathbf{x}_1) \bar{\mathbf{t}} J + N_2(\mathbf{x}_2) \bar{\mathbf{t}} J \tag{4.6}$$

4.2.4 Rigid bodies

Rigid bodies are used to model moving displacement boundary conditions as shown in Fig. 40. Rigid bodies are also represented by particles as deformable solids. However, there are some simplifications. Indeed, material properties are not required since these regions will not deform; their purpose is only to impose boundary conditions on deformable regions.

The positions and velocities of the rigid particles are simply given by (performed after updating the deformable particles)

$$\begin{aligned} \mathbf{v}_p^{\text{rigid},t+\Delta t} &= \mathbf{v}_0 \\ \mathbf{x}_p^{\text{rigid},t+\Delta t} &= \mathbf{x}_p^{\text{rigid},t} + \Delta t \mathbf{v}_0 \end{aligned} \tag{4.7}$$

The velocities of the rigid particles are transferred to the normal particles as follows. First, the elements which contain the rigid particles are determined. Second, all the nodes of these elements are marked (Fig. 40). These nodes are assigned the velocities of the rigid particles in case of a no slip contact between the rigid body and the deformable one. For frictional contacts, see Section 5.1.5. For implementation, we present the MUSL algorithm when rigid particles are present in Algorithm 4. It is worthy noting that this algorithm is simply a special case of multi-material contact algorithm of Bardenhagen et al. [2000].

Algorithm 4 Solution procedure of explicit MPM using MUSL with rigid bodies.

while $t < t_f$ **do**
  Reset grid quantities: $m_I^t = 0, (m\mathbf{v})_I^t = \mathbf{0}, \mathbf{f}_I^{\text{ext},t} = \mathbf{0}, \mathbf{f}_I^{\text{int},t} = \mathbf{0}$
  Mapping from deformable particles to nodes (P2G)
  end
  Update the momenta $(m\tilde{\mathbf{v}})_I^{t+\Delta t} = (m\mathbf{v})_I^t + \mathbf{f}_I^t \Delta t$
  Fix Dirichlet nodes I: I e.g., $(m\mathbf{v})_I^t = \mathbf{0}$ and $(m\tilde{\mathbf{v}})_I^{t+\Delta t} = \mathbf{0}$
  Fix Dirichlet nodes II: J e.g., $\mathbf{v}_J^t = \mathbf{v}_0$ and $\tilde{\mathbf{v}}_J^{t+\Delta t} = \mathbf{v}_0$
  Update deformable particle velocities and grid velocities (double mapping)
    Get nodal velocities $\tilde{\mathbf{v}}_I^{t+\Delta t} = (m\tilde{\mathbf{v}})_I^{t+\Delta t} / m_I^t$
    Update particle positions $\mathbf{x}_p^{t+\Delta t} = \mathbf{x}_p^t + \Delta t \sum_I \phi_I(\mathbf{x}_p^t) \tilde{\mathbf{v}}_I^{t+\Delta t}$
    Update particle velocities $\mathbf{v}_p^{t+\Delta t} = \alpha\left(\mathbf{v}_p^t + \sum_I \phi_I(\mathbf{x}_p^t)\left[\tilde{\mathbf{v}}_I^{t+\Delta t} - \mathbf{v}_I^t\right]\right) + (1-\alpha)\sum_I \phi_I(\mathbf{x}_p^t)\tilde{\mathbf{v}}_I^{t+\Delta t}$
    Update grid velocities $(m\mathbf{v}_I)^{t+\Delta t} = \sum_p \phi_I(\mathbf{x}_p^t)(m\mathbf{v})_p^{t+\Delta t}$
    Fix Dirichlet nodes I: $(m\mathbf{v})_I^{t+\Delta t} = \mathbf{0}$
    Fix Dirichlet nodes II: $\mathbf{v}_J^{t+\Delta t} = \mathbf{v}_0$
  end
  Update deformable particles (G2P)
  end
  Update rigid particles (G2P)
    $\mathbf{x}_p^{\text{rigid},t+\Delta t} = \mathbf{x}_p^{\text{rigid},t} + \Delta t \mathbf{v}_0$
  end
end while

4.3 Implementation of CPDI

We therein only present the implementation of only CPDI-Q4 as once you know how to implement CPDI-Q4, the implementation of the other variants is straightforward. The algorithm to compute the CPDI shape functions and derivatives for a given particle is given in Algorithm 5 and Fig. 41. One extra step, compared to all other MPM variants, is to update the particle domain vectors (for CPDI-R4) or particle corners (for CPDI-Q4). This is done at the end of every time step.

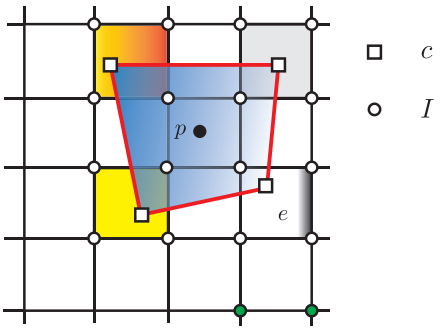


Figure 41: Particle and node interaction in CPDI-Q4: white solid circles are those nodes with non-zero shape functions at the particle i.e., $\phi_{Ip} \neq 0$, green solid circles are those do not interact with particle p. Corners of the particle domain are denoted by squares. Shaded elements are those contain the particle corners.

Algorithm 5 Algorithm to evaluate CPDI shape functions/derivatives.

1: Compute the four function weights w_c^f;
2: Compute the four gradient weights w_c^g;
3: For each corner, determine which element contains it;
4: Get the nodes of the four elements, I, that contain the four corners;
5: For each node of those, loop over the corner and compute $N_I(\mathbf{x}_c)$;
6: Use Equations (3.36) and (3.37) to compute ϕ_I and $\nabla\phi_I$.

4.4 MPM using an unstructured grid

Although very computationally intensive, MPM formulations adopting an unstructured background grid is popular in the geo-technical engineering community [Wieçkowski et al., 1999, Wieçkowski, 2004, Beuth et al., 2011, Jassim et al., 2013]. One advantage is the ease with which boundary conditions can be enforced. Note, however, that this is only the case for fixed boundaries. Another motivation of adopting unstructured grids is to handle complex geometries (e.g., multiple soil layers). Herein we discuss aspects that are specific to MPM using unstructured meshes. First, shape functions of the MPM using unstructured grids, dubbed uMPM, are discussed in Section 4.4.1. Second, the problem of particle registration to the grid is treated in Section 4.4.2. Finally, mixed integration, as a means to reduce cell-crossing error in the quasi-static uMPM, is given in Section 4.4.3. A more efficient uMPM with C^1 shape functions is given in Section 4.4.4.

4.4.1 Shape functions

Since it is obvious that developing GIMP (and CPDI) and B-splines for an unstructured mesh is difficult, uMPM simply employs the well-known isoparametric finite elements in which the shape functions are written in terms of the so-called natural coordinates. Thus, there is a need to convert the particle position (in global coordinates) to the natural coordinates. For illustration, consider a grid made of four-node quadrilateral (Q4) elements. By employing the isoparametric concept one writes

$$x = \sum_{I=1}^{4} N_I(\xi,\eta) x_I^e, \quad y = \sum_{I=1}^{4} N_I(\xi,\eta) y_I^e \tag{4.8}$$

where (x_I^e, y_I^e) denote the nodal coordinates of element e that contains the particle (x, y) under consideration; $N_I(\xi,\eta)$ are the Q4 shape functions.

One solves Equation (4.8) by using the iterative Newton-Raphson method:

$$\begin{bmatrix} \frac{\partial N_I}{\partial \xi}\Big|_{\xi_0} x_I^e & \frac{\partial N_I}{\partial \eta}\Big|_{\eta_0} x_I^e \\ \frac{\partial N_I}{\partial \xi}\Big|_{\xi_0} y_I^e & \frac{\partial N_I}{\partial \eta}\Big|_{\eta_0} y_I^e \end{bmatrix} \begin{bmatrix} \Delta\xi \\ \Delta\eta \end{bmatrix} = \begin{bmatrix} x - N_I(\xi_0,\eta_0) x_I^e \\ y - N_I(\xi_0,\eta_0) y_I^e \end{bmatrix} \tag{4.9}$$

which provides $\Delta\xi, \Delta\eta$ and the solution is updated $\xi = \xi_0 + \Delta\xi$, $\eta = \eta_0 + \Delta\eta$. The iteration continues until convergence is attained. The initial value for (ξ_0, η_0) are usually $(0, 0)$ i.e., the iterative procedure starts from the center of the element.

4.4.2 Particle registration

Particle registration is required to perform the particle-to-node and node-to-particle mapping. For example, to calculate the nodal mass $m_I^t = \sum_p \phi_I(\mathbf{x}_p^t) m_p$, one needs to know which particle p contributes to which nodes I. While this step is fast and efficient when a Cartesian grid is used, the registration of particles to the grid in uMPM is inefficient as the grid is unstructured. The problem is how to quickly locate an element that contains a given particle. Just a few works discussed this problem. In Wang et al. [2005] a ray-crossing algorithm is employed to determine whether the material points are inside or outside of arbitrary quadrilateral cells. An in-depth analysis of this problem was given in the master thesis [Pruijn, 2016].

4.4.3 Mixed integration

In the uMPM, the standard particle-based quadrature of the MPM is adopted for dynamics problems but each element is filled with many elements e.g., 10 material points per one single linear tetrahedral [Al-Kafaji, 2013]. For quasi-static problems, a mixed integration is adopted [Beuth et al., 2011]. In this mixed integration scheme, Gaussian integration is applied to all elements that are fully filled with material, whereas material point based integration is only adopted for partially filled elements. Elements (cells) in the interior of a body are assumed to be fully filled. Conversely, partially filled elements are assumed to occur only along the boundary of a body. Precisely, an element located on the boundary of the body is considered to be partially filled when the volume sum of all particles inside this element is less than a prescribed percentage of the element volume.

4.4.4 uMPM with C^1 shape functions

We think that the mixed integration is not an elegant solution to cell-crossing issue. Recently, de Koster et al. [2019] developed C^1 basis functions over unstructured grids using Powell-Sabin functions [Powell and Sabin, 1977, May et al., 2016]. While this constitutes a C^1 uMPM formulation which is free of cell-crossing problem similar to BSMPM (MPM with a Cartesian grid and B-splines as weighting functions), evaluation of the Powell-Sabin functions seems complex and to the best knowledge of the authors, no work on 3D extension has been reported.

To close this discussion on the uMPM, we anticipate that a total Lagrangian uMPM would be the most efficient uMPM as the shape functions and its derivatives and the particle registration need only be calculated once for all.

4.5 Visualization

In the FEM, visualization is typically performed on the mesh. Information stored at the integration points are extrapolated to the nodes to this end. Additionally some averaging are carried out to obtain smooth fields as each node is connected to a number of elements. In MPM, the computational grid is fixed and thus not suitable for visualization. There are at least three approaches to visualizing MPM results [Childs et al., 2012]

- Particle visualization: the particles are visualized directly as spheres/circles (3D/2D). Particle data (position, stresses etc.) are written to files and can be processed by `Paraview` or `VisIt` or `Ovito`[11].
- Particle mesh visualization: in standard MPM, one can generate a mesh from the particle positions using a Delaunay triangulation and use this mesh for visualization. In this way, available visualization technologies developed for the FEM can be reused. Note that this method is expensive since the particles positions are evolving in time. And then visualization can be done with `Paraview` or `VisIt`.
- Voxel-based visualization: each particle is assigned with a domain so that the particles are alike to pixels/voxels of digital images [Andersen and Andersen, 2010b, Choudhury et al., 2010]. This visualization technique is best suited for CPDI formulation.

Fig. 42 illustrates particle based and mesh-based visualization. For particle-based visualization, `Ovito` [Stukowski, 2009] is a very good tool as it is a scientific visualization and analysis software for atomistic and particle simulation data.

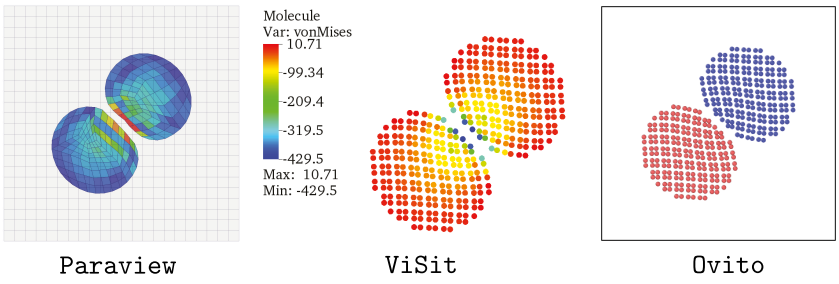


Figure 42: CPDI and MPM/GIMP visualization of MPM data.

Andersen and Andersen [2010a], Dunatunga and Kamrin [2015] showed that although the stress field at the individual material points may show a noisy variation, a smooth physically realistic stress field can be extracted by a mass-weighted mapping via the computational grid:

$$\begin{aligned} \boldsymbol{\sigma}_I &= \frac{1}{m_I}\left(\sum_p \phi_I(\mathbf{x}_p)\boldsymbol{\sigma}_p m_p\right) \\ \boldsymbol{\sigma}_p^{\text{smooth}} &= \sum_I \phi_I(\mathbf{x}_p)\boldsymbol{\sigma}_I \end{aligned} \tag{4.10}$$

We refer to Section 5.3.1 and Section 6.1.4 for illustrations of noisy particle results. It is interesting to note that Andersen and Andersen [2010a] realized that attempt to utilize these more realistic stresses $\boldsymbol{\sigma}_p^{\text{smooth}}$ directly in the computational MPM scheme has been unsuccessful. But it might be the motivation for Zhang et al. [2011] to have proposed the dual domain MPM.

[11] `http://www.paraview.org`, `https://wci.llnl.gov/codes/visit/`, `https://www.ovito.org`.

4.6 `Karamelo`, a 3D MPM code

Computational mechanics researchers constantly face the trade-off between rapid prototyping and good software engineering. The former allows researchers to code quickly (so as to try different ideas and rapid publication of the work) but inevitably leads to issues such as unsatisfactory performance, poor portability and maintainability. The latter results in reusable codes for future projects, but slows down the research progress due to low-level engineering. Open source MPM codes belong to the former category can be found for example in Sinaie et al. [2017] who presented a Julia implementation and at https://csmbrannon.net/2018/11/09/matlab-and-mms-source-files/ for a Matlab code. Open source MPM codes belong to the latter are presented in Section 1.4.2.

There exist a few works describing MPM implementations. For example, Parker [2002] developed a parallel MPM code using Message Passing Interface (MPI) with excellent scalability of more than 1000 processors as demonstrated in Parker et al. [2006] for simulations having about 16 million particles. Huang et al. [2008] described a parallel MPM using OpenMP for shared memory machines and Li and Sulsky [2000] presented a 3D parallel Fortran MPM code using MPI. Ma et al. [2010] described an object-oriented C++ implementation of MPM. Most recently Dong and Grabe [2018] presented a GPU implementation. Amongst all the open-source MPM codes, `Uintah` developed by Parker [2002] is probably the most efficient one to date. Unfortunately, this large code is difficult to understand and modify.

Therefore, there was a need to develop a portable, efficient, and easy to modify code that can be used in either 1D, 2D or 3D. In order to achieve this goal, a new code called `Karamelo` was developed in-house. The structure of this code is based on that of the popular molecular dynamics simulator LAMMPS [Plimpton, 1995]. LAMMPS is a highly parallel simulator that can be used on multi-CPU machines that support MPI, and also on GPUs (Graphic Processing Units). Moreover, through its ingenious hierarchical class system, it is easy to add new functionalities in LAMMPS. This is illustrated by the fact that although LAMMPS is a molecular dynamics code, modules for particle based methods such as SPH and peridynamics (see e.g., Silling [2000]) have been added to it [Ganzenmüller, 2014]. Because of the use of a background grid, it would be difficult to directly implement the MPM in LAMMPS, this is why it was decided to create a new code that uses LAMMPS's hierarchical class system, but adapted to the requirements of the MPM. `Karamelo` was born.

This section presents a short description of `Karamelo`. This code uses a number of libraries given in Section 4.6.1. Input file format and outputs are described in Section 4.6.2. MPI parallelization is treated in Section 4.6.3. Current functionalities are outlined in Section 4.6.4 and how to add new stuff is discussed in Section 4.6.5. For more details on this package, we refer to de Vaucorbeil and Nguyen [2020].

4.6.1 Dependencies

For all linear algebra operations (and we use many), the library `Eigen` (http://eigen.tuxfamily.org) is used. For plotting using the Python library `matplotlib`, the library `matplotlib-cpp` (https://github.com/lava/matplotlib-cpp) is used. It is a library to invoke `matplotlib` from C++. By snowball effect, the code is then also depending on Python 2.7. Finally, gzstream is used to compress the LAMMPS dump files.

4.6.2 Input file format and outputs

Interacting with `Karamelo` is performed through input files using an easy and flexible syntax. One can for example intuitively add new variables that can be constant:

```
E = 211
```

or depend on internal variables such as time – using the `time` variables – or particle positions – using the `x`, `y` or `z` variables:

```
T     = 1
r     = sqrt(x*x+y*y)
theta = atan2(y,x)
g     = sin(PI*time/T)
```

Everything else is controlled through functions. For instance, the global dimensionality of the simulation is set by the dimension(...) command:

```
dimension(2) # Sets the dimensionality to 2D
```

The evolution of the simulations (material points and grid nodes' position and properties) can be visualized either with `Ovito` or be plotted directly with `PyPlot` package [Johnson, 2012]. `Ovito` reads a series of LAMMPS dump files (dump_p.*.LAMMPS and dump_g.*.LAMMPS) created as follows:

```
dump(dump01, all, particle, N_dump, dump_p.*.LAMMPS)
dump(dump02, all, grid,     N_dump, dump_g.*.LAMMPS)
```

where `N_dump` is the step interval at which the output is written. Whereas

```
dump(dump03, all, pyplot, N_dump, plot.*.pdf, 640, 480)
```

plots every `N_dump` steps the grid nodes' and particles' positions as well as their domains (if using GIMP or CPDI) and saves it as a pdf (plot.*.pdf).

4.6.3 Parallelization using MPI

Similarly to LAMMPS, `Karamelo` supports multi-CPU compatitions through the use of MPI. As of today, though it does not support GPUs yet. In `Karamelo`, the total domain defined by the span of the background grid is equally split amongst the different CPU used (see Fig. 43). The connection between the different sub-domains is performed by the means of ghost nodes. The use of ghost nodes was preferred to that of ghost particles since Ruggirello and Schumacher [2014] observed that the later is superior over the former for scaling to large-scale problems. Therefore the nodes making of all the boundary mesh cells are shared amongst multiple CPUs (Fig. 43). First, the interaction between these nodes and the local particles (i.e., the particles present in the domain allocated to the current CPU) are first calculated. The results are then reduced over all the CPUs sharing the same nodes. In one time step, reduction is done once after calculating the nodes' mass, and every time their forces and velocities are computed. In the case of CPDI, however, as the domain of a given particle can overlap multiple CPUs, both ghost nodes and ghost particle's approach is used.

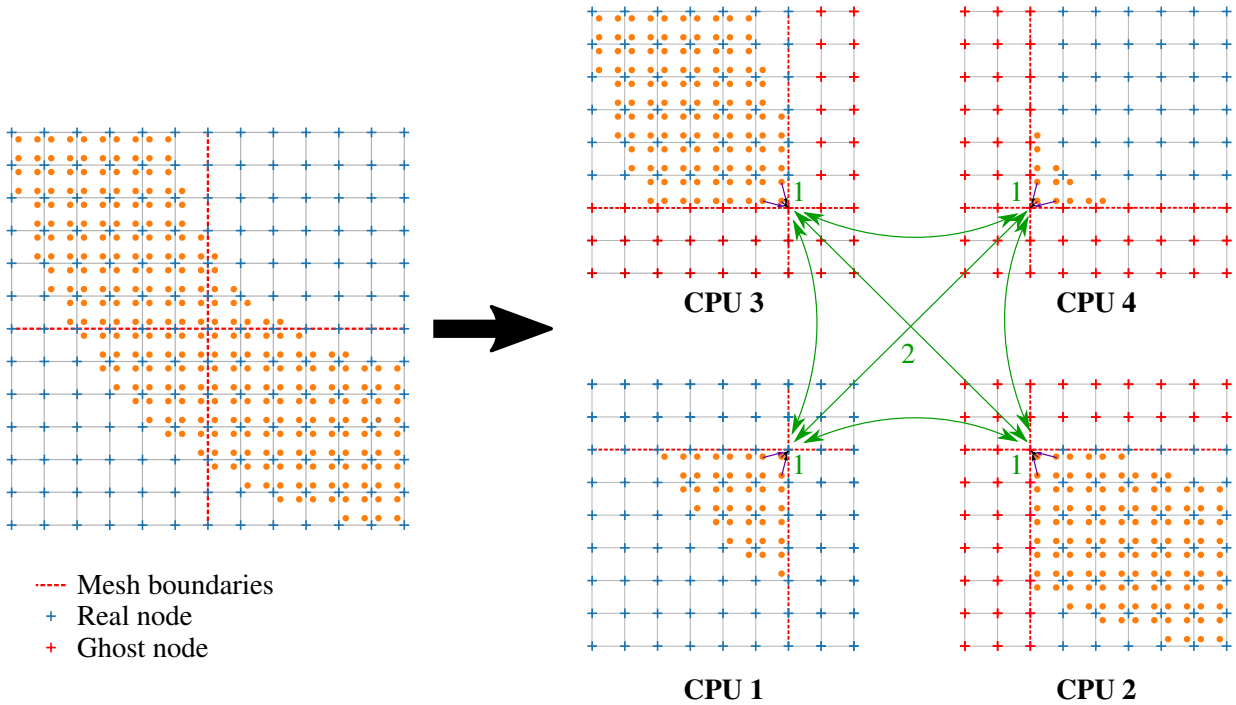


Figure 43: Illustration of the mapping from particles to node order for nodes that are shared amongst different CPUs. 1: mapping is done locally, 2: the mass, or forces and velocities are reduced. The CPU that on which resides the real node is the one that manages the reduction.

4.6.4 Functionalities

`Karamelo` is a young code (born on January 2019), but it already supports the TLMPM (with hat functions, cubic B-splines and quadratic Bernstein functions) and the ULMPM (hat functions, CPDI, cubic B-splines). Only explicit time integration is coded. The code can be used to model solids (elastic, hyperelastic and elasto-damage-plastic) and weakly compressible fluids/gases.

4.6.5 Extending `Karamelo`

As `Karamelo` uses the same pyramidal class system as LAMMPS, adding new functionalities is an easy task. Indeed, in order to do so, one only needs to add a new class – through the addition of a header (*.h) and a source file (*.cpp) – that can be easily derived from existing classes. For instance adding total Lagrangian CPDI can be done by copying `ulcpdi.cpp` and `ulcpdi.h` to `tlcpdi.cpp` and `tlcpdi.h`, respectively, changing the class name – i.e., from ULCPDI to TLCPDI – and changing a few line to only compute the shape functions in the first time step.

5 Advanced topics

Up to this point, a basic MPM formulation for solid mechanics and its implementation has been presented. While this is sufficient for simulating various interesting problems in solid mechanics, topics such as frictional contacts, fluids and gases, volumetric locking, adaptivity and improved MPM variants were not discussed. This section is devoted to such topics. Section 5.1 presents the most widely used contact algorithm of Bardenhagen et al. [2000]. Volumetric locking is discussed in Section 5.2. Section 5.3 provides a coupled thermo-mechanical MPM. How fluids and gases can be treated using the MPM is discussed in Section 5.4. Improved higher order MPM formulations are given in Section 5.5. And Section 5.6 is devoted to the topic of grid adaptive refinement and particle splitting/merging.

5.1 Contacts

This section presents the contact algorithm of Bardenhagen et al. [2000] for deformable bodies. The case of contact between deformable bodies and rigid bodies will be treated subsequently as simplification. This contact algorithm might be viewed as a *predictor-corrector* scheme, in which the (trial) nodal velocities are predicted from the solution of each body separately (as if no contact occurred) and then corrected using a contact model. The contact algorithm applies only for *contact nodes* which are defined as those who receive contribution from particles of more than one body, cf. Fig. 44. For simplicity, the discussion is confined to contacts between different bodies, self-contact requires a special treatment and is discussed in Homel and Herbold [2017].

For each body $k = 1, 2, \dots n$ where n denotes the number of bodies, one solves the standard MPM problem

$$\begin{aligned} m_I^{t,(k)} &= \sum_{p=1}^{n_p^{(k)}} \phi_{Ip} m_p, \quad \mathbf{v}_I^{t,(k)} = \frac{1}{m_I^{t,(k)}} \sum_{p=1}^{n_p^{(k)}} \phi_{Ip} m_p v_p \\ \mathbf{a}_I^{t,(k)} &= \frac{\mathbf{f}_I^{(k)}}{m_I^{t,(k)}} \\ \tilde{\mathbf{v}}_I^{t+\Delta t,(k)} &= \mathbf{v}_I^{t,(k)} + \Delta t \mathbf{a}_I^{t,(k)} \end{aligned} \tag{5.1}$$

where $n_p^{(k)}$ denotes the number of particles making up body k. Note that the tilded velocity field is not final and needs to be corrected for contact nodes. The corrected velocity $\mathbf{v}_I^{t+\Delta t,(k)}$ (without a tilde) will be used for updating particle's stress, position and velocity.

The next step after getting $\tilde{\mathbf{v}}_I^{t+\Delta t,(k)}$ is to detect, at contact nodes, whether two bodies are approaching or departing each other. In what follows the presentation is restricted to two bodies for sake of simplicity. The algorithm is general and can be applied to multiple bodies though. Bardenhagen et al. [2000] proposed an algorithm which is linear in the number of bodies[12]:

$$\left(\tilde{\mathbf{v}}_I^{t+\Delta t,(k)} - \mathbf{v}_I^{\text{cm}}\right) \cdot \mathbf{n}_I^{(k)} = \begin{cases} \geq 0 & \text{contact} \\ < 0 & \text{release} \end{cases} \tag{5.2}$$

where $\mathbf{v}_I^{\text{cm}}$ is the so-called *center of mass* velocity field which is given by

$$\mathbf{v}_I^{\text{cm}} = \frac{(m_I \tilde{\mathbf{v}}_I)^{t+\Delta t,(1)} + (m_I \tilde{\mathbf{v}}_I)^{t+\Delta t,(2)}}{m_I^{t,(1)} + m_I^{t,(2)}} \tag{5.3}$$

[12] Actually York, in his PhD dissertation York [1997], proposed to use center-of-mass velocities.

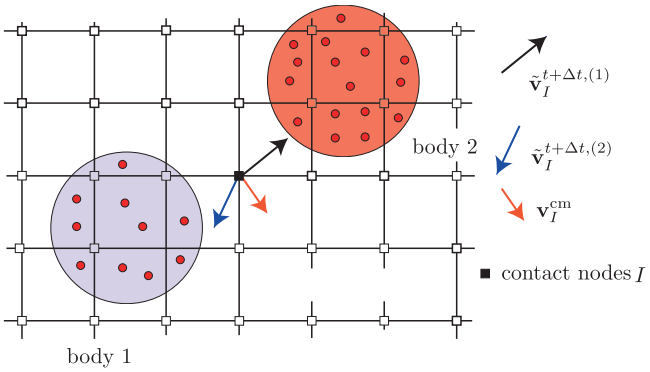


Figure 44: Two bodies come into contact. Contact or overlapped nodes (black solid square) are those who receive contribution from particles of both bodies.

which is the velocity obtained from the contribution of the particles of the two bodies. One can refer to this velocity as the system velocity field. The key to the algorithm is not to consider pairwise interactions of bodies, but rather use a common frame (global quantities) so that contact of all bodies can be achieved at once.

If Equation (5.2) determines that the two bodies are approaching together, then one needs to correct the velocities $\tilde{\mathbf{v}}_I^{t+\Delta t,(k)}$ to get the final one $\mathbf{v}_I^{t+\Delta t,(k)}$. Otherwise they are kept unchanged. How to correct the grid velocities depends on the contact model (to be discussed in Section 5.1.1 for the case of non-slip contact and in Section 5.1.2 for the case of Coulomb frictional contact). Finally, the particle velocity, position and stresses are updated using the following equations

$$\begin{aligned}
\mathbf{a}_I^{t+\Delta t,(k)} &= \frac{\mathbf{v}_I^{t+\Delta t,(k)} - \mathbf{v}_I^{t,(k)}}{\Delta t} \\
\mathbf{x}_p^{t+\Delta t,(k)} &= \mathbf{x}_p^{t,(k)} + \Delta t \sum_{p=1}^{n_p^{(k)}} \phi_I(\mathbf{x}_p^t)\mathbf{v}_I^{t+\Delta t,(k)} \\
\mathbf{v}_p^{t+\Delta t,(k)} &= \mathbf{v}_p^{t,(k)} + \Delta t \sum_{p=1}^{n_p^{(k)}} \phi_I(\mathbf{x}_p^t)\mathbf{a}_I^{t,(k)}
\end{aligned} \tag{5.4}$$

where the first equation is to compute the corrected accelerations (only needed for contact nodes). Note that stress update was skipped as it is standard. The above discussion corresponds to the USL formulation.

5.1.1 Contact without friction

The contact-release algorithm applied for a contact node I is quite simple. If contact is occurring, the nodal velocity is corrected so that the normal component of the body velocity is set equal to the normal component of the center-of-mass velocity. Otherwise, the two bodies move in their own velocities. Mathematically, one writes

$$\mathbf{v}_I^{t+\Delta t,(k)} = \begin{cases} \tilde{\mathbf{v}}_I^{t+\Delta t,(k)} - \left[\left(\tilde{\mathbf{v}}_I^{t+\Delta t,(k)} - \mathbf{v}_I^{\mathrm{cm}}\right)\cdot \mathbf{n}_I^{(k)}\right]\mathbf{n}_I^{(k)} & \text{contact} \\ \tilde{\mathbf{v}}_I^{t+\Delta t,(k)} & \text{release} \end{cases} \tag{5.5}$$

If there is no friction between the bodies, then the above adjustment of the normal component of the body velocity, Equation (5.5), is all that is required for contact treatment. The tangential component of the body velocity is unconstrained.

Remark 31 *The fact that the normal component of the body velocity is equal to the normal component of the center-of-mass velocity can be proved as:*

$$\begin{aligned}
\mathbf{v}_I^{t+\Delta t,(k)}\cdot\mathbf{n}_I^{(k)} &= \tilde{\mathbf{v}}_I^{t+\Delta t,(k)}\cdot\mathbf{n}_I^{(k)} - \left\{\left[\left(\tilde{\mathbf{v}}_I^{t+\Delta t,(k)} - \mathbf{v}_I^{cm}\right)\cdot\mathbf{n}_I^{(k)}\right]\mathbf{n}_I^{(k)}\right\}\cdot\mathbf{n}_I^{(k)} \\
&= \tilde{\mathbf{v}}_I^{t+\Delta t,(k)}\cdot\mathbf{n}_I^{(k)} - \left(\tilde{\mathbf{v}}_I^{t+\Delta t,(k)} - \mathbf{v}_I^{cm}\right)\cdot\mathbf{n}_I^{(k)} \\
&= \tilde{\mathbf{v}}_I^{t+\Delta t,(k)}\cdot\mathbf{n}_I^{(k)} - \tilde{\mathbf{v}}_I^{t+\Delta t,(k)}\cdot\mathbf{n}_I^{(k)} + \mathbf{v}_I^{cm}\cdot\mathbf{n}_I^{(k)} = \mathbf{v}_I^{cm}\cdot\mathbf{n}_I^{(k)}
\end{aligned}$$

The fact that the tangential component of the corrected entity velocity is the same as the tangential component before the correction can be proved as:

$$\mathbf{v}_I^{t+\Delta t,(k)} - [\mathbf{v}_I^{t+\Delta t,(k)} \cdot \mathbf{n}_I^{(k)}]\mathbf{n}_I^{(k)} = \tilde{\mathbf{v}}_I^{t+\Delta t,(k)} - \left[\left(\tilde{\mathbf{v}}_I^{t+\Delta t,(k)} - \mathbf{v}_I^{cm}\right) \cdot \mathbf{n}_I^{(k)}\right]\mathbf{n}_I^{(k)} - (\mathbf{v}_I^{cm} \cdot \mathbf{n}_I^{(k)})\mathbf{n}_I^{(k)}$$
$$= \tilde{\mathbf{v}}_I^{t+\Delta t,(k)} - (\tilde{\mathbf{v}}_I^{t+\Delta t,(k)} \cdot \mathbf{n}_I^{(k)})\mathbf{n}_I^{(k)}$$

5.1.2 Contact with Coulomb friction

In the case of frictional sliding, one needs to modify the tangential component of the grid velocity. To apply Coulomb friction, first calculate the force necessary to cause the bodies to stick completely. This force can be determined from the relative tangential velocity: $\left(\tilde{\mathbf{v}}_I^{t+\Delta t,(k)} - \mathbf{v}_I^{\text{cm}}\right) - \left[\left(\tilde{\mathbf{v}}_I^{t+\Delta t,(k)} - \mathbf{v}_I^{\text{cm}}\right) \cdot \mathbf{n}_I^{(k)}\right]\mathbf{n}_I^{(k)} = \mathbf{n}_I^{(k)} \times \left[\left(\tilde{\mathbf{v}}_I^{t+\Delta t,(k)} - \mathbf{v}_I^{\text{cm}}\right) \times \mathbf{n}_I^{(k)}\right]$. This allows us to compute the stick force and then the Coulomb friction force. From that, the corrected velocity field is given by [Bardenhagen et al., 2001, Coetzee, 2003]

$$\mathbf{v}_I^{t+\Delta t,(k)} = \tilde{\mathbf{v}}_I^{t+\Delta t,(k)} - \left[\Delta\mathbf{v} \cdot \mathbf{n}_I^{(k)}\right]\left(\mathbf{n}_I^{(k)} + \mu'\mathbf{n}_I^{(k)} \times \boldsymbol{\omega}\right) \tag{5.6}$$

where

$$\Delta\mathbf{v} := \tilde{\mathbf{v}}_I^{t+\Delta t,(k)} - \mathbf{v}_I^{\text{cm}}, \quad \boldsymbol{\omega} = \frac{\left[\Delta\mathbf{v} \times \mathbf{n}_I^{(k)}\right]}{\left|\left|\Delta\mathbf{v} \times \mathbf{n}_I^{(k)}\right|\right|}, \quad \mu' = \min\left[\mu, \frac{\left|\left|\Delta\mathbf{v} \times \mathbf{n}_I^{(k)}\right|\right|}{\Delta\mathbf{v} \cdot \mathbf{n}_I^{(k)}}\right] \tag{5.7}$$

For 2D problems, Equation (5.6) is explicitly given as

$$\begin{bmatrix} v_{xI}^{t+\Delta t,(k)} \\ v_{yI}^{t+\Delta t,(k)} \end{bmatrix} = \begin{bmatrix} \tilde{v}_{xI}^{t+\Delta t,(k)} \\ \tilde{v}_{yI}^{t+\Delta t,(k)} \end{bmatrix} - D\left(\begin{bmatrix} n_{xI}^{(k)} \\ n_{yI}^{(k)} \end{bmatrix} + \frac{\mu'}{||\mathbf{C}||}\begin{bmatrix} n_{yI}^{(k)}(\Delta v_x n_{yI}^{(k)} - \Delta v_y n_{xI}^{(k)}) \\ -n_{xI}^{(k)}(\Delta v_x n_{yI}^{(k)} - \Delta v_y n_{xI}^{(k)}) \end{bmatrix}\right) \tag{5.8}$$

with $\mu' = \min(\mu, ||\mathbf{C}||/D)$, $D = \Delta\mathbf{v}_I \cdot \mathbf{n}_I^{(k)}$. This equation is obviously reduced to Equation (5.5) for a frictionless contact if $\mu = 0$ and $D > 0$ since $\mu' = 0$.

5.1.3 Calculation of normal vector

The normal vector at grid nodes for each body is needed to complete the contact algorithm. Calculation of the normal vector $\mathbf{n}_I^{(k)}$ is crucial to obtaining accurate results [Lemiale et al., 2010, Nairn, 2007b].

The usual practice for finding the normal is to handle each body separately with relation to the system velocities. Thus, the normal is found from the mass gradient of the material under consideration. For each entity, the particle mass is interpolated to the element centers, $\mathbf{x}_c$ and divided by the element volume V_e, to obtain a density ρ_c. The gradient of ρ_c evaluated at the grid nodes provides the normal direction at the surface of each body [Sulsky and Brackbill, 1991, York, 1997, Bardenhagen et al., 2000].

The cell-centered density is defined by

$$\rho_c = \frac{1}{V_e}\sum_{p=1}^{n_p} m_p S^2(\mathbf{x}_p - \mathbf{x}_c) \tag{5.9}$$

where S^2 are bi-quadratic B-spline functions. In two dimensions, they are defined by $S^2 = S^x(x)S^y(y)$ where the one-dimensional quadratic B-spline function is given by

$$S^x(x) = \begin{cases} \frac{1}{2h_x^2}x^2 + \frac{3}{2h_x}x + \frac{9}{8}, & -\frac{3}{2h_x} \le x \le -\frac{1}{2h_x} \\ -\frac{1}{h_x^2}x^2 + \frac{3}{4}, & -\frac{1}{2h} \le x \le \frac{1}{2h_x} \\ \frac{1}{2h_x^2}x^2 - \frac{3}{2h_x}x + \frac{9}{8}, & \frac{1}{2h_x} \le x \le \frac{3}{2h_x} \\ 0, & \text{otherwise} \end{cases} \tag{5.10}$$

where h_x denotes the cell spacing in the x direction. Fig. 45 depicts some quadratic B-splines on a one-dimensional mesh. As can be seen, each particle contributes to three cells–the one it locates and the two neighbor cells. In 2D, each particle contributes to 9 cells and to 27 cells in 3D.

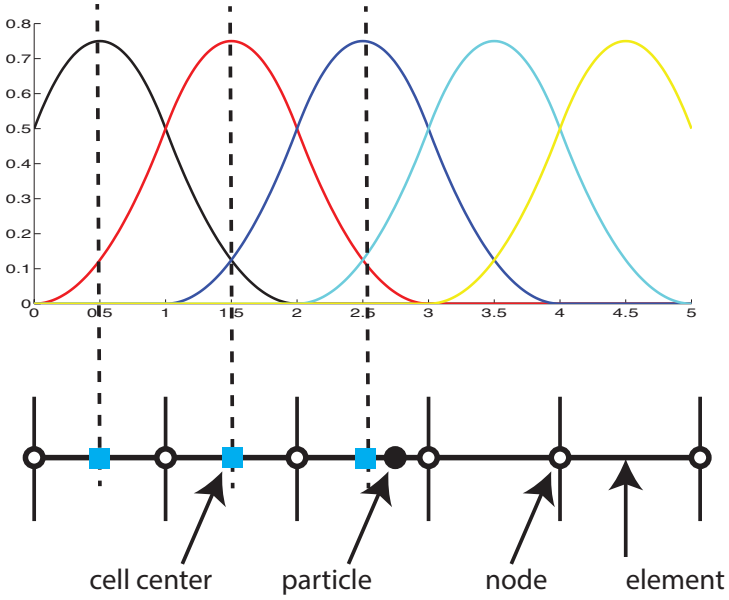


Figure 45: Quadratic B-spline functions used in defining cell center density.

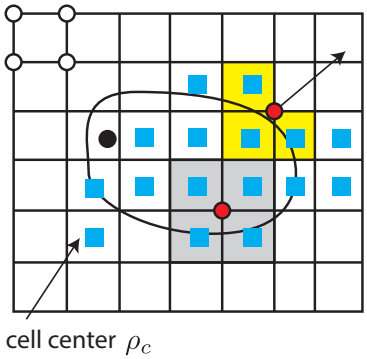


Figure 46: Computation of grid normal using Equation (5.11).

The grid normal vector is then given by

$$\mathbf{n}_I = \sum_c \nabla\phi_I(\mathbf{x}_c)\rho_c, \quad \mathbf{n}_I = \frac{\mathbf{n}_I}{||\mathbf{n}_I||} \tag{5.11}$$

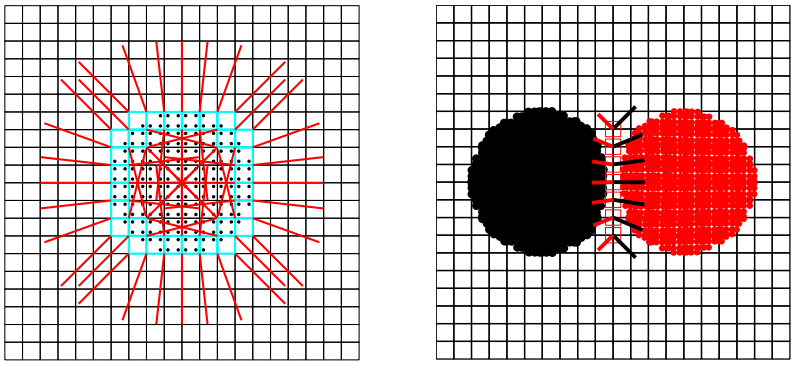
where $\nabla\phi_I$ denotes the gradient of the MPM weighting functions. Note that the sum is performed on the cells that have the node under consideration in their connectivity, cf. Fig. 46. This way of computing the normals is similar to SPH [Randles and Libersky, 1996].

Example on calculation of grid normals. Let us consider a circular disk which is represented by particles and a background grid. We are going to use Equations (5.9) and (5.11) to compute the normal vectors at boundary grid nodes. First, the density at the cell centers are computed using Equation (5.9). Next, we identify boundary elements which are those cut by the circle and are not empty. Finally, we loop over the boundary elements and their nodes and use Equation (5.11) to compute the grid normals. Fig. 47 shows the result.

Remark 32 *Recent works adopts a simpler method to compute the normals [Lemiale et al., 2010, Huang et al., 2011, Homel and Herbold, 2017]. In this method, the normals are simply calculated from the gradient of the particle mass:*

$$\mathbf{n}_I = \sum_p \nabla\phi_I(\mathbf{x}_p)m_p, \quad \mathbf{n}_I = \frac{\mathbf{n}_I}{||\mathbf{n}_I||} \tag{5.12}$$

There is no need to define the cell-centered density. We have tested the both ways, Equations (5.11) and (5.12) and found that they provide identical normals.



(a) Normals of a circle (b) Violation of collinearity of **n**

Figure 47: An example of normal vectors (red lines) defined at boundary nodes for a circular disk (a). The disk is represented by unconnected points or particles (black dots). Cyan squares are the boundary elements which are defined as those cut by the circle. Also shown are unneeded grid normals. Violation of collinearity of normal vectors in the MPM (b). This results in non-conversation of momentum [Bardenhagen et al., 2001]. This occurred because contacting is not handled at the true contact point but rather at the grid nodes which do not locate on neither body.

5.1.4 Algorithm

The previously presented contact algorithm can be incorporated into the standard MPM algorithm quite straightforwardly. Algorithm 6 is the algorithm for non-friction contact described in Section 5.1.1. One just need to modify line 21 for Coulomb friction. Even though the USF is shown, modification for USL is easy. One modification to the standard MPM code is that each grid node now contains the body velocity, mass and the system mass and velocity.

Algorithm 6 Solution procedure of an explicit contact MPM (USF).

Solve momentum equations for body b
  Mapping from particles to nodes
    Compute nodal mass $m_I^{t,(b)} = \sum_p \phi_I(\mathbf{x}_p^t) m_p$
    Compute nodal momentum $(m\mathbf{v})_I^{t,(b)} = \sum_p \phi_I(\mathbf{x}_p^t)(m\mathbf{v})_p^t$
    Compute body nodal velocities $\mathbf{v}_I^{t,(b)} = (m\mathbf{v})_I^{t,b} / m_I^{t,(b)}$
    Compute gradient velocity $\mathbf{L}_p^t = \sum_I \nabla\phi_I(\mathbf{x}_p)\mathbf{v}_I^{t,(b)}$
    Compute gradient deformation tensor $\mathbf{F}_p^t = (\mathbf{I} + \mathbf{L}_p^t \Delta t)\mathbf{F}_p^t$
    Update volume $V_p^t = \det \mathbf{F}_p^t V_p^0$
    Update stresses $\boldsymbol{\sigma}_p^t = \boldsymbol{\sigma}_p^t + \Delta\boldsymbol{\sigma}_p$
    Compute external force $\mathbf{f}_I^{\text{ext},t,(b)}$, internal force $\mathbf{f}_I^{\text{int,t,(b)}} = -\sum_{p=1}^{n_p} V_p^t \boldsymbol{\sigma}_p^t \nabla\phi_I(\mathbf{x}_p^t)$
    Compute nodal force $\mathbf{f}_I^{t,(b)} = \mathbf{f}_I^{\text{ext},t,(b)} + \mathbf{f}_I^{\text{int},t,(b)}$
  end
  Update the body momenta $(m\mathbf{v})_I^{t+\Delta t,(b)} = (m\mathbf{v})_I^{t,b} + \mathbf{f}_I^{t,(b)} \Delta t$
  Update the system momenta $(m\mathbf{v})_I^{t+\Delta t} = (m\mathbf{v})_I^{t+\Delta t} + (m\mathbf{v})_I^{t+\Delta t,b}$
  Update the system mass $m_I^t = m_I^t + m_I^{t,(b)}$
end
Correct velocity for contact nodes
  for contact node I of body b **do**
    Compute normal to b, $\mathbf{n}_I^{(b)}$
    Retrieve center of mass velocity $\mathbf{v}_I^{\text{cm}} = (m\mathbf{v})_I^{t+\Delta t} / m_I^t$
    Check contact or release $\alpha = \left(\mathbf{v}_I^{t+\Delta t,b} - \mathbf{v}_I^{\text{cm}}\right) \cdot \mathbf{n}_I^{(b)}$
    If $\alpha \geq 0$: $\mathbf{v}_I^{t+\Delta t,(b)} = \mathbf{v}_I^{t+\Delta t,b} - \left[(\mathbf{v}_I^{t+\Delta t,(b)} - \mathbf{v}_I^{\text{cm}}) \cdot \mathbf{n}_I^{(b)}\right] \mathbf{n}_I^{(b)}$
    If $\alpha < 0$: keep $\mathbf{v}_I^{t+\Delta t,(b)}$
    Compute nodal acceleration $\mathbf{a}_I^{t+\Delta t,(b)} = (1/\Delta t)(\mathbf{v}_I^{t+\Delta t,(b)} - \mathbf{v}_I^{t,(b)})$
  end for
end
Update particle positions and velocities
  Update particle velocities $\mathbf{v}_p^{t+\Delta t,b} = \mathbf{v}_p^{t,b} + \Delta t \sum_I \phi_I(\mathbf{x}_p)\mathbf{a}_I^{t+\Delta t,b}$
  Update particle positions $\mathbf{x}_p^{t+\Delta t,b} = \mathbf{x}_p^{t,b} + \Delta t \sum_I \phi_I(\mathbf{x}_p)\mathbf{v}_I^{t+\Delta t,b}$
end

Remark 33 *To the best of our knowledge MUSL stress update has not yet been used with frictional contacts. We anticipate that the reason for this is efficiency concerns. In the MUSL, one would need to do the contact treatment twice.*

5.1.5 Contact between a deformable solid and a rigid wall

In this section, the general contact algorithm previously presented for deformable solids is specialized to the case of contact between a solid and a rigid wall. Some examples are shown in Fig. 48. Unless for some particular cases when the rigid walls are either horizontal or vertical and stationary, rigid walls are usually discretized by the so-called rigid particles, cf. Fig 49. Let us denote the velocity of a rigid wall by $\mathbf{v}^{\text{r}}$ which is apparently zero for stationary rigid walls. Since the rigid wall has an infinite mass, the center-of-mass velocity defined in Equation (5.3) is actually $\mathbf{v}^{\text{r}}$. Therefore, the contact algorithm previously presented also applies with the following minor modifications: (1) the center-of-mass velocity is the velocity of the rigid wall (which is most often zero), (2) rigid particles are skipped in all standard MPM operations. Actions on them are only two things: define contact nodes and the normal vector. If the rigid bodies are moving with a constant velocity, then one needs to update their positions.

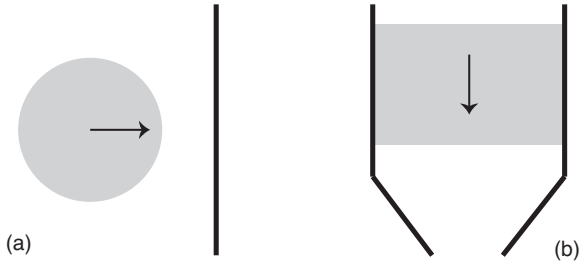


Figure 48: Contact between a deformable body and rigid walls: (a) moving ball impacting on a rigid wall and (b) silo discharging problem.

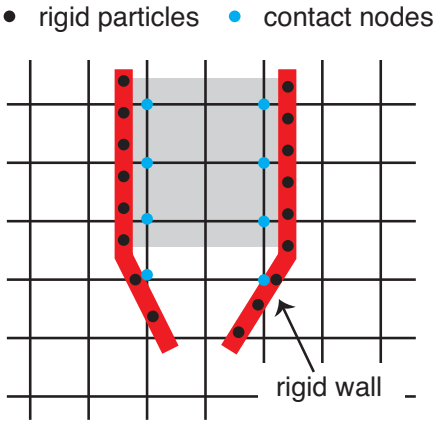


Figure 49: Rigid wall is represented by rigid particles.

5.1.6 Final remarks

The MPM can handle contacts quite efficiently as there is no need to find the contact surfaces and master/slave nodes commonly used in the FEM. We have just presented the most basic algorithm. There are refinements such as how to improve the normal vector calculation, see e.g., Lemiale et al. [2010], Nairn et al. [2018], and better contact detection criteria, see Bardenhagen et al. [2001].

5.2 Volumetric locking

It is well known that overly stiff numerical solutions appear when Poisson's ratio approaches 0.5 or when a plastic flow is constrained by the volume conservation condition. This is known as volumetric locking. And this is more severe with low order shape functions employed in the standard MPM. Various solutions to this problem have been proposed, all borrowing methods developed for the FEM [Love and Sulsky, 2006a,b, Mast et al., 2012, Yang et al., 2018, Coombs et al., 2018, Iaconeta et al., 2019]. In particular, Love and Sulsky [2006a], Mast et al. [2012] adopted the Hu–Washizu multi-field variational principle which introduces independent approximations for the volumetric and the deviatoric components of the strain and stress fields. On the other hand, a $\mathbf{u} - p$ mixed formulation was used in Iaconeta et al. [2019]. All these works employ the standard MPM which is known for its poor accuracy due to, among others, cell-crossing issue. Coombs et al. [2018] applied the F-bar method, developed for the FEM [de Souza Neto et al., 1996, Neto et al., 2005] to quasi-static MPM and GIMP.

In what follows we present this simple F-bar technique. We firstly recall key ideas of the F-bar method in the FEM and later adapt it to MPM. Note that this technique was implemented in the Uintah MPM code, where it is called *pressure stabilization*, long before Coombs et al. [2018].

The isochoric/volumetric split of the deformation gradient is defined as

$$\mathbf{F} = \mathbf{F}_{\text{iso}}\mathbf{F}_{\text{vol}} \tag{5.13}$$

where the isochoric and volumetric parts are given by

$$\mathbf{F}_{\text{iso}} = \frac{\mathbf{F}}{(\det \mathbf{F})^{1/3}}, \quad \mathbf{F}_{\text{vol}} = (\det \mathbf{F})^{1/3}\mathbf{I} \tag{5.14}$$

Note that $\det \mathbf{F}_{\text{iso}} = [(\det \mathbf{F})^{-1/3}]^3 \det \mathbf{F} = 1$, thus justify its name.

In the F-bar method, one defines the following modified gradient deformation tensor [Neto et al., 2005]

$$\bar{\mathbf{F}} = \left(\frac{\det \mathbf{F}_0}{\det \mathbf{F}}\right)^{1/3} \mathbf{F} \tag{5.15}$$

where $\mathbf{F}_0$ is the gradient deformation tensor evaluated at the centroid of the finite element. It can be shown that the isochoric/volumetric parts of the modified deformation tensor are given by

$$\begin{aligned} \bar{\mathbf{F}}_{\text{iso}} &= (\det \mathbf{F})^{-1/3}\mathbf{F} = \mathbf{F}_{\text{iso}} \\ \bar{\mathbf{F}}_{\text{vol}} &= (\det \mathbf{F}_0)^{1/3}\mathbf{I} = \mathbf{F}_{0,\text{vol}} \end{aligned} \tag{5.16}$$

i.e. the isochoric component of $\bar{\mathbf{F}}$ coincides with the current (integration point) isochoric deformation gradient while its volumetric part corresponds to the dilatation at the centroid of the element. Now, in order to compute the stresses at a particular integration point one uses the modified gradient deformation tensor rather than the original one. For nearly incompressible materials, the pressure variable is often decoupled from the stress and hence the pressure is a function of det $\bar{\mathbf{F}}$ and thus constant within a cell using the F-bar method.

In the MPM, the deformation gradient at the element centroid is replaced by the cell-centered or cell-averaged deformation gradient computed according to

$$\det \mathbf{F}_0^t = J_0^t = \frac{V_0^t}{V_0^0} = \frac{\sum_p m_p/\rho_p^0 J_p^t}{\sum_p m_p/\rho_p^0} \tag{5.17}$$

i.e., det $\mathbf{F}_0^t$ is defined as the ratio of the cell-centered current volume and the cell-centered initial volume. In the above equation, the sum is over all particles in the cell at hand, cf. Fig. 50. For implementation, the procedure is presented in Algorithm 7. We refer to a related work [Moutsanidis et al., 2019b] on this topic.

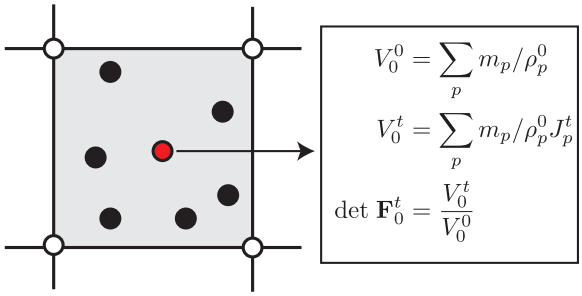


Figure 50: F-bar method applied for the material point method.

Algorithm 7 Algorithm for F-bar in an explicit MPM code.

for p=1:np **do**
    Compute/retrieve the particle deformation gradient and $J_p = \det \mathbf{F}_p$
    Get index of cell contains p, named c
    $cellVol0(c) = cellVol0(c) + (m_p/\rho_p^0)$
    $cellVol(c) = cellVol(c) + (m_p/\rho_p^0)J_p$
end for
Compute the centered gradient deformation determinant $cellJ = cellVol/cellVol0$
for p=1:np **do**
    Retrieve the particle deformation gradient and $J_p = \det \mathbf{F}_p$
    Get index of cell contains p, named c
    Retrieve the cell-centered $J0 = cellJ(c)$
    Compute modified gradient deformation $\bar{\mathbf{F}}_p = (J0/J_p)^{1/3}\mathbf{F}_p$
end for

5.3 Thermo-mechanical problems

Thermo-mechanical problems arise in several engineering applications, particularly manufacturing processes (e.g., welding processes, machining processes and hot/warm metal forming processes). In a thermo-mechanical problem, the unknowns are the deformation field (displacement/velocity) and the temperature field. The latter modifies the former via temperature-dependent constitutive models and the deformation affects the temperature via plastic work dissipated as heat.

This section presents a simple thermo-mechanical MPM formulation. First, the thermal problem is discussed in Section 5.3.1, then the coupled thermo-mechanical algorithm is given in Section 5.3.2 where the explicit MPM formulation treated in Section 2.5.2 is slightly modified to incorporate the temperature. For details, we refer to Nairn and Guilkey [2015], Tao et al. [2016].

5.3.1 Thermal problem

Our derivation of the MPM equation for the thermal problem is to adopt the FEM equation and use the material points as quadrature points. To get the FEM semi-discrete equation, we follow the standard procedure of going from the strong form to the weak form, followed by the introduction of the FE approximations of the trial and test functions. Details can be found in Tao et al. [2016]. Note that this thermal MPM algorithm follows the mechanical MPM one. For rectilinear geometries, thanks to the background Eulerian grid, a simple (and robust) finite difference method can be used, see e.g., Chen et al. [2008].

The partial differential equation for the thermal problem is the internal energy balance equation that reads

$$\rho c \dot{T} + \nabla \cdot \mathbf{q} = \gamma^* \qquad \text{in } \Omega \tag{5.18}$$

where ρ and c being the mass density and specific heat of the material, respectively and the heat source is represented by γ^*. In the above equation, $\mathbf{q}$ is the heat flux which is given by the following Fourier's law

$$\mathbf{q} = -k \nabla T \tag{5.19}$$

where k is the thermal conductivity. A thermal conductivity tensor can be equally used.

From Equation (5.18), one can obtain the following semi-discrete equation

$$C_{IJ} \dot{T}_J = Q_I^{\text{int}} + Q_I^{\text{ext}}, \quad I = 1, 2, \ldots, n_n \tag{5.20}$$

where

$$C_{IJ} := \int_\Omega \rho c \phi_I \phi_J \, \mathrm{d}\Omega, \quad Q_I^{\text{int}} := \int_\Omega \mathbf{q} \cdot \nabla \phi_I \, \mathrm{d}\Omega, \quad Q_I^{\text{ext}} := \int_\Omega \gamma^* \phi_I \, \mathrm{d}\Omega - \int_{\Gamma_q} \phi_I q^* \mathrm{d}\Gamma \tag{5.21}$$

where q^* is the prescribed heat flux similar to the tractions in solid mechanics. These quantities are approximated as follows in the spirit of the MPM

$$C_{IJ} = \sum_p m_p c_p \phi_I(\mathbf{x}_p) \phi_J(\mathbf{x}_p) \tag{5.22}$$

$$Q_I^{\text{int}} = \sum_p V_p \mathbf{q}_p \cdot \nabla \phi_I(\mathbf{x}_p), \quad Q_I^{\text{ext}} = \sum_p V_p \gamma^* \phi_I(\mathbf{x}_p) \tag{5.23}$$

where we have omitted the external force due to q^*, see Section 4.2.2 for its treatment.

Similar to the lumped mass matrix, the matrix C_{IJ} is made diagonal and thus Equation (5.20) is simplified to

$$C_I \dot{T}_I = Q_I^{\text{int}} + Q_I^{\text{ext}}, \quad C_I = \sum_p m_p c_p \phi_I(\mathbf{x}_p) \tag{5.24}$$

Again, in the same manner that the particle velocity is mapped to the grid nodes, one does the same thing for the temperature

$$T_I^t = \left(\frac{1}{C_I^t} \right) \sum_p \phi_I(\mathbf{x}_p^t) (mc)_p^t T_p^t \tag{5.25}$$

The complete algorithm is given in Algorithm 8. Note that a forward Euler was adopted to advance Equation (5.24) in time. Other time integration schemes can also be used. As can be seen, the algorithm adopts the MUSL scheme in which the updated particle temperature is mapped back to the grid and this updated grid temperature is used to compute the heat flux $\mathbf{q}$. Without doing so would result in very bad results. Moreover, we enforce the Dirichlet BCs on the nodes, but it can be enforced on the particles as done in Tao et al. [2016]. As this thermal MPM solver is meant to be coupled with a mechanical MPM solver, we use $\mathbf{x}_p^t$ instead of just $\mathbf{X}_p$ as in thermal analysis the particles do not move.

Algorithm 8 Solution procedure of explicit thermo MPM.

while $t < t_f$ **do**
  Mapping from particles to nodes (P2G)
    Compute nodal mass $C_I^t = \sum_p \phi_I(\mathbf{x}_p^t) m_p c_p$
    Compute nodal temperature $T_I^t = \left(\frac{1}{C_I^t}\right) \sum_p \phi_I(\mathbf{x}_p^t) m_p c_p T_p^t$
    Compute external force $Q_I^{\text{ext},t} = \sum_p V_p \gamma^* \phi_I(\mathbf{x}_p^t)$
    Compute internal force $Q_I^{\text{int},t} = \sum_p V_p \mathbf{q}_p^t \cdot \nabla \phi_I(\mathbf{x}_p^t)$
    Compute nodal force $Q_I^t = Q_I^{\text{ext},t} + Q_I^{\text{int},t}$
  end
  Update the temperature $\tilde{T}_I^{t+\Delta t} = T_I^t + Q_I^t \Delta t / C_I^t$
  Fix Dirichlet nodes I e.g., $T_I^t = T^*$ and $\tilde{T}_I^{t+\Delta t} = T^*$
  Update particles (G2P)
    Update particle temperature $T_p^{t+\Delta t} = T_p^t + \sum_I \phi_I(\mathbf{x}_p^t) \left[\tilde{T}_I^{t+\Delta t} - T_I^t\right]$
    Update grid temperature $T_I^{t+\Delta t} = \left(\frac{1}{C_I^t}\right) \sum_p \phi_I(\mathbf{x}_p^t) (mc)_p^t T_p^{t+\Delta t}$
    Fix Dirichlet nodes $T_I^{t+\Delta t} = T^*$
    Update flux $\mathbf{q}_p^{t+\Delta t} = -k \sum_I \nabla \phi_I(\mathbf{x}_p^t) T_I^{t+\Delta t}$
  end
  Advance time $t = t + \Delta t$
end while

5.3.2 Coupled thermo-mechanical MPM

By combining the thermal algorithm given in Section 5.3.1 and the mechanical algorithm in Section 2.5.2, one can come up with a simple coupled thermal-mechanical MPM formulation. In its most basic form, the two problems are solved on the same background grid and the same time step are used. The resulting algorithm is shown in Algorithm 9.

To close the thermal-mechanical MPM one needs a temperature-dependent constitutive model. For simplicity, we consider a thermo-elastic material model in which the stress is given by

$$\dot{\boldsymbol{\sigma}} = (\lambda \text{tr}\dot{\boldsymbol{\epsilon}}^e)\mathbf{I} + 2\mu\dot{\boldsymbol{\epsilon}}^e, \quad \dot{\boldsymbol{\epsilon}}^e = \dot{\boldsymbol{\epsilon}} - \dot{\boldsymbol{\epsilon}}^T, \quad \dot{\boldsymbol{\epsilon}}^T = \frac{\alpha}{\Delta t}(T^{t+\Delta t} - T^t)\mathbf{I} \tag{5.26}$$

where α is the coefficient of thermal expansion. The algorithm applies to other thermal-elasto-plastic materials.

Algorithm 9 Solution procedure of explicit thermo-mechanical MPM.

Initialization
  Set up the Cartesian grid, set time $t = 0$
  Set up particle data: $\mathbf{x}_p^0, \boldsymbol{\sigma}_p^0, \mathbf{F}_p^0, V_p^0, m_p$
end
while $t < t_f$ **do**
  Mapping from particles to nodes (P2G)
    Mechanical fields
      Compute nodal mass $m_I^t = \sum_p \phi_I(\mathbf{x}_p^t) m_p$
      Compute nodal momentum $(m\mathbf{v})_I^t = \sum_p \phi_I(\mathbf{x}_p^t)(m\mathbf{v})_p^t$
      Compute external force $\mathbf{f}_I^{\text{ext},t} = \sum_p \phi_I(\mathbf{x}_p) m_p \mathbf{b}(\mathbf{x}_p)$
      Compute internal force $\mathbf{f}_I^{\text{int},t} = -\sum_p V_p^t \boldsymbol{\sigma}_p^t \nabla \phi_I(\mathbf{x}_p^t)$
      Compute nodal force $\mathbf{f}_I^t = \mathbf{f}_I^{\text{ext},t} + \mathbf{f}_I^{\text{int},t}$
    end
    Thermal field
      Compute nodal mass $C_I^t = \sum_p \phi_I(\mathbf{x}_p^t) m_p c_p$
      Compute nodal temperature $T_I^t = \left(\frac{1}{C_I^t}\right) \sum_p \phi_I(\mathbf{x}_p^t) m_p c_p T_p^t$
      Compute external force $Q_I^{\text{ext},t} = \sum_p V_p \gamma^* \phi_I(\mathbf{x}_p^t)$
      Compute internal force $Q_I^{\text{int},t} = \sum_p V_p \mathbf{q}_p^t \cdot \nabla \phi_I(\mathbf{x}_p^t)$
      Compute nodal force $Q_I^t = Q_I^{\text{ext},t} + Q_I^{\text{int},t}$
    end
  end
  Update the momenta $(m\tilde{\mathbf{v}})_I^{t+\Delta t} = (m\mathbf{v})_I^t + \mathbf{f}_I^t \Delta t$
  Update the temperature $\tilde{T}_I^{t+\Delta t} = T_I^t + Q_I^t \Delta t / C_I^t$
  Fix mechanical Dirichlet nodes I e.g., $(m\mathbf{v})_I^t = \mathbf{0}$ and $(m\tilde{\mathbf{v}})_I^{t+\Delta t} = \mathbf{0}$
  Fix thermal Dirichlet nodes I e.g., $T_I^t = T^*$ and $\tilde{T}_I^{t+\Delta t} = T^*$
  Update particle velocities and grid velocities (double mapping)
    Get nodal velocities $\tilde{\mathbf{v}}_I^{t+\Delta t} = (m\tilde{\mathbf{v}})_I^{t+\Delta t} / m_I^t$
    Update particle velocities $\mathbf{v}_p^{t+\Delta t} = \alpha\left(\mathbf{v}_p^t + \sum_I \phi_I(\mathbf{x}_p^t)\left[\tilde{\mathbf{v}}_I^{t+\Delta t} - \mathbf{v}_I^t\right]\right) + (1-\alpha)\sum_I \phi_I(\mathbf{x}_p^t)\tilde{\mathbf{v}}_I^{t+\Delta t}$
    Update grid velocities $(m\mathbf{v}_I)^{t+\Delta t} = \sum_p \phi_I(\mathbf{x}_p^t)(m\mathbf{v})_p^{t+\Delta t}$
    Fix Dirichlet nodes $(m\mathbf{v})_I^{t+\Delta t} = \mathbf{0}$
    Update particle temperature $T_p^{t+\Delta t} = T_p^t + \sum_I \phi_I(\mathbf{x}_p^t)\left[\tilde{T}_I^{t+\Delta t} - T_I^t\right]$
    Update grid temperature $T_I^{t+\Delta t} = \left(\frac{1}{C_I^t}\right) \sum_p \phi_I(\mathbf{x}_p^t)(mc)_p^t T_p^{t+\Delta t}$
    Fix Dirichlet nodes $T_I^{t+\Delta t} = T^*$
  end
  Update particles (G2P)
    Get nodal velocities $\mathbf{v}_I^{t+\Delta t} = (m\mathbf{v})_I^{t+\Delta t} / m_I^t$
    Update particle positions $\mathbf{x}_p^{t+\Delta t} = \mathbf{x}_p^t + \Delta t \sum_I \phi_I(\mathbf{x}_p^t)\mathbf{v}_I^{t+\Delta t}$
    Compute gradient velocity $\mathbf{L}_p^{t+\Delta t} = \sum_I \nabla \phi_I(\mathbf{x}_p^t)\mathbf{v}_I^{t+\Delta t}$
    Updated gradient deformation tensor $\mathbf{F}_p^{t+\Delta t} = (\mathbf{I} + \mathbf{L}_p^{t+\Delta t}\Delta t)\mathbf{F}_p^t$
    Update volume $V_p^{t+\Delta t} = \det \mathbf{F}_p^{t+\Delta t} V_p^0$
    Update stresses $\boldsymbol{\sigma}_p^{t+\Delta t} = \boldsymbol{\sigma}_p^t + \Delta \boldsymbol{\sigma}_p(\mathbf{L}_p^{t+\Delta t}, T_p^{t+\Delta t})$
    Update flux $\mathbf{q}_p^{t+\Delta t} = -k \sum_I \nabla \phi_I(\mathbf{x}_p^t) T_I^{t+\Delta t}$
  end
  Advance time $t = t + \Delta t$
end while

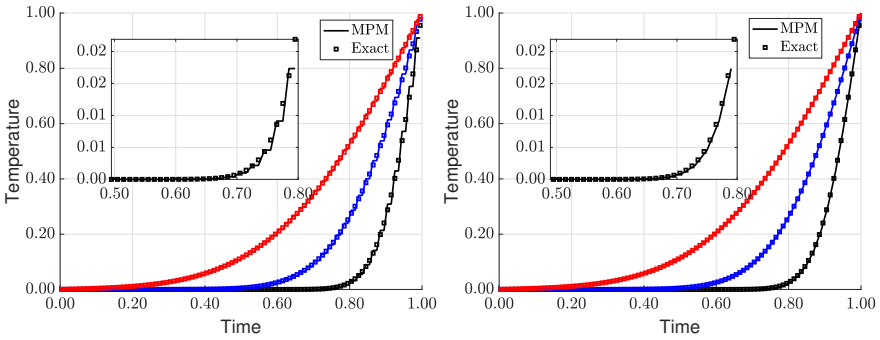
5.3.3 Verification tests

We present two simple tests for the verification of the thermal MPM algorithm. We refer to Tao et al. [2016] for coupled thermal-mechanical tests.

One dimensional thermal test. Let consider a bar of length $L = 1$, with $\rho = c = k = 1$. The bar temperature is initially set at $T_0 = 0$ and the right extremity is heated up to T_1 at time $t = 0$ and held fixed. Any set of consistent units suffice. The exact solution for the temperature in the bar is given by Tao et al. [2016]

$$T^{\text{exact}}(x,t) = T_0 + (T_1 - T_0)x + 2(T_1 - T_0)\sum_{n=1}^{\infty} \frac{(-1)^n}{n\pi} \exp(-[n\pi]^2 t)\sin(n\pi x) \tag{5.27}$$

The MPM solutions, with $T_0 = 0$ and $T^* = T_1 = 1$, are given in Fig. 51 in terms of particle temperature and grid temperature. The time step is chosen to be $\Delta t = 8 \times 10^{-5}$ which is smaller than the theoretical maximum value of $(h_x)^2/(2k)$. However, we find that larger time steps also yield accurate results. The results given in the referred figure confirms the finding in the literature that the particle data are noisy whilst the grid data (i.e., $T_I^{t+\Delta t}$ not $\tilde{T}_I^{t+\Delta t}$) are not.



(a) particle temperature (b) grid temperature

Figure 51: One dimensional heat conduction: solutions obtained with the standard MPM with 100 grid cells and 1 particle per cell. Black is for $t = 0.0075$, blue is for $t = 0.03$ and red is for $t = 0.05$.

Two dimensional thermal test. Let consider a rectangular plate of dimensions $L \times H$, with $\rho = c = k = 1$. The bar temperature is initially set at $T_0 = 0$ and the external surface is heated up to $T_1 = 100$ at time $t = 0$ and held fixed. Any set of consistent units suffice. The exact temperature field in the plate is given by Tao et al. [2016]

$$T^{\text{exact}}(x,y,t) = T_1 + \frac{16(T_0 - T_1)}{\pi^2} \sum_{i=1,3,\ldots} \sum_{j=1,3,\ldots} \frac{\exp(-\pi^2(i^2/L^2 + j^2/H^2)t)}{ij} \sin\left(\frac{i\pi x}{L}\right) \sin\left(\frac{j\pi y}{H}\right) \tag{5.28}$$

In the simulations, a unit square is considered and a grid of 20×20 cells with four particles per cell is used. The temperature boundary condition is applied on the particles residing in the boundary cells. The nodal temperature of both MPM and exact solutions are given in Fig. 52 at time $t = 0.05$. A constant time step of $\Delta t = 10^{-4}$ was adopted. Note again that, the particle temperature distribution (not shown) is not as smooth as the nodal temperature field.

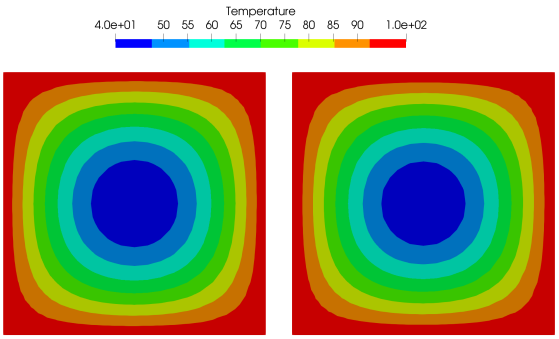


Figure 52: Two dimensional heat conduction: standard MPM solutions (left) and exact solution (right).

5.4 Fluids and gases

Even though MPM has been developed for solid mechanics applications, it has also been used to model fluids and gases. This section presents a brief discussion on the so-called weakly compressible MPM for fluids. We refer to Zhang et al. [2017] and references therein for advanced incompressible MPM formulations for free surface flow. This simple weakly compressible formulation allows us to model various interesting FSI problems, see York et al. [1999, 2000], Gan et al. [2011], Mao [2013], Yang et al. [2018], Su et al. [2019].

The governing equations for fluids/gases are the same as for the solid (Equation (2.20)) except the constitutive model. An artificial equation of state is adopted to describe the pressure. Therefore, an MPM code for solids can be equally used to model fluids/gases.

5.4.1 Fluids

For fluids, the stress field is defined as

$$\boldsymbol{\sigma}^f = 2\mu_N \dot{\boldsymbol{\epsilon}} + \lambda_N \mathrm{tr}(\dot{\boldsymbol{\epsilon}})\mathbf{I} - \hat{p}\mathbf{I} \tag{5.29}$$

where $\hat{p}$ is the pressure which is determined from an equation of state (EOS), μ_N and λ_N are the shear viscosity [Pa s = kg/(ms)] and the bulk viscosity, respectively. Recall that the strain rate tensor is defined as $\dot{\boldsymbol{\epsilon}} = 0.5(\mathbf{L} + \mathbf{L}^{\mathrm{T}})$ where $\mathbf{L}$ denotes the gradient velocity tensor. A superscript f was used to label the fluid stress, as in an FSI problem, one has to deal with two stress fields – one for the solid and one for the fluid.

If $\lambda_N = \frac{2\mu_N}{3}$, the so-called Stokes condition, the stress field thus becomes

$$\boldsymbol{\sigma}^f = 2\mu_N \dot{\boldsymbol{\epsilon}} - \frac{2\mu_N}{3}\mathrm{tr}(\dot{\boldsymbol{\epsilon}})\mathbf{I} - \hat{p}\mathbf{I} = 2\mu_N \left[\dot{\boldsymbol{\epsilon}} - \frac{1}{3}\mathrm{tr}(\dot{\boldsymbol{\epsilon}})\mathbf{I}\right] - \hat{p}\mathbf{I} \tag{5.30}$$

where the term in the bracket is the deviatoric part of the strain rate tensor.

The pressure of the fluid particle is updated by an EOS. In the case of water and air, the EOS is given by [Monaghan, 1994, Cueto-Felgueroso et al., 2004]

$$\hat{p} = \kappa \left[\left(\frac{\rho}{\rho_0}\right)^{\gamma} - 1\right] \tag{5.31}$$

where ρ_0 is the initial density and κ is the bulk modulus [Pa=kg/(ms^2)] chosen such that the fluid is nearly incompressible and $\gamma = 7$ for water and $\gamma = 1.4$ for air. The advantage of this EOS, providing a direct relationship between pressure and density, is that there is no need to solve any additional equation for the pressure. The bulk modulus can be very high for a nearly incompressible fluid, such as water, resulting in a very small time step. To increase the time step, a reduced bulk modulus can be used as long as the change in density is less than 3%.

5.4.2 Gases

And for an ideal gas, the stress field is given by [Hu and Chen, 2006]

$$\boldsymbol{\sigma}^f = -\hat{p}\mathbf{I}, \quad \hat{p} = (\gamma - 1)\rho e \tag{5.32}$$

where e is the specific internal energy and γ is the ratio of specific heats. The specific internal energy is updated using the balance of energy equation (thermal effect was neglected)

$$e_p^{t+\Delta t} = e_p^t + \Delta t \boldsymbol{\sigma}_p^{t+\Delta t} : \Delta \mathbf{e}_p^{t+\Delta t} / \rho_p^{t+\Delta t} \tag{5.33}$$

where the density is updated using the following equation

$$\rho_p^{t+\Delta t} = \frac{\rho_p^t}{1 + \Delta t \mathrm{tr}(\Delta \mathbf{e}_p^{t+\Delta t})} \tag{5.34}$$

It is well known that most numerical simulations of compressible-fluid shocks provide more accurate results if some type of artificial viscosity is used at the shock front. The following artificial viscosity is added to the particle pressure, see e.g., Zhang et al. [2016] for details

$$q = \rho L_e \left(c_0 L_e \dot{\epsilon}_{kk}^2 - c_1 a \dot{\epsilon}_{kk}\right), \quad \dot{\epsilon}_{kk} < 0 \tag{5.35}$$

where c_1 and c_2 are coefficients of artificial viscosity, $a = \sqrt{\gamma \hat{p}/\rho}$ is the local sound speed and L_e is the minimum length of cell sides of the background grid.

Dam break problems. A simplified dam break simulation with a barrier is depicted in Fig. 53. A water column of initial height h_0 and length l_0 is initially constrained by a gate and rests on a smooth, flat surface. At an arbitrary starting time, say $t = 0$, the gate is removed and the water is allowed to flow freely under the force of gravity. This problem has been tackled extensively in the SPH literature and also studied using the MPM by e.g., Mast et al. [2012] where it was demonstrated that the standard MPM inhibits severe locking.

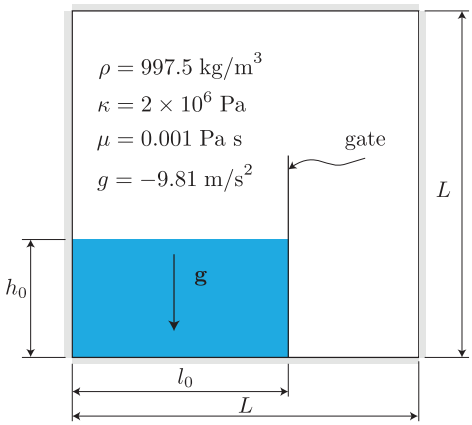


Figure 53: Dam break problem. The computational domain is $L \times L$ with $L = 6$ m, $l_0 = 4$ m, $h_0 = 2$ m.

The behavior of the water follows Equations (5.30) and (5.31). The domain is discretized by 70×70 cells with 4 PPC (4 371 particles). A constant time step of $\Delta = 0.1 h_x/c$ with $c = \sqrt{\kappa/\rho}$ was used. The component normal to the boundaries of the grid velocities are set to zero. The aims of this example are two-fold. First, the MPM is demonstrated for fluid flow. Second, it shows that the standard MPM (with hat functions) and MUSL does not subject to volumetric locking, or probably it locks but to a very small extent. Also, even though not shown here, GIMP and BSMPM do not lock, either with USL or MUSL.

The simulation snapshots are given in Fig. 54. Note that, for fluid problems, one should use $\alpha = 1$ in the particle velocity update (Equation (2.53)).

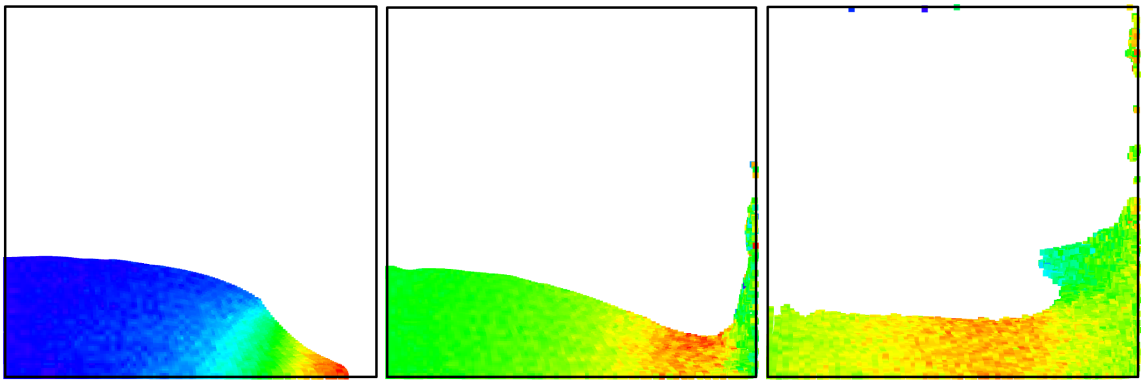


Figure 54: Dam break problem. Simulation snapshots: from left to right, $t = 0.4$ s, $t = 0.72$ s and $t = 1.70$ s.

Remark 34 *For solid mechanics problems exhibiting plastic deformation given in Section 6, we have also not observed volumetric locking. These simulations were done with the standard MPM and the MUSL without pressure stabilization (Section 5.2). We think that there is a need to re-examine locking issues in the MPM using MUSL and the pressure stabilization to verify its performance compared with advanced but complex methods such as the one of Mast et al. [2012].*

5.5 Improved MPM formulations

Convergence tests (presented in Section 6.2) reveal that MPMs do not converge at an optimal rate (i.e., the convergence rate for L_2 error of the displacement is not two). There are many reasons for this undesired behavior. It is quite clear that

quadrature error is one of them as particles are arbitrarily positioned on the grid. Methods to improve this error include MPM variants with smooth weighting functions (GIMP/CPDI/BSMPM) and the standard MPM with the Gaussian quadrature (that is particle data are reconstructed at the quadrature points and they are used for evaluating the weak form integrals similar to the FEM). In Wallstedt and Guilkey [2007] it has been shown that the grid momenta calculation, with linear shape functions, is not able to provide an exact projection of a linear velocity field for arbitrary particle positions. This led Sulsky and Gong [2016] to develop the improved MPM (iMPM) where moving least square (MLS) method is adopted to reconstruct the particle data (velocity, density and stress) at the grid nodes and cell centers. Wobbes et al. [2019] presented a similar idea using local Taylor basis instead of MLS. This section presents the theory and implementation of the iMPM of Sulsky and Gong [2016].

5.5.1 Velocity (momentum) projection

In Wallstedt and Guilkey [2007] it has been shown that Equation (2.43), with linear shape functions, is not able to provide an exact projection of a linear velocity field for arbitrary particle positions. And this introduces another source of error in the MPM–mapping error. They also proposed a technique to reduce this error. As a simple demonstration of this, we consider a grid of three equally spaced cells ($h = 2$) with 3 particles located at the cell centers. Assume that all the particles have a mass of unity and the particle velocity field is linear i.e., $v(x) = x$, cf. Fig. 55. Obviously, Equation (2.43) results in $v_1 = 1$ (correct value is 0) and $v_4 = 5$ (correct value is 6). The situation is getting worse with off-center particles. As shown in Sulsky and Gong [2016] and in Appendix D, MLS can solve this issue.

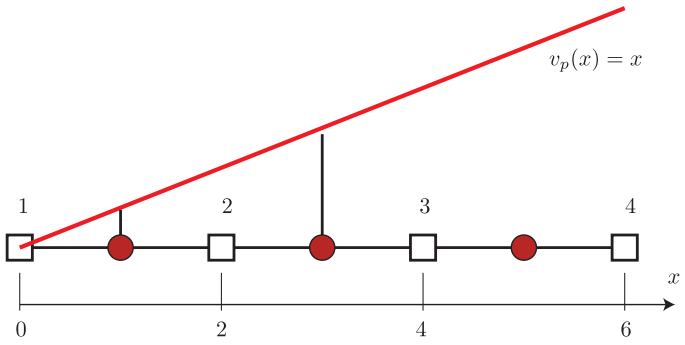


Figure 55: Error in projecting particle velocities to grid nodes. With particles located at cell centers, the internal nodes have correct projected velocities while the boundary nodes do not.

The affine PIC. In order to minimize the velocity projection error, Wallstedt and Guilkey [2007] suggested to enhance the particle's velocities using the already available velocity gradient. In 1D, the particle to grid projection was therefore defined as:

$$m_i \mathbf{v}_I = \sum_p m_p \Phi_p(\mathbf{x}_I) \left[\mathbf{v}_p - \frac{\partial \mathbf{v}_p}{\partial \mathbf{x}} (\mathbf{x}_p - \mathbf{x}_I) \right] \tag{5.36}$$

More recently, the same idea was followed by Jiang et al. [2015a] in the computer graphics community, but extended to 3D. This variation of MPM was named the Affine Particle-In-Cell (APIC) method, which is a bit confusing as the method is indeed an MPM. The APIC particle velocity to grid projection is given by

$$m_i \mathbf{v}_I = \sum_p m_p \Phi_p(\mathbf{x}_I) \left[\mathbf{v}_p + \mathbf{B}_p \mathbf{D}_p^{-1} (\mathbf{x}_I - \mathbf{x}_p) \right] \tag{5.37}$$

where

$$\mathbf{B}_p = \sum_I \Phi_p(\mathbf{x}_I) \mathbf{v}_I (\mathbf{x}_I - \mathbf{x}_p)^{\mathrm{T}}, \quad \mathbf{D}_p = \sum_I \Phi_p(\mathbf{x}_I) \mathbf{x}_{Ip} \mathbf{x}_{Ip}^{\mathrm{T}} \tag{5.38}$$

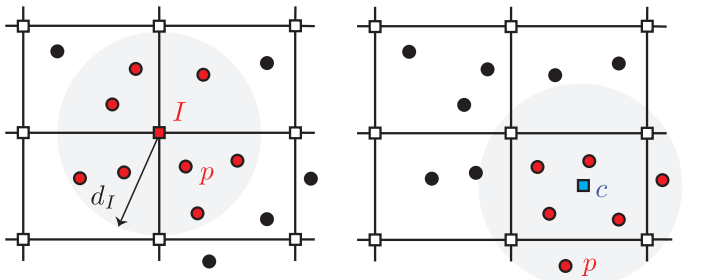
Remark 35 *Jiang et al. [2015a] proved that APIC conserves angular momentum. We have implemented it in* `Karamelo`*, but it does not work well with the TLMPM. There is a need to check the performance of APIC for engineering problems as this method is simple.*

5.5.2 The improved MPM (iMPM)

In order to have a high-order MPM, one needs to fullfill the following conditions: (1) high order grid basis functions such as cubic B-splines, (2) a proper projection of the particle velocities to the grid and (3) accurate integration of the weak form integrals (the nodal mass and forces). Note that lack of any of these three conditions would result in a first-order MPM for large deformation problems where the particle positions are arbitrary with respect to the grid. Furthermore the stress update of the constitutive equations must also be high order to have an overall high order accurate method. Herein we focus on the last two conditions as cubic B-splines have been discussed. The device to fulfill these last conditions is moving least square approximation [Gong, 2015, Sulsky and Gong, 2016]. The ideas are as follows

- Reconstruction of particle momenta using MLS and project this reconstructed momenta to the grid nodes. This will solve the velocity projection issue of MPM (including GIMP, CPDI);
- Integration of the mass matrix and nodal forces using one-point quadrature rule located at the element centers. This will improve the quadrature error of MPM. Note that CPDI does not subject to this error with the price of a particle FE mesh. Again MLS is used to construct the data at the element centers. Precisely the particle stresses and densities are approximated using MLS and they are computed at the element centers.
- Standard FE shape functions i.e., the hat functions are still employed in the calculation of the mass matrix and internal/external forces.

Graphical illustration of the two MLS approximations involved in the above the first two items is given in Fig. 56. The last point warrants a further discussion.



(a) Velocity projection (b) Density/stress projection

Figure 56: Moving least square approximations (a) to project particle velocities to the grid and (b) to construct cell-centered densities and stresses.

Remark 36 *Using one-point quadrature rule improves the numerical integration error and eliminates the cell-crossing error (as the weighting gradients are always evaluated at the cell centers). Yet it is not accurate for cells on the solid boundary. A mixed quadrature was presented in [Song et al., 2019] where for fully filled cells, one-point quadrature is used (similar to iMPM) and for partially filled cells, material point based integration is used. Partially filled cells are cells intersecting with the solid boundaries.*

Remark 37 *Mixed quadrature was used for the first time, to the best of our knowledge, in the geo-technical engineering community [Beuth et al., 2011]. The motivation was due to the fact that particle based quadrature resulted in oscillation in the stress field. Even though this oscillation was due to cell crossing [Bardenhagen and Kober, 2004] which can be removed using C^1 weighting functions, constructing such smooth functions is not easy with an unstructured grid.*

Velocity projection. From the particle velocities one can obtain the grid node velocities as follows

$$\mathbf{v}_I = \sum_p \Phi_p^{\text{MLS}}(\mathbf{x}_I)\mathbf{v}_p \tag{5.39}$$

where the sum is over the particles within the domain of influence of node I; $\Phi_p^{\text{MLS}}(\mathbf{x}_I)$ denotes the MLS shape functions, of which details can be found in Appendix D.

One point quadrature. The nodal mass and nodal forces (internal and external) are numerically computed using only one quadrature point–the element center $\mathbf{x}_c$. Therefore, we have

$$\begin{aligned} m_I &\approx \sum_{c=1}^{nc} \rho_c N_I(\mathbf{x}_c) V_c \\ \mathbf{f}_I^{\text{int}} &\approx -\sum_{c=1}^{nc} \boldsymbol{\sigma}_c \nabla N_I(\mathbf{x}_c) V_c \end{aligned} \tag{5.40}$$

where V_c denotes the volume of the element ($V_c = h_x h_y$ for 2D), and nc is the number of elements surrounding node I. For interior nodes, $nc = 2$ in 1D, $nc = 4$ in 2D and $nc = 8$ in 3D. It should be emphasized that using one single quadrature point might not be sufficient. Either hourglass control can be added or full quadrature can be employed. Note that, in Equation (5.40), we use $N_I(\mathbf{x})$ not $\phi_I(\mathbf{x})$ as will be seen, the hat functions are sufficient to get a quadratic convergence rate.

The density and stresses at the element center are computed using the MLS approximation:

$$\begin{aligned} \rho_c &= \sum_p \Phi_p^{\text{MLS}}(\mathbf{x}_c) \rho_p \\ \boldsymbol{\sigma}_c &= \sum_p \Phi_p^{\text{MLS}}(\mathbf{x}_c) \boldsymbol{\sigma}_p \end{aligned} \tag{5.41}$$

where the sum is over the particles within the domain of influence of the element center $\mathbf{x}_c$. One can consider the cell centers as stress points often used in SPH [Dyka and Ingel, 1995]

Implementation. This section presents the implementation of the iMPM. The complete flowchart is given in Algorithm 10 which requires some modifications of a standard MPM code only. First, we introduce two data structures to store the cell-centered density and stress. Second, in the 'P2G' step, we proceed in two sub-steps: (1) computation of nodal mass and forces by looping over the elements and then over the centers and (2) computation of the nodal velocities by sweeping over the grid nodes. Third, in the 'G2P' step, we construct the cell-centered density and stress from the newly updated particle density and stress. There is one change to the initialization step as well–one needs to initialize the cell-centered density and stresses in addition to conventional particle data.

Algorithm 10 iMPM with USL: step from t to $t + \Delta t$.

Mapping from particles to nodes (P2G)
  Compute nodal mass $m_I^t = \sum_c N_I(\mathbf{X}_c) \rho_c^t V_c$
  Compute nodal velocities $\mathbf{v}_I^t = \sum_p \Phi_p^{\text{MLS}}(\mathbf{x}_I) \mathbf{v}_p^t$
  Compute internal force $\mathbf{f}_I^{\text{int},t} = -\sum_c V_c \boldsymbol{\sigma}_c^t \nabla N_I(\mathbf{X}_c)$
  Compute nodal force $\mathbf{f}_I^t = \mathbf{f}_I^{\text{ext},t} + \mathbf{f}_I^{\text{int},t}$
end
Update velocities $\mathbf{v}_I^{t+\Delta t} = \mathbf{v}_I^t + \left(\mathbf{f}_I^t / m_I^t\right) \Delta t$
Fix Dirichlet nodes
Update particles + construct cell-centered density/stress (G2P)
  Update particle velocities $\mathbf{v}_p^{t+\Delta t} = \mathbf{v}_p^t + \sum_I N_I(\mathbf{x}_p^t)(\mathbf{v}_I^{t+\Delta t} - \mathbf{v}_I^t)$
  Update particle positions $\mathbf{x}_p^{t+\Delta t} = \mathbf{x}_p^t + \Delta t \sum_I N_I(\mathbf{x}_p^t) \mathbf{v}_I^{t+\Delta t}$
  Compute gradient velocity $\mathbf{L}_p^{t+\Delta t} = \sum_I \nabla \phi_I(\mathbf{x}_p^t) \mathbf{v}_I^{t+\Delta t}$
  Updated gradient deformation tensor $\mathbf{F}_p^{t+\Delta t} = (\mathbf{I} + \mathbf{L}_p^{t+\Delta t} \Delta t) \mathbf{F}_p^t$
  Update volume and stress $V_p^{t+\Delta t} = \det \mathbf{F}_p^{t+\Delta t} V_p^0$, $\boldsymbol{\sigma}_p^{t+\Delta t} = \boldsymbol{\sigma}_p^t + \Delta \boldsymbol{\sigma}_p$
  Update density $\rho_p^{t+\Delta t} = \rho_p^0 / J$
  Construct cell-centered density $\rho_c^{t+\Delta t} = \sum_p \Phi_p^{\text{MLS}}(\mathbf{X}_c) \rho_p^{t+\Delta t}$
  Construct cell-centered stress $\boldsymbol{\sigma}_c^{t+\Delta t} = \sum_p \Phi_p^{\text{MLS}}(\mathbf{X}_c) \boldsymbol{\sigma}_p^{t+\Delta t}$
end

In order to make it general the MLS approximation for the velocity can be different from the MLS approximation used for the density/stress. It is well known that MLS shape functions are computationally expensive and so is iMPM. Therefore, Tran et al. [2019] adopted an improved MLS, developed by Liew et al. [2005], where orthogonal basis are used leading to a diagonal moment matrix which is trivial for inverting.

5.6 Adaptivity

A basic MPM implementation adopts a fixed uniform Cartesian (or unstructured) grid and a fixed number of material points. In order to better resolve regions of high gradients while keeping a reasonable computational cost, adaptive grid has been proposed, see Section 5.6.1. Accompany with grid refinement is particle splitting in which original particles are split into new ones to be added to the new grid cells. Another situation where particle splitting is needed is to prevent numerical fracture. This particle splitting is discussed in Section 5.6.2.

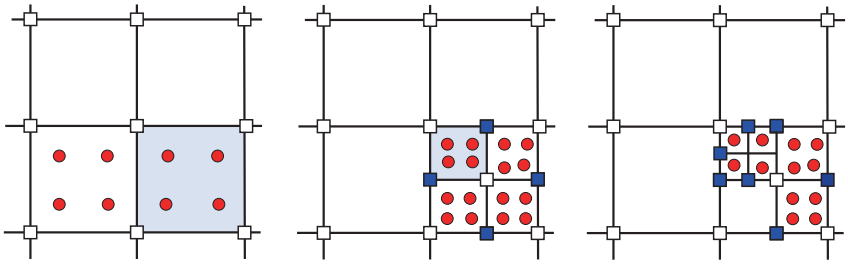


Figure 57: Multi-level grid refinement and particle splitting. Shaded cells are the ones needed to be refined. Solid squares denote hanging nodes.

5.6.1 Grid adaptive refinement

Adaptive refinement MPM was reported in Tan and Nairn [2002] for fracture mechanics applications. Details are not provided on how to handle hanging nodes. Ma et al. [2006] presented a multi-level grid refinement for GIMP. The standard grid functions are modified to handle hanging nodes, and the modified functions are convoluted with the particle characteristic functions, in the conventional GIMP way, to obtain the final weighting function $\phi_I(\mathbf{x}_p)$. A similar method was given in the community of computer graphics [Gao et al., 2017]. Cheon and Kim [2019] reports a similar grid refinement, for the standard MPM, within the context of phase-field fracture simulation. In the field of free surface flows, Mao et al. [2015] reported an adaptive MPM to accurately handle free surfaces. The algorithm allows particle splitting and merging.

5.6.2 Particle splitting and merging

We confine to 2D problems for simplicity. Assume that a given particle is split into 4 particles. The mass and volume of the new four particles are 1/4 of the corresponding values for the original particle. All intrinsic material state properties (e.g., density, deformation gradient, stress, damage, etc.) are set equal to that of the original particle [Homel et al., 2016]. When it comes to when one should perform particle splitting there exists different criteria, see Ma et al. [2009a], Gracia et al. [2019]. Basically, when a particle is stretched too much, it is split. The splitting criterion uses the local particle data such as $\mathbf{F}$, its original/current size and the grid cell size.

6 Numerical examples

This section presents some simulations carried out by the authors using various in-house MPM codes that we have developed including a Matlab code, a Julia code and two C++ codes. Most of these codes are open source and thus these simulations can be reproduced. These simulations cover signature MPM simulations: large deformation, contact and fracture. But they are certainly not representative of all interesting simulations performed using the MPM reported in the literature.

We divide the simulations into three sets. The first contains *implementation tests*: simple problems used to verify MPM implementations (Section 6.1). The second set, given in Section 6.2 and named *convergence tests*, consists of fabricated simulations using the method of manufactured solutions to verify the convergence of MPMs (whether they converge and at which rate) and expose their algorithmic shortcomings. The third set, presented in Section 6.3, provides some practical

engineering simulations which are validated against corresponding experiments. Unless otherwise stated, the particle velocity is updated using FLIP ($\alpha = 1$).

6.1 Implementation tests

This section presents some simple tests served to verify an MPM implementation. A one dimensional vibrating bar, with analytical solutions, is discussed in Section 6.1.1. This test also discusses energy conservation of various MPM algorithms. Collision of 2D elastic bodies is given in Section 6.1.2. These two tests are simple as the material is linear elastic and yet sufficient to verify an MPM code for solid mechanics problems. For ductile solids, simulation of the impact of an elastic disk onto an aluminum target is presented in Section 6.1.3. Finally, a test on gas dynamics is given in Section 6.1.4. This test serves to verify the implementation of the MPM for fluids and gases. For other benchmark tests for fluid mechanics, we refer to Zhao et al. [2017], Vargas et al. [2018], Sun et al. [2018].

6.1.1 Axial vibration of a continuum bar

In this example, we study the axial vibration of a continuum bar with Young's modulus $E = 100$, density $\rho = 1$, and the bar length $L = 25$. One end ($x = 0$) of the bar is fixed, and the other ($x = L$) is free. Any set of consistent unit suffice.

Exact solutions for mode n are (refer to e.g., Bardenhagen [2002])

$$v(x,t) = v_0 \cos(\omega_n t)\sin(\beta_n x) \tag{6.1}$$

$$u(x,t) = \frac{v_0}{\omega_n}\sin(\omega_n t)\sin(\beta_n x) \tag{6.2}$$

where $\omega_n = \beta_n\sqrt{E/\rho}$ and $\beta_n = \frac{2n-1}{2}\frac{\pi}{L}$. The period of vibration is $2\pi/\omega_1 = 10$. In the computations, $v_0 = 0.1$ was used. The initial velocity is given by

$$v(x,0) = v_0 \sin(\beta_n x) \tag{6.3}$$

The grid consists of 13 two-noded line elements (14 grid nodes) and 13 material points (i.e., one particle per element) placed at the center of the elements are used. The time increment is chosen as $\Delta t = 0.1\Delta x/c$ where Δx denotes the nodal spacing and $c = \sqrt{E/\rho}$. We consider two modes – mode 1 ($n = 1$) and mode 10 ($n = 10$) and for both cases, the quantity of interest used to compare the numerical and exact solution is the center of mass velocities which are given by

$$v_{\text{cm}}^{\text{exa}}(t) = \frac{v_0}{\beta_n L}\cos(\omega_n t); \quad v_{\text{cm}}^{\text{num}}(t) = \frac{\sum v_p(t) m_p}{\sum m_p} \tag{6.4}$$

for the exact solution and the MPM solution, respectively.

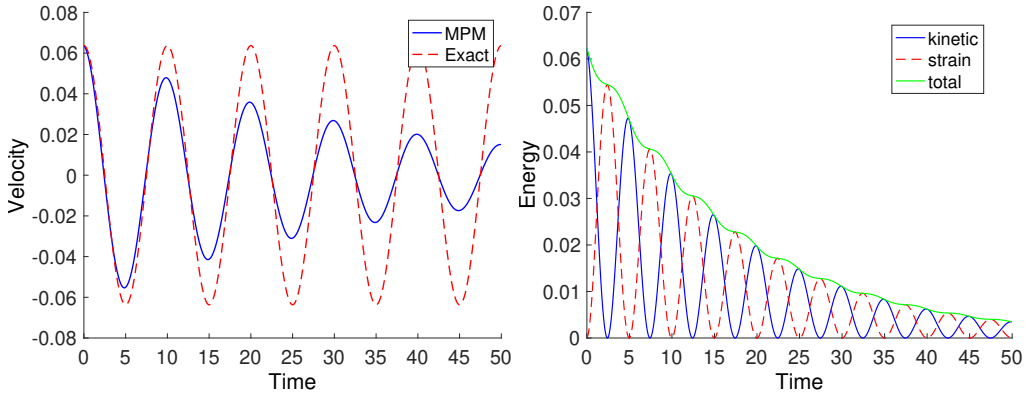
The results are given in Fig. 58 for mode 1 and in Fig. 59 for mode 10, respectively. As can be seen from Fig. 58 , PIC results in a big numerical dissipation while FLIP is not. For this mode, both USL and USL conserve energy (so does the USF of Bardenhagen [2002]). However, for mode 10, USL is dissipative whereas MUSL conserves energy much better.

6.1.2 Impact of two elastic bodies

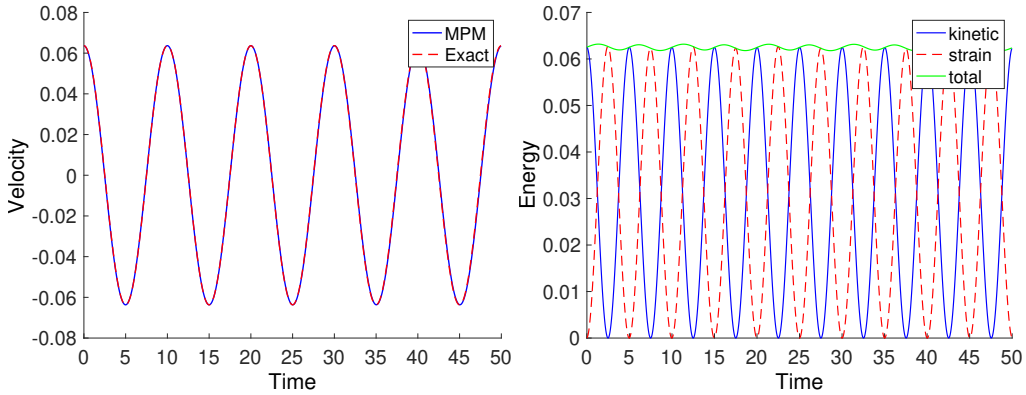
The problem of collision of two elastic bodies, first presented in Sulsky et al. [1994], is perhaps the easiest 2D problem to check the implementation of a 2D MPM code. As shown in Fig. 60, two identical elastic disks, moving in opposite directions towards each other, will collide each other and rebound. The computational domain is a square, of which side is one millimeter. There is no boundary conditions in this problem since the simulation stops before the particles move out of the computational box after impact. A plane strain condition is assumed.

The computational domain is discretized into 20×20 square elements. Four particles are used per elements resulting in a total of 416 particles. Simulations are performed in the absence of gravity. A constant time step $\Delta t = 0.001$ s was used. Even though the USL can be used with a cut-off value for small nodal mass, we adopted the MUSL. Initial condition for this problem is the initial velocities of the particles, $\mathbf{v}_p = \mathbf{v}$ for lower-left particles and $\mathbf{v}_p = -\mathbf{v}$ for upper-right particles. A constant time step of 0.001 s was used.

In order to check the energy conservation, the strain and kinetic energy are computed for each time step. They are defined as



(a) PIC ($\alpha = 0$) (b) PIC ($\alpha = 0$)



(c) FLIP ($\alpha = 1$) (d) FLIP ($\alpha = 1$)

Figure 58: Vibration of a bar: mode 1 case: PIC versus FLIP with USL algorithm used (16 cells, PPC=2). For this mode, USL and MUSL perform similarly.

$$E_s = \sum_{p=1}^{n_p} u_p V_p, \quad E_k = \frac{1}{2} \sum_{p=1}^{n_p} \mathbf{v}_p \cdot \mathbf{v}_p m_p \tag{6.5}$$

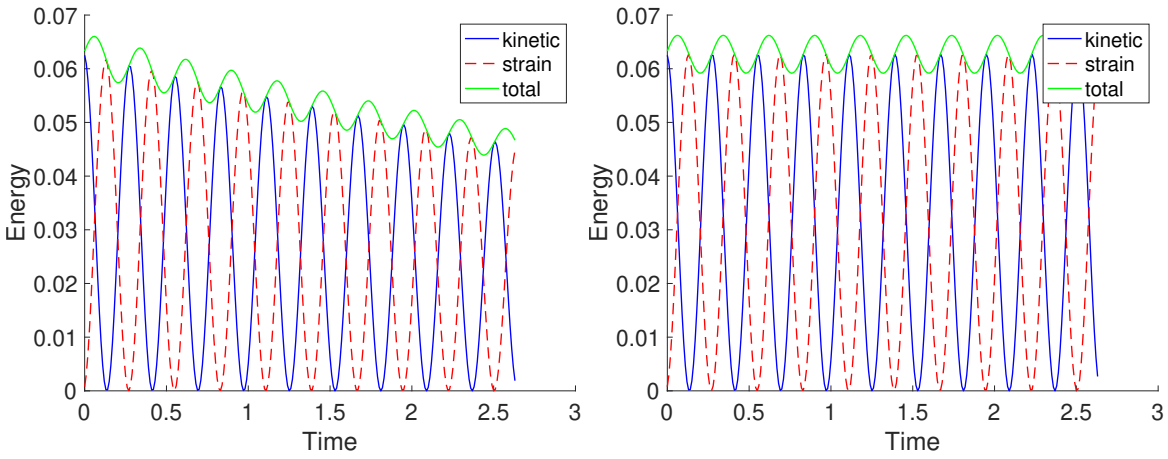
where u_p denotes the strain energy density of particle p, $u_p = 1/2\sigma_{p,ij}\epsilon_{p,ij}$.

The movement of two disks is given in Fig. 61. The collision occurs in a physically realistic fashion, although no contact law has been specified. Fig. 62 plots the evolution of the kinetic, strain and total energy. All of the initial energy is kinetic energy. The initial kinetic energy is $K = 2 \times (0.5 \times (v^2 + v^2) \times \rho \times \pi \times r^2) = 2.513$. The kinetic energy decreases during impact and is then mostly recovered after separation. The strain energy reaches its maximum value at the point of maximum deformation during impact and then decreases to a value associated with free vibration of the disk. The result is identical to the ones reported in Sulsky et al. [1994], Buzzi et al. [2008], Coetzee [2003] which confirms the implementation. However, the contact did occur earlier than it should have been. The correct contact time is $t = AB/(2\sqrt{2}v) = 1.5858$ s where $v = 0.1$. In the simulation contact happened at $t = 1.3$ s. This result is expected as the contact is resolved at the grid nodes not the particles i.e., contact is detected even when the particles of the two bodies are one cell separate. The situation is more severe if a C^{p-1} smooth basis function is used as in GIMP or B-splines MPM where the nodal support is larger. A simple mesh refinement can improve the result or a contact algorithm which is based on the particle distance should be used.

6.1.3 High-velocity impact

To test the code for elasto-plastic solids, we present a 2D simulation involving an AISI 52-100 chromium steel disk impacting an elastic-perfectly plastic target of 6061-T6 aluminum under plane strain condition. The geometry and initial conditions are given in Fig. 63. The disk is assumed to be elastic and the aluminum target to be small-strain elastic-perfect plastic and obeying a von Mises yield criterion. The stress update for this plastic material can be found e.g., in Simo and Hughes [1998].

The background grid has a 50×50 layout, and is not shown in the figures to avoid clutter. The simulation consists of a total of 6908 particles, 208 of which constitute the steel disk and 6700 particles for the aluminum target. The simulation was performed over constant time increments of $\Delta t = 10^{-8}$ s up to $t = 40.0 \times 10^{-6}$ s. No special treatment for the contact



(a) FLIP, USL (b) FLIP, MUSL

Figure 59: Vibration of a bar: mode 10 case: USL versus MUSL (50 cells, PPC=1).

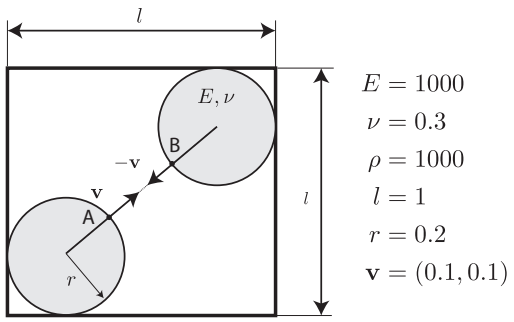
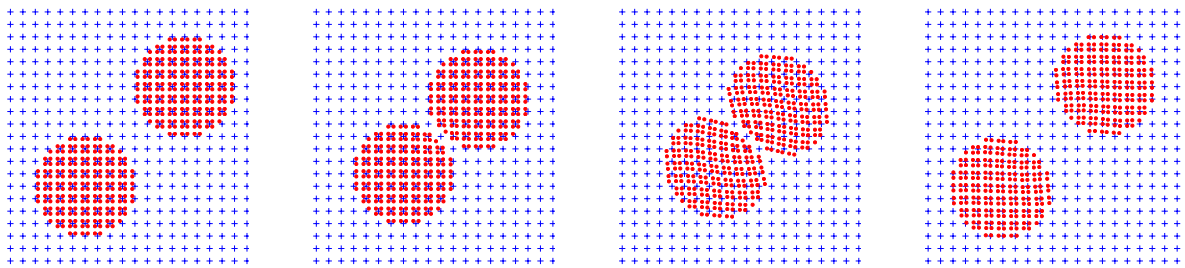


Figure 60: Impact of two elastic bodies: problem statement. The computational domain is a unit square and the radius of the disks is 0.2 mm. Any set of consistent units is sufficient.



(a) $t = 1.0$ s (b) $t = 1.5$ s (c) $t = 2.0$ s (d) $t = 3.0$ s

Figure 61: Impact of two elastic bodies: simulation snapshots. These images were created using the `PyPlot` graphical package [Johnson, 2012].

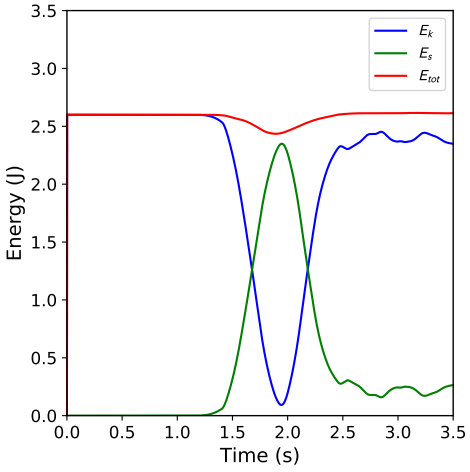


Figure 62: Impact of two elastic bodies: evolution of kinetic, strain and total energies.

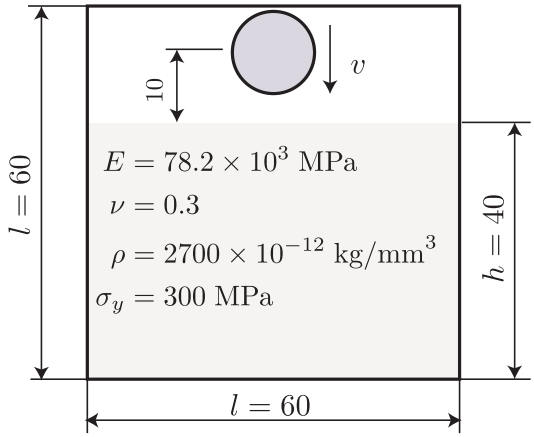


Figure 63: Setup and initial conditions for the high-velocity impact problem. The disk has an initial velocity of 1160 m/s and the disk's diameter is 9.53 mm. The boundary represents the computational domain. Length are in mm.

between the disk and the target was used. In other words, the inherent automatic no-slip, no-penetration contact of MPM is exploited. Fig. 64 shows the penetration of the disk into the target at four different time states. The results are very similar to the ones reported by Sulsky et al. [1995], Coetzee [2003], which serves as verification for the codes implemented in this study.

6.1.4 Sod's shock tube

Sod's problem is a test case commonly used in computational hydrodynamics to see how well a certain computational approach works [Sod, 1978]. This problem, shown in Fig. 65, consists of a shock tube where a diaphragm is located in the middle of the tube. Two sides of the diaphragm have different pressures and densities, which make the fluid flows when the diaphragm is broken. The left side of density is 1 and pressure is 1. The right side of density is 0.125 and pressure is 0.1, and both sides have a zero initial velocity. Any set of consistent units suffice. At time $t = 0$, the diaphragm is removed. The fluid is modeled as an ideal gas with $\gamma = 1.4$ (cf. Section 5.4).

Only a small modification was made to the one dimensional MPM code developed for solid mechanics: the internal energy e is stored for every fluid particles. The initial value for e is computed using an EOS and the initial density and pressure. The results, obtained with the standard MPM (MUSL update), at time $t = 0.143$, when the shock traveled a distance of about 0.25, with 300 cells and three particles per cell are given in Figs. 66 and 67 without and with artificial viscosity. Note that the data are plotted at the material points. In Fig. 68 we also plot the data at the grid nodes. To compute the grid nodal pressure, the particle stress is mapped to the grid nodes in the same manner as the velocity mapping. As can be seen a smoother distribution was obtained by this technique as earlier mentioned in the works of e.g., [Andersen and Andersen, 2010a].

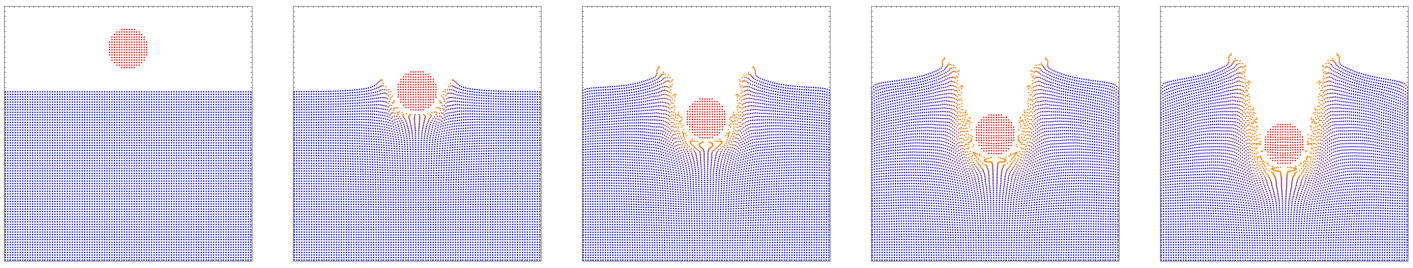


Figure 64: Snapshots for the high-velocity impact problem as the steel disk penetrated into the aluminum target. The color represents the equivalent plastic strain. These images were created using the `PyPlot` graphical package.

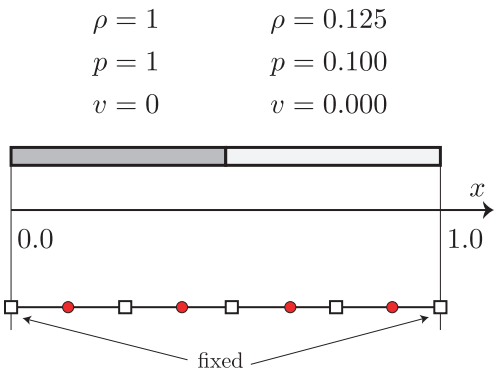


Figure 65: Sod's problem: initial configuration.

6.2 Convergence tests

As MPM are typically used for nonlinear problems of which solutions do not exist, the method of manufactured solutions has been used to check the convergence of MPM [Wallstedt and Guilkey, 2008, Kamojjala et al., 2015]. This section presents two common convergence tests: axis-aligned unit cubes in Sections 6.2.1 and 6.2.2 and the generalized vortex problem in Section 6.2.3. The first test involves only compression/tension deformation and the second one deals with shear deformation. Both tests concern only zero-traction boundary conditions as it is notoriously difficult to enforce non-zero Neumann boundary conditions in particle methods. As it is always easier to code and debug 1D problems, we first present a 1D test in Section 6.2.1 and a 2D/3D test in Section 6.2.2. Furthermore, 1D problems allow to use extremely fine meshes which reveal instability of some MPM variants. We refer to Appendix A for details on the MMS method and the derivation of various results given in this section.

6.2.1 Axis-aligned unit segment

Let's consider a unit segment i.e., the spatial domain is $0 \leq X \leq 1$. The manufactured displacement field is assumed to be

$$u(X,t) = G\sin(\pi X)\sin(c\pi t) \tag{6.6}$$

where G is the maximum amplitude of the displacement; $c = \sqrt{E/\rho_0}$ and E denotes the Young's modulus. The period is thus given by $T = \frac{2\pi}{c\pi}$; X denotes the material coordinates i.e., coordinates in the reference configuration. and the time domain is $0 \leq t \leq T$ i.e., one period of oscillation is considered. The corresponding body force and boundary/initial conditions for this manufactured solution are given in Appendix A.1.

This MMS verification test is done using the following material parameters: $E = 10^7$ Pa, $\nu = 0.3$, and $\rho_0 = 1000\,\text{kg/m}^3$. Both small and large displacements are tested by setting the maximum amplitude displacement $G = 10^{-4}$ and $G = 0.05$, respectively. The same time step of $0.2h/c$, where h is the cell size, was used. We believe that this small time step eliminates any error due to time discretization. And note that herein we focus on spatial convergence only. We study the convergence of the FEM (see Appendix F for the formulation), MPM, TLMPM, GIMP and iMPM. Note that CPDI performs identically to cpGIMP for this problem and thus not discussed. To this end, grids of size $h = 2^{-k}$, $k = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12\}$, with the smallest cell size is .000244141, are considered. In the literature, for convergence tests, rather coarse meshes have been used with which convergence was obtained, as Gong [2015] pointed out that when the mesh is very fine, many MPM variants

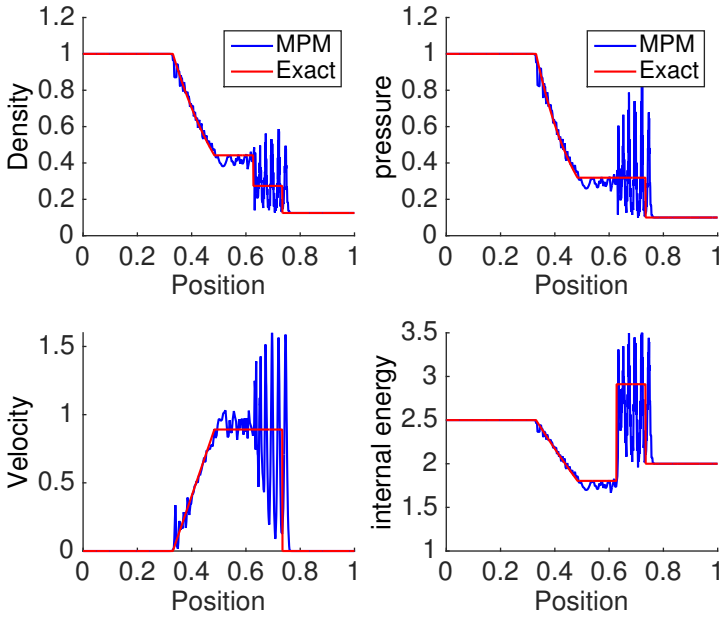


Figure 66: Sod's problem: 200 elements with 3 particle per element. No artificial viscosity.

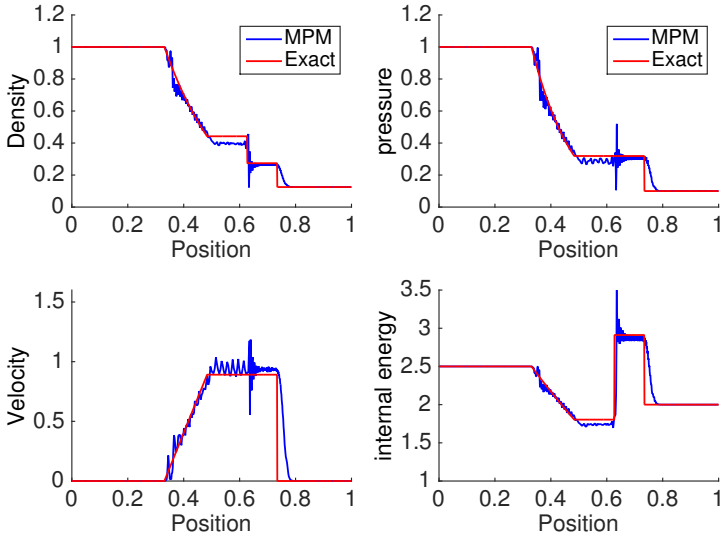


Figure 67: Sod's problem: 200 elements with 3 particle per element. With artificial viscosity.

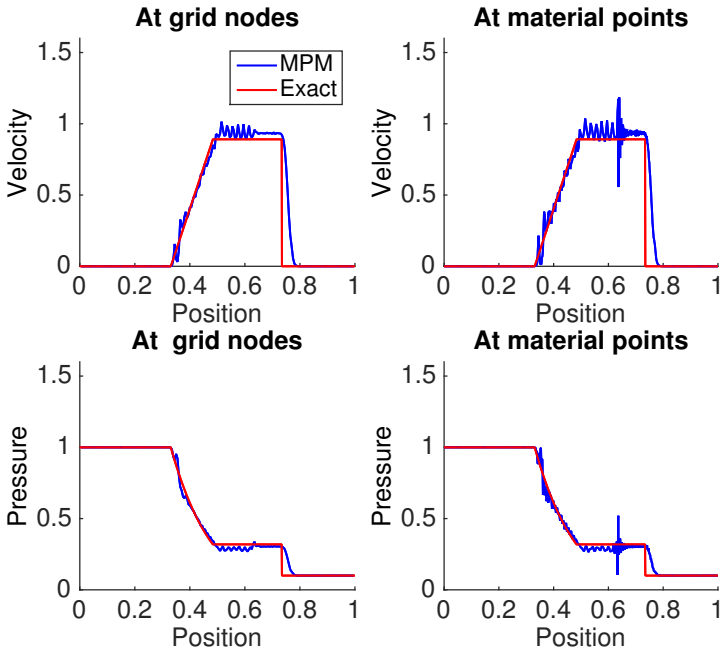


Figure 68: Sod's problem: 200 elements with 3 particle per element with artificial viscosity. Data plotted at material points and at grid nodes.

exhibit divergence. That is why, herein we are utilizing very fine meshes to have a fair view of the convergence behavior of MPMs.

The results are given in Fig. 69 where it can be seen that optimal convergence (of order 2) was obtained with the FEM and TLMPM. This is expected as there is no error in updating the stress for this manufactured solution (the constitutive model is a hyperelasticity). In the TLMPM, there is no quadrature error (for this axis-aligned problem) and cell-crossing instability. The TLMPM accuracy is certainly slightly lower than the FEM accuracy. The standard MPM does converge for the small deformation case up to a certain mesh level and the error plateaus. However, it does not converge at all for the large deformation case. The convergence is better with uGIMP even though the error increases for fine meshes (large deformation case). The cpGIMP does perform better when large deformation occurs but the convergence rate is far from optimal. The convergence of the iMPM is optimal for meshes with size smaller than 10^{-3} and all of a sudden, it diverges for more refined meshes. The reason of that is unclear at this stage [Gong, 2015].

While performing this convergence test, we have realized that the convergence does depend on the error measure. To demonstrate this, considering the following root mean square error

$$e^{\text{RMSE}} = \sqrt{\frac{\sum_{t^n=0}^{t_f} \sum_{p=1}^{n_p} \left|\left|\mathbf{u}_p^h(t^n) - \mathbf{u}^{\text{exact}}(\mathbf{X}_p, t^n)\right|\right|^2}{n_T \times n_p}} \tag{6.7}$$

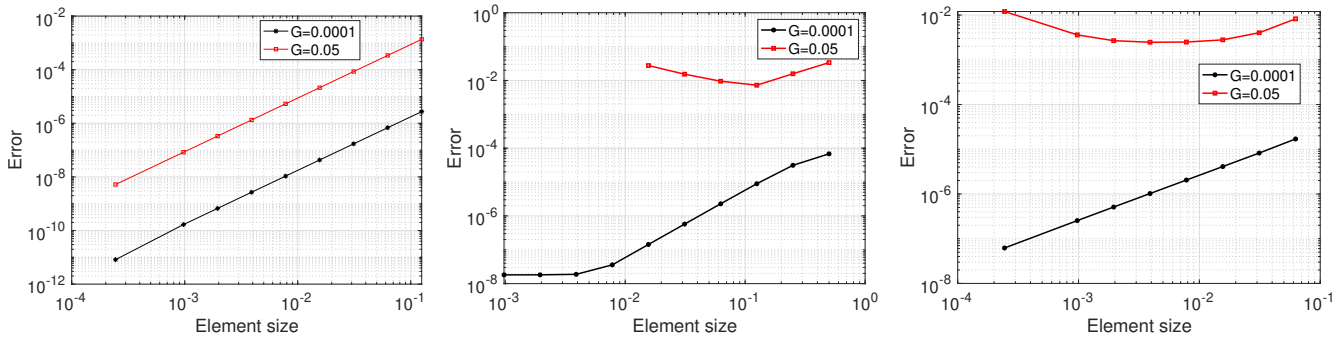
It is obvious that when the mesh gets finer, both n_p and n_T (the number of time steps) gets bigger and thus the error measure in Equation (6.7) will always give convergence, cf. Fig. 69d. We do not think this RMSE is objective for dynamics simulations because one can get good convergence simply by refining the time steps i.e., making n_T bigger.

6.2.2 Axis-aligned unit cube

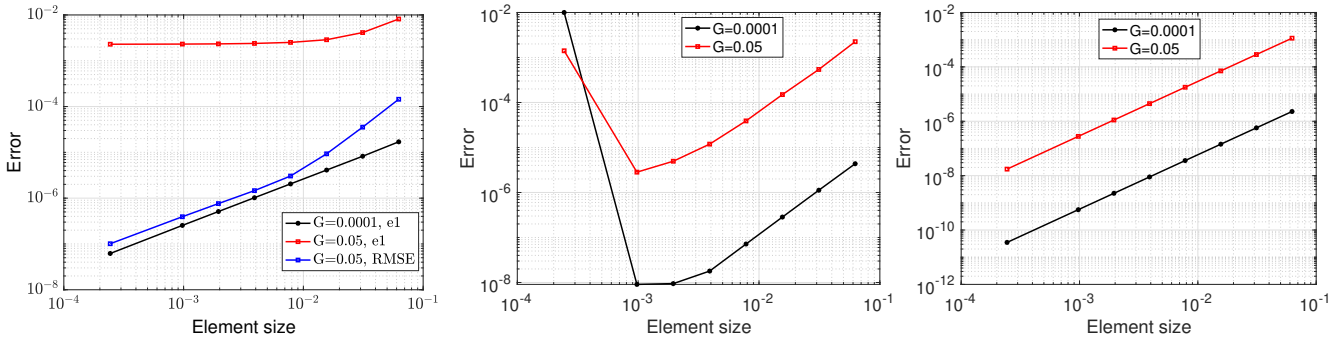
This MMS test involves only tension and compression as the displacement field is assumed to be:

$$\mathbf{u}(\mathbf{X}, t) = G \sin(\pi \mathbf{X}) \sin\left(\sqrt{\frac{E}{\rho_0}} \pi t + \boldsymbol{\phi}\right) \tag{6.8}$$

where G is the maximum amplitude of displacement, E denotes Young's modulus, and $\boldsymbol{\phi} = [0, \pi, \pi]$, an arbitrary phase angle. $\mathbf{X}$ denotes the material coordinates in the reference configuration. The solid is a unit cube occupying the domain delimited by $0 \leq X_1 \leq 1$, $0 \leq X_2 \leq 1$, and $0 \leq X_3 \leq 1$. One period of oscillation is considered, *i.e.* the time domain is $0 \leq t \leq T$, with $T = \frac{2\pi}{\pi\sqrt{E/\rho_0}}$. From the prescribed displacements, and using the compressible Neo-Hookean model introduced in Appendix B.2, the necessary body forces applied on the nodes are obtained as (see de Vaucorbeil et al. [2019]



(a) TLFEM (b) MPM, MUSL (c) uGIMP, USL



(d) cpGIMP, USL (e) iMPM, USL (f) TLMPM, MUSL

Figure 69: Axis-align unit segment test: convergence of FEM and various MPM variants using the un-normalized error measure e_1 given in Equation (A.12).

for derivation details):

$$\mathbf{b}(\mathbf{X},t) = \begin{bmatrix} \frac{\pi^2 u_1}{\rho_0}\left(\frac{\lambda}{F_{11}^2}\left(1-\ln(F_{11}F_{22}F_{33})\right)+\mu\left(1+\frac{1}{F_{11}^2}\right)-E\right) \\ \frac{\pi^2 u_2}{\rho_0}\left(\frac{\lambda}{F_{22}^2}\left(1-\ln(F_{11}F_{22}F_{33})\right)+\mu\left(1+\frac{1}{F_{22}^2}\right)-E\right) \\ \frac{\pi^2 u_3}{\rho_0}\left(\frac{\lambda}{F_{33}^2}\left(1-\ln(F_{11}F_{22}F_{33})\right)+\mu\left(1+\frac{1}{F_{33}^2}\right)-E\right) \end{bmatrix}. \tag{6.9}$$

We also impose the following boundary conditions, obtained using Equation (6.8), onto the nodes of the background mesh:

$$\mathbf{v}(X_i = 0, t) = \mathbf{v}(X_i = 1, t) = 0, \quad \text{for } i = 1,2,3 \tag{6.10}$$

as well as the following initial conditions:

$$\mathbf{v}(\mathbf{X},0) = \pi\sqrt{\frac{E}{\rho_0}}G\sin(\pi\mathbf{X})\cos(\boldsymbol{\phi}) \tag{6.11}$$

$$\mathbf{P}(\mathbf{X},0) = \mathbf{0} \tag{6.12}$$

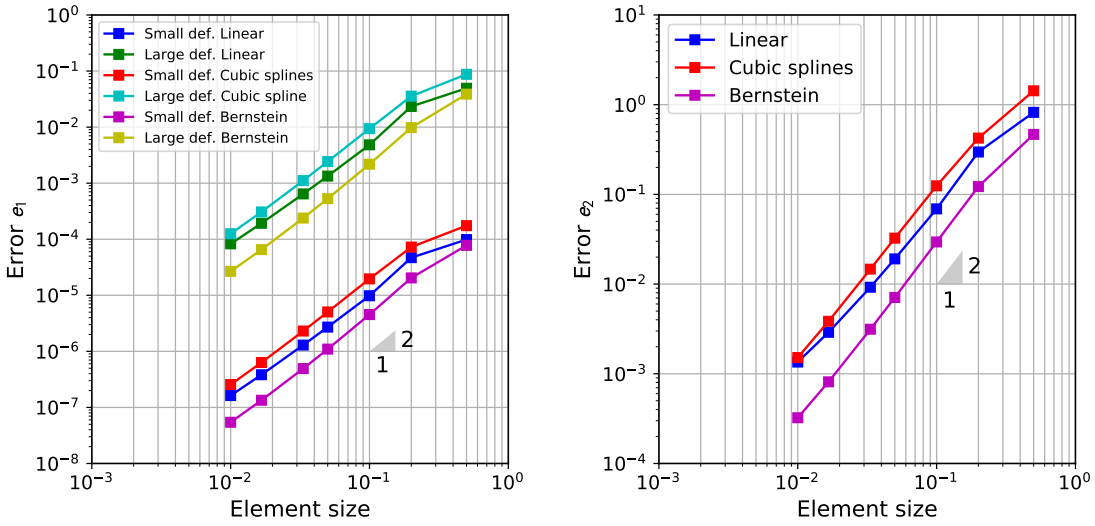
As can be seen, this test is suitable for testing a 2D/3D MPM code without having to deal with Neumann boundary conditions (as the traction is zero at the boundary).

This MMS verification test is performed using the following material parameters: $E = 10^7$ Pa, $\nu = 0.3$, and $\rho_0 = 1000$ kg/m^3. Both small and large displacements are tested by setting the maximum amplitude displacement $G = 10^{-4}$ and $G = 0.05$, respectively. In both cases, the test is done using linear, cubic splines, and Bernstein shape function. The same time step of $0.2h/c$, where h is the background grid element size, was used. We believe that this small time step eliminates any error due to time discretization. And note that herein we focus on spatial convergence only.

For all the tested cases, the error as a function of the background grid cell size made by the TLMPM using a combination of PIC and FLIP with $\beta = 0.99$ for the velocity update is plotted in Fig. 70. It can be seen that contrary to the standard MPM which does not converge (see Fig. 69b), the TLMPM converges for all the different shape functions tested. In particular, its

convergence is quadratic when using either cubic splines or Bernstein shape functions and quasi quadratic when using linear shape functions. The trends are similar for either error measure e_1 or e_2 used.

Remark 38 *Note that as e_1 is not normalized and is therefore proportional to G, the absolute error is higher for large than for small deformations, while as e_2 is normalized, and no difference was seen between these two cases, Fig. 70b does not distinguish between large and small deformations.*



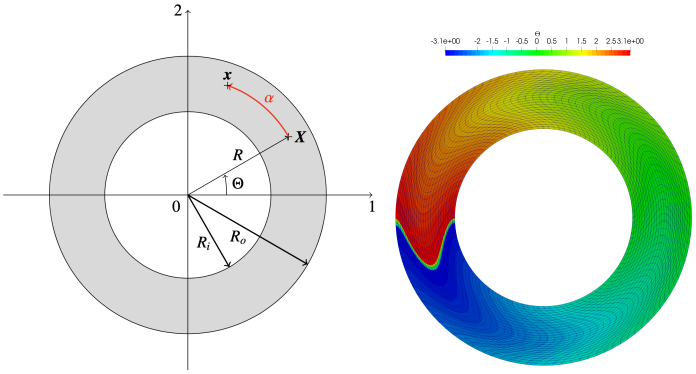
(a) Non-normalized error plot

(b) Normalized error plot

Figure 70: Plot of the displacement error as a function of the background mesh refinement for small and large deformation obtained using TLMPM. Note that as e_2 is normalized, and since we did not see any difference between small and large deformations, this distinction is not made in (b).

6.2.3 Generalized vortex problem

The generalized vortex problem is an example of MMS involving simple shear with superimposed rotation which is a complicated problem to simulate using MPM and its advanced (improved) variants [Brannon et al., 2011, Kamojjala et al., 2015, Wang et al., 2019]. This problem features a 2D ring, cf. Fig. 71a, which is locally subjected to a pure circular motion. The local displacement is purely angular and varies with the radial coordinate. Therefore, the material is only subjected to shear and circular motion, see Fig. 71b.



(a) (b)

Figure 71: Generalized vortex problem. Geometry and variable definition (a), and deformed shape (b).

The ring center is at the origin ($x = y = 0$) and its inner and outer radii are R_i and R_o, respectively. The current position **x** of any given point on the ring is thus given as:

$$\mathbf{x}(t) = \mathbf{Q}(t) \cdot \mathbf{X} \tag{6.13}$$

with $\mathbf{Q}$ being a standard rotation matrix in 2D:

$$\mathbf{Q}(t) = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \tag{6.14}$$

where α is the rotation angle which varies only with time and radial coordinate R ($R = \sqrt{X^2 + Y^2}$). It is given as:

$$\alpha(t, R) = g(t)h(R) \tag{6.15}$$

where $g(t)$ controls the amplitude of the deformation with time and $h(R)$ controls the relative radial variation of the rotation; $g(t)$ is taken as periodic, and $h(R)$ is chosen such that the outer and inner radii do not move: $h(R_i) = h(R_o) = 0$. Zero traction on the boundaries is insured by having: $h'(R_i) = h'(R_o) = 0$ as well. Following Brannon et al. [2011], they are given by:

$$h(R) = 1 - 8\left(\frac{R - \bar{R}}{R_i - R_0}\right)^2 + 16\left(\frac{R - \bar{R}}{R_i - R_0}\right)^4, \quad g(t) = G\sin\left(\frac{\pi t}{T_0}\right) \tag{6.16}$$

with $\bar{R} = (R_0 + R_i)/2$ and T_0 is the period.

Using the procedure shown in Appendix A.1 the body forces necessary to obtain this vortex motion are expressed as (see de Vaucorbeil et al. [2019] for detailed derivation):

$$\begin{aligned} b_1(R,t) &= b_R(R,t)\cos\Theta - b_\Theta(R,t)\sin\Theta \\ b_2(R,t) &= b_\Theta(R,t)\cos\Theta + b_R(R,t)\sin\Theta \end{aligned} \tag{6.17}$$

where:

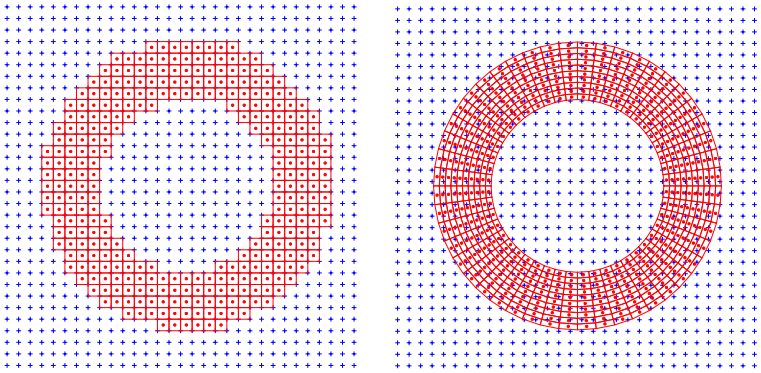
$$\begin{aligned} b_R(R,t) = &\left[\frac{\mu}{\rho_0}\left(3g(t)h'(R) + Rg(t)h''(R)\right) - Rg''(t)h(R)\right]\sin\alpha \\ &+ \left[\frac{\mu}{\rho_0}R(g(t)h'(R))^2 - R(g'(t)h(R))^2\right]\cos\alpha \\ b_\Theta(R,t) = &\left[-\frac{\mu}{\rho_0}\left(3g(t)h'(R) + Rg(t)h''(R)\right) + Rg''(t)h(R)\right]\cos\alpha \\ &+ \left[\frac{\mu}{\rho_0}R(g(t)h'(R))^2 + R(g'(t)h(R))^2\right]\sin\alpha \end{aligned} \tag{6.18}$$

where $h'(R)$ denotes the first derivative of h with respect to R and $h''(R)$ is the second derivative. Furthermore, the initial velocity and stress are identically zero.

Similarly to Kamojjala et al. [2015], the inner and outer radii are taken as 0.75 m and 1.25 m, respectively. Material parameters are taken as: $E = 10^3$ Pa, $\rho_0 = 1000$ kg/m^3, $\nu = 0.3$, and the maximum displacement amplitude $G = 1$; $T_0 = 1$ s and the total simulation time is one second i.e., one period is considered. The particles' velocities is updated using a mix of PIC and FLIP with the mixing factor $\alpha = 0.99$. Moreover, the velocity of all the nodes outside of the ring is set to zero, and the body forces (Equation (6.17)) are applied directly on the nodes. We solve this problem using three MPM variants: the TLMPM (with linear and quadratic Bernstein functions), the boundary non-conforming CPDI-Q4 and the boundary conforming CPDI-Q4. In the boundary non-conforming CPDI, the particles are generated using the same algorithm for the standard MPM and thus the particle domains do not fit the boundary, see Fig. 72a. The TLMPM has the same particle positions. On the other hand, in the conforming CPDI, the particle domains are generated using a mesh generator (we used Gmsh [Geuzaine and Remacle, 2009]). The particle domains are conforming to the boundary, see Fig. 72b.

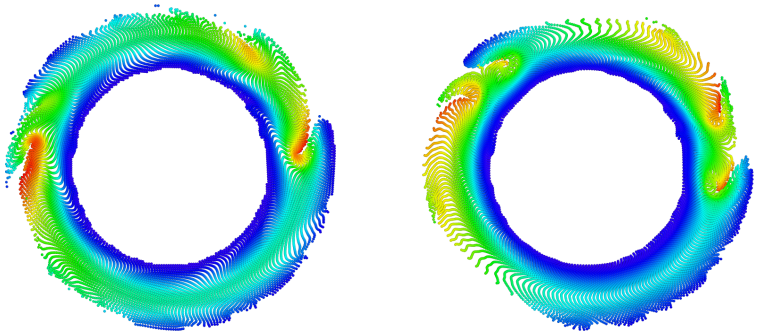
The results obtained with CPDI-Q4 are given in Fig. 73. The performance of CPDI-Q4 is much better than the standard MPM and uGIMP (not shown here), but the deformation is not well captured: the boundaries of the domain are not smooth and circular. The particle domains are so much distorted. For this reason, we do not perform mesh convergence for CPDI. The remaining of this section focuses on the TLMPM.

Figure 74 shows how the displacement errors e_1 and e_2 change according to the element size. It can be seen that at large element size, the convergence rate is quadratic, but decreases progressively as the cell size decreases. This is the sign of a competition between two errors: cell size related errors and mapping errors. As the cell size decreases, so does the error associated to it, while the mapping error which appeared negligible for large cell sizes becomes dominant. Thus the error



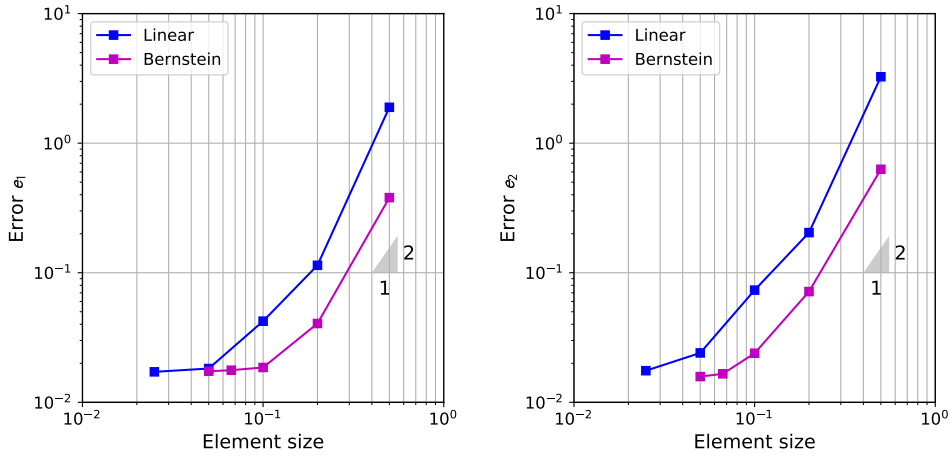
(a) Non-conforming CPDI-Q4 (b) Conforming CPDI-Q4

Figure 72: Generalized vortex problem. Grid and initial particles used in CPDI. These images were created using the `PyPlot` graphical package [Johnson, 2012].



(a) Non-conforming (b) Conforming

Figure 73: Deformed configuration obtained with CPDI-Q4. The color represents the magnitude of the displacement field. Blue is zero and read is maximum. The grid cell size is 0.05 and there are about 14 000 particles. Images obtained using `Ovito` [Stukowski, 2009]

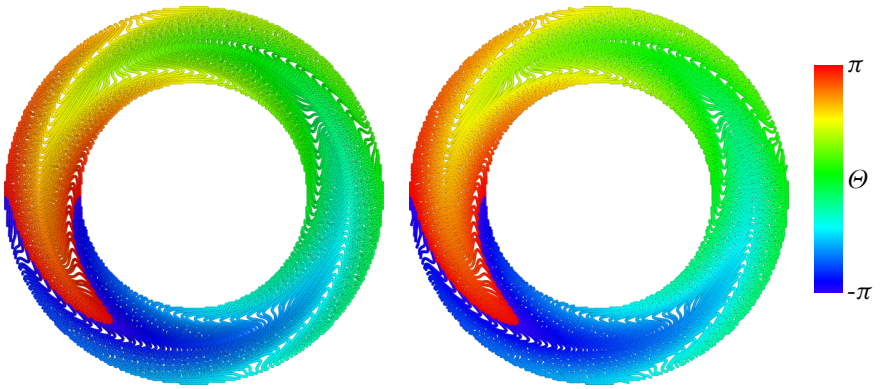


(a) Non-normalized error plot (b) Normalized error plot

Figure 74: Plot of the displacement error as a function of the background mesh refinement obtained using TLMPM. Note that as e_2 is normalized.

plateaus. Note that there is another source of error – the mapping of particle momenta to the grid as shown in Sulsky and Gong [2016]. We did not implement those improvements to mitigate this error and leave it as a future work. However, the

order of magnitude of this plateaus is so low that very good qualitative agreement exist between the deformed configurations at peak rotation angle as obtained with TLMPM using Bernstein shape functions and the analytical solution (see Fig. 75).



(a) analytical solution (b) TLMPM solution

Figure 75: Deformed configuration at $t = 0.5\ s$ obtained (a) analytically and (b) with TLMPM using quadratic Bernstein polynomials shape functions and a cell size of 0.033.

6.3 Experimentally validated simulations

This section presents simulations that involve large deformation, contacts and fracture which are signature application of MPMs (or any meshfree methods). Furthermore, these simulations are practical engineering problems of which experiments have been conducted. Concretely, the following problems are presented

- Taylor anvil test (Section 6.3.1);
- Necking and fracture of a tensile specimen (Section 6.3.2);
- Lateral compression of thin-walled structures (Section 6.3.3);
- Lateral compression of thin-walled structures (Section 6.3.4);

All results were obtained with the MUSL formulation.

6.3.1 Taylor anvil test

The Taylor anvil test is a well suited problem to evaluate the performance of elasto-plastic constitutive models and numerical codes when large plastic deformations occur [Wilkins and Guinan, 1973, Johnson and Holmquist, 1988, Predebon et al., 1991]. This test involves a OFHC Copper cylinder of original length $L_0 = 25.4$ mm, and original diameter $D_0 = 7.6$ mm, hitting a stationary rigid wall at a high velocity $v_0 = 190$ m/s. It was used by Johnson and Holmquist [1988] to compare various constitutive models for both OFHC Copper and Armco Iron. In their work, they performed experiments to determine the material parameters for these two materials using different constitutive models. They also performed numerical simulations to compare the models. Herein, we are interested in how the MPM prediction of the deformed bar in terms of the final diameter, bulge and length (see Fig. 76 for an illustration of these quantities) compares with the experiments. This problem is now solved using the standard MPM and the TLMPM. In the literature, this test has been studied using the axi-symmetric standard MPM [Sulsky and Schreyer, 1996] and recently by Liang et al. [2019].

This material is modeled using the elasto-plastic constitutive model presented in Appendix B.3, but without damage and temperature effects. The material parameters used are taken from Sulky's work and are listed in Table 5. The performance of the MPM is accessed using the error measure introduced by Johnson and Holmquist [1988] and defined as:

$$\bar{\Delta} = \frac{1}{3}\left[\frac{|\Delta L|}{L_T} + \frac{|\Delta D|}{D_T} + \frac{|\Delta W|}{W_T}\right] \tag{6.19}$$

where L_T, D_T and W_T are the length, diameter and bulge measured experimentally and $\Delta L = L_f - L_T$, $\Delta D = D_f - D_T$ and $\Delta W = W_f - W_T$. The MPM length, diameter and bulge are denoted by L_f, D_f and W_f, respectively. All these lengths

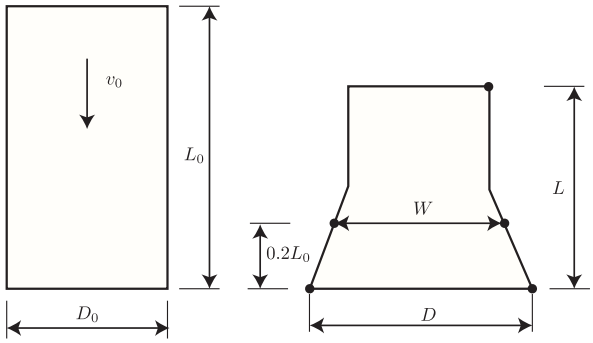


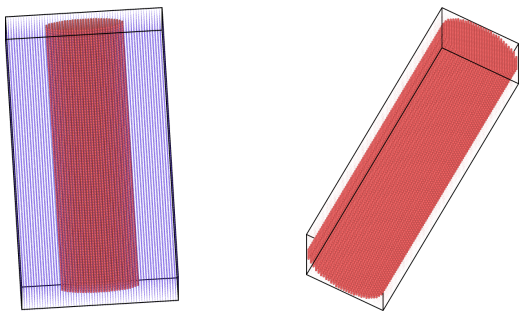
Figure 76: Taylor bar impact for an elasto-plastic OFHC copper cylinder given an initial downward velocity of 190 m/s.

are measured based on the distances between particles marked in Fig. 76. As it is impossible to have particles at exact locations defining the bulge W, our prediction for W is worse than the length and the diameter. Locating these particles and their distances were done manually using tools provided in `Ovito` [Stukowski, 2009].

Material parameters		Parameters for Johnson-Cook model		Parameters for EOS	
Density	8940 kg/m^3	A	65 MPa	c_0	3933 m/s
Young's modulus	115 GPa	B	356 MPa	S_α	1.5
Poisson's ratio	0.31	C	0.013	Γ_0	0
		n	0.37		

Table 5: Material parameters for the OFHC Copper material used in the simulation of the Taylor bar impact test.

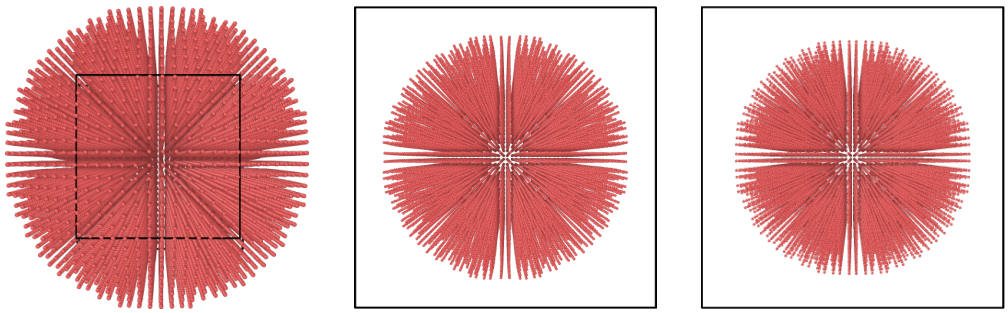
Figure 77 shows the distribution of the material points and the grid used in the ULMPM and TLMPM. As can be seen, the TLMPM requires a grid just encompassing the solid in its initial undeformed configuration, while for the ULMPM the grid must cover a larger space that will cover the entire deformation space of the solid. For the TLMPM (linear shape functions), the cell size is $h = 0.25$ mm with 1 material point per element for a total of 74 052 points. For the ULMPM (or just MPM) we use the hat functions and the cubic B-spline functions (BSMPM). Eight particles per cell are used for both cases. The presence of the wall is simulated by zeroing v_{xI} of the bottom nodes (where the wall is located).



(a) ULMPM (b) TLMPM

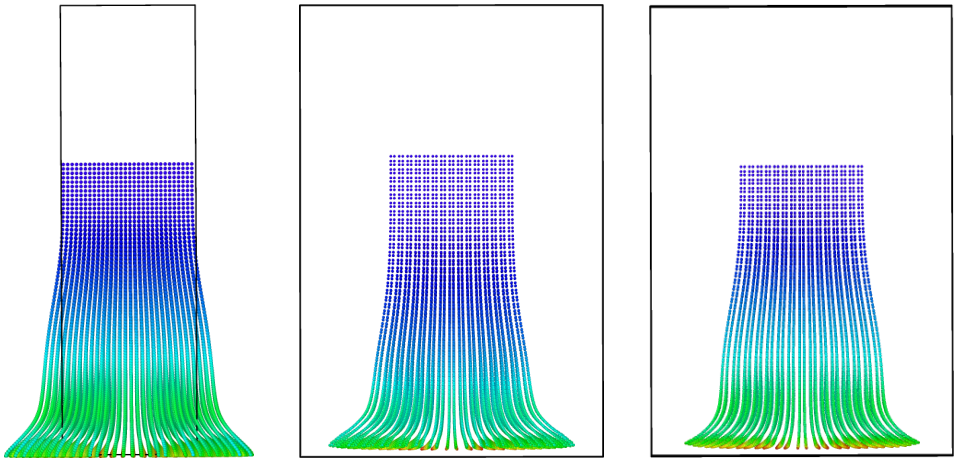
Figure 77: Taylor anvil test: initial particle distribution and grid of ULMPM and TLMPM. Note that the cylinder axis is in the x direction and for the TLMPM there is no gap between the cylinder and the wall.

Figure 78 presents the geometry (looking from the bottom) of the deformed cylinder at the end of the simulation i.e., $t \approx 63$ μs when the kinetic energy is close to zero. All MPM variants perform very well and much better than LS-DYNA SPH given in Ma et al. [2009a]. Fig. 79 presents the deformed configuration of the bar. The predicted values by TLMPM and ULMPM (MPM and BSMPM) for the final diameter, bulge and length are given in Table 6. Also presented is the results from Sulsky and Schreyer [1996], just for reference.



(a) TLMPM (b) MPM (c) BSMPM

Figure 78: Final configurations of the Taylor bar (bottom view). The black squares denote the background grids.



(a) TLMPM (b) MPM (c) BSMPM

Figure 79: Taylor anvil test. Deformed configurations obtained with different MPM variants. The color presents the equivalent plastic strain. Visualization done with `Ovito` [Stukowski, 2009].

	Experiment	TLMPM	MPM	BSMPM	MPM (Sulsky)
Diameter [mm]	13.5	13.9	14.4	13.9	14.6
Bulge [mm]	10.1	9.4	8.9	9.5	9.12
Length [mm]	16.2	16.2	17.2	16.5	18.3
$\bar{\Delta}$ [-]	n/a	0.03	0.08	0.04	0.1

Table 6: Results of the Taylor anvil test.

Remark 39 *It was this Taylor anvil test that led Ma et al. [2009a] to observe that explicit MPM is faster than LS-DYNA ULFEM, but with a slightly lower accuracy. In our humble opinion, having a faster method would not make industry people to choose it. This is simple as people are so familiar to the FEM and there are excellent FEM packages with user friendly user interfaces for both pre and post processing.*

6.3.2 Tensile test specimen experiencing necking and damage

As a simplest demonstration for problems exhibiting large plastic deformations and eventually fracture, we present 3D simulations of smooth cylindrical tensile samples made of ductile Weldox steels all the way to failure. The MPM results

will be compared with experimental results given in Dey et al. [2004, 2006] and also with the FEM (using Abaqus Explicit). The ability to handle very large deformation and fracture is important for a large range of important engineering problems such as wear, material penetration etc. As we have previously demonstrated, the TLMPM is the most efficient and accurate MPM to date, we thus use it for these simulations. As will be seen, the TLMPM is stable when material instabilities occur as experienced during necking, and it can simulate damage and fracture of ductile materials without requiring any algorithm changes, contrary to Total-Lagrangian Smoothed Particle Hydrodynamics [de Vaucorbeil and Hutchinson, 2019].

The tensile specimens are 3D smooth cylinders of 30 mm in length and 6 mm in diameter made of three different Weldox steel alloys: W460E, W700E, and W900E. These materials were selected because material parameters for the widely used Johnson-Cook constitutive model (see Appendix B.3) are available [Dey et al., 2004]. For these three alloys, their material parameters are directly taken from the literature and are listed in Tables 7 and 8. All simulations are supposed to be quasi-static. Therefore, the influence of the strain rate is not taken into account which is why $C = 0$ for all.

ρ_0 [kg/m^3]	E (GPa)	ν	c_0 [m/s]	S_α	Γ_0
7750	211	0.33	5166	1.5	0

Table 7: Material parameters for Weldox steels. ρ_0 is the reference bulk density, E Young modulus, ν Poisson's ratio, $c_0 = \sqrt{E/(2(1-2\nu)\rho_0)}$ is the bulk speed of sound, Γ_0 Grüneisen Gamma in the reference state.

Material	**Yield stress**	**Strain hardening**		**Damage**			
	A (MPa)	B (MPa)	n	C	D_1	D_2	D_3	D_4
Weldox 460E	499	382	0.458	0	0.636	1.936	-2.969	-0.0140
Weldox 700E	859	329	0.579	0	0.361	4.768	-5.107	-0.0013
Weldox 900E	992	364	0.568	0	0.294	5.149	-5.583	0.0023

Table 8: Material constants for the Johnson-Cook constitutive model and damage criterion as proposed by Dey et al. [2006].

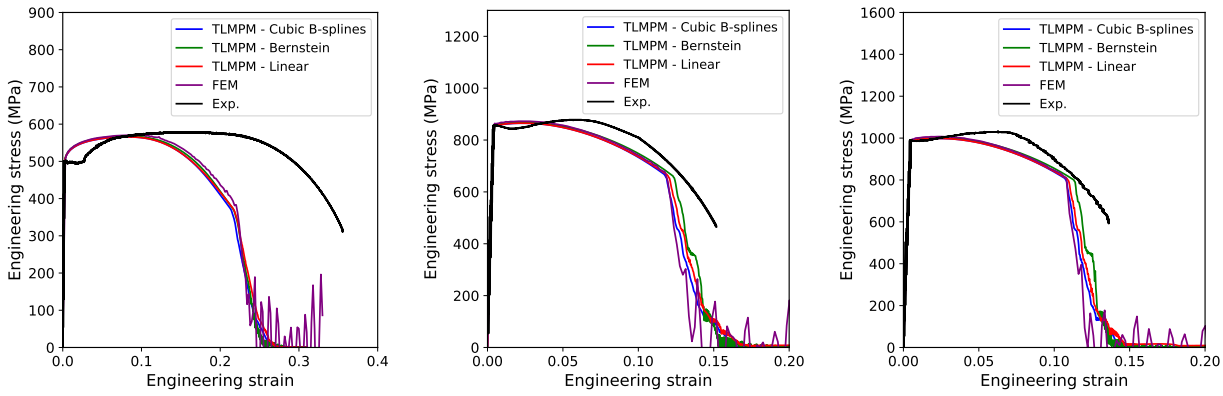
For each of the three materials, the TLMPM simulations are performed using each of the three different shape functions presented in Section 3: linear (hat functions), cubic B-splines, and quadratic Bernstein polynomials. For all these simulations, the specimens were discretized using a single particle for each cell of the background mesh, when they lie within the solid's limits. The nodes of the background mesh coincident with the top and bottom faces of the cylinders were subjected to a velocity $\pm v$ of the form $v_{max}(1-e^{-t})$ along the specimen's axis. A high velocity of $v_{max} = 1$ mm/ms was chosen in order to decrease the computational time and has no impact on the results as no strain rate effect was taken into account.

The results of these simulations are presented in Fig. 80, alongside results from FEM simulations (carried out by the authors) and experimental data published by Dey et al. [2006]. One can see that the stress-strain curves as predicted by the TLMPM is in very good agreement with the FEM. Also, as expected, after damage initiation a steady decline of stress is observed, similarly to what happens in the finite element results, but with greater stability. Finally, the numerical simulations are in reasonable agreement with the experimental data. For all that, they do not quantitatively predict the strain at failure. This was expected since all the materials parameters are directly taken from the literature and no calibration was carried out in order to obtain a better match.

Figure 81 shows the typical evolution of both the equivalent stress and damage variable inside a tensile sample. It shows formation and evolution of necking in the specimen, as well as how the damage initiates and propagates inside the sample. These results prove that the simulation of ductile materials all the way to failure is possible using the TLMPM. Note that one should be careful when using the ULMPM for modeling fracture as numerical fracture is inherent in any particle methods adopting an UL formulation [Homel et al., 2016].

6.3.3 Thin walled tube under lateral loading

To evaluate the application of the MPM for the design of energy absorption systems, Sinaie et al. [2018b] carried out a series of 3D simulations on thin-walled tubes. The setup of the simulations, i.e. geometry, material constants, loading and boundary conditions, were based on three separate sets of experiments reported in the literature. They include the quasi-static tests



(a) W460E (b) W700E (c) W900E

Figure 80: Comparison of the stress-strain curves obtained with the TLMPM, FEM and experimentally by Dey et al. Dey et al. [2006].

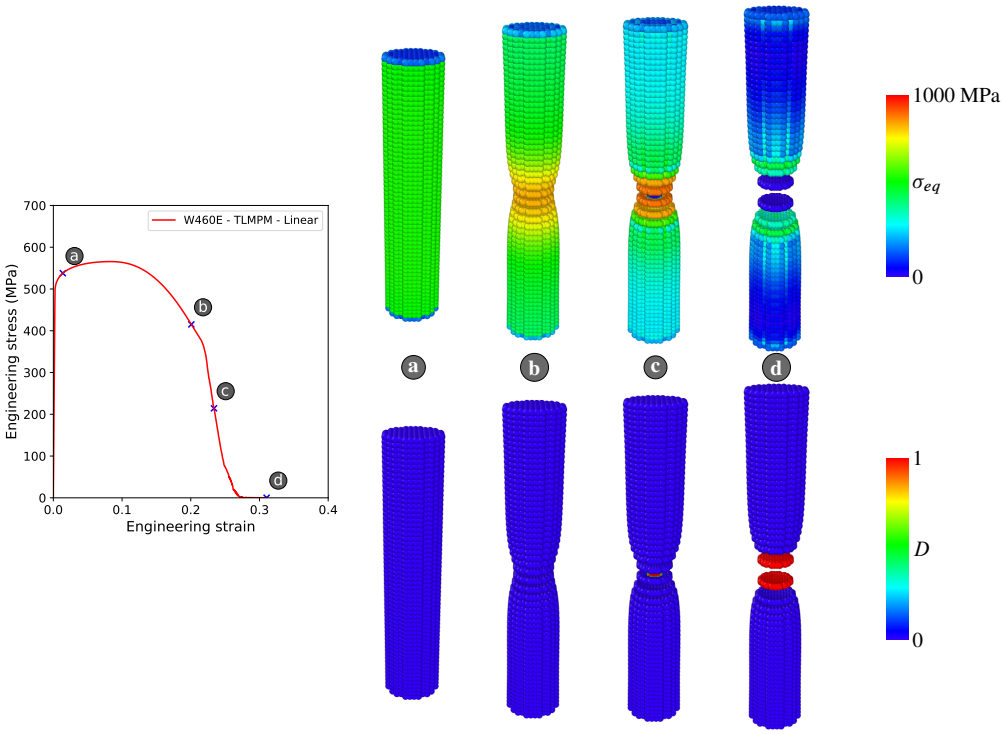


Figure 81: Details of the evolution of the equivalent von Mises stress (σ_{eq}) and the damage (D) distribution during the tensile test of a W460E smooth cylinder using linear shape functions. Visualization done with `Ovito` [Stukowski, 2009].

carried out by Xiang et al. [2017], the high velocity impact tests by Xu et al. [2015] and the wave propagation tests by Shim et al. [2007]. Schematics of the experimental setups are illustrated in Fig. 82.

For the MPM simulations, material parameters and geometry dimensions are given in Tables 9 and 10. A simple small strain J2 plasticity model with isotropic hardening was adopted. Fig. 83 shows the cross section of the tube, specifically focusing on element density in terms of the number of CPDI-Tet4 particles along the circumference and across the thickness of the tube. There is only one layer of particle along the length direction and all the z-components of the nodal quantities are set to zero. All contacts between the sample and the load platens (or walls) are no-slip. And thus, the inherent contact capability of the MPM was exploited. Furthermore, the load platens and the walls are modeled as rigid bodies using rigid particles.

Using these three sets of experiments, it was demonstrated that the MPM models successfully predict the quasi-static, dynamic and wave-propagation characteristics of thin walled tubes. Sinaie et al. [2018b] also examined the effect of grid resolution and particle count on simulation results. They proposed a set of guidelines for grid spacing and number of particles

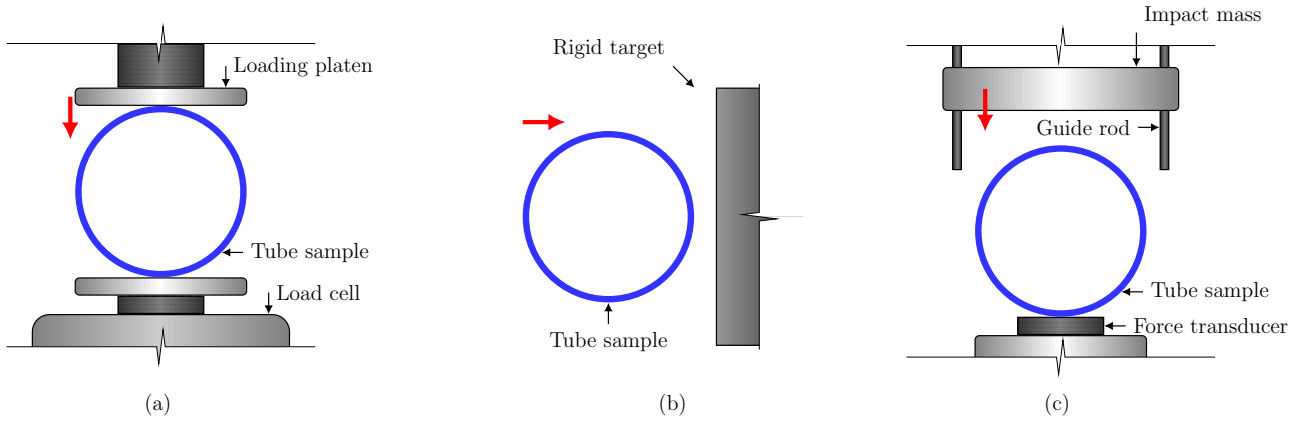


Figure 82: Schematics of the three different tests carried out in the reference papers. (a) Quasi-static compression tests by Xiang et al. [2017], (b) Impact tests by Xu et al. [2015] and (c) Wave propagation tests by Shim et al. [2007].

Variable	meaning	Experiments	
		Xiang et al. [2017]	Xu et al. [2015]
D	inner diameter	47.9 mm	25.4 mm
t	thickness	1.48 mm	0.91 mm
ρ	density	7800 kg/m^3	2760 kg/m^3
E	elastic modulus	210.0 GPa	70.0 GPa
σ_y	yield stress	310.0 MPa	290.0 MPa
b	hardening parameter	4.0	10.0
Q	hardening parameter	150.0 MPa	60.0 MPa

Table 9: Geometric and material properties of the quasi-static and impact tests.

Table 10: Geometric and material properties of the wave speed problem [Shim et al., 2007].

D	diameter	38.1 mm
t	thickness	1.0 mm
ρ	density	2693 kg/m^3
E	elastic modulus	65.9 GPa

per grid, in relation to the radius of the ring and its thickness. Some details are given below.

Quasi-static compression. Snapshots of the quasi-static simulation are shown in Fig. 84. Simulations were carried out for different levels of model refinement, specifically in terms of grid resolution and element density. Unsurprisingly, simulated results demonstrated increasingly better agreement with experimental results as the model refinement is increased. However, at a deeper level, these graphs also illustrated the underlying contribution of grid resolution and element density. This becomes especially helpful where one has to sacrifice accuracy for runtime performance.

High velocity impact. Fig. 86 shows snapshots for the high-velocity impact of thin walled tubes on a rigid wall. Results are given for different initial velocities in Fig. 87. These include the peak force, the rebound velocity and the contact duration and indicate a good agreement with measurements from the experimental work of Xu et al. [2015].

Stress wave propagation. Fig. 88 shows snapshots of the simulations on wave propagation in thin walled tubes. Through simulations, the time duration for the stress wave to travel from the excitation point at the top of the ring to the bottom of the

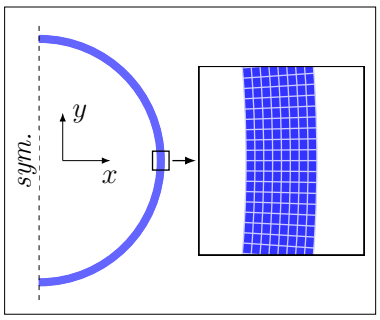


Figure 83: Illustration of the simulated tube showing divisions along the circumference (360 units) and divisions across the thickness (8 units). Each blue box in the magnified view represents a single CPDI-Tet4 division unit.

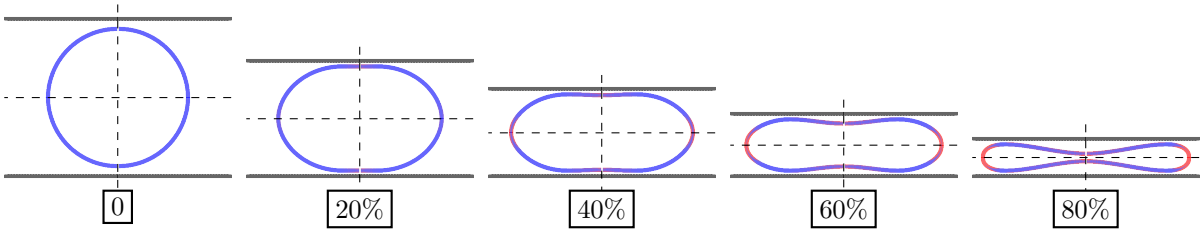


Figure 84: Snapshots of the simulations carried out on thin walled tube under quasi-static compression [Sinaie et al., 2018b].

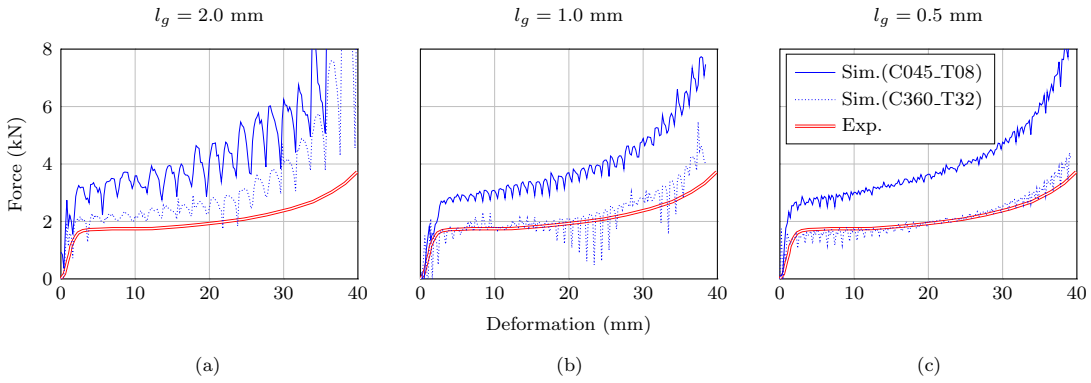


Figure 85: Simulated force-deformation curves in comparison to the experimental curve from Xiang et al. [2017]. Each figure corresponds with a value of l_g–the grid cell size. The number of CPDI elements along the circumference and across the thickness are the values following C and T, respectively.

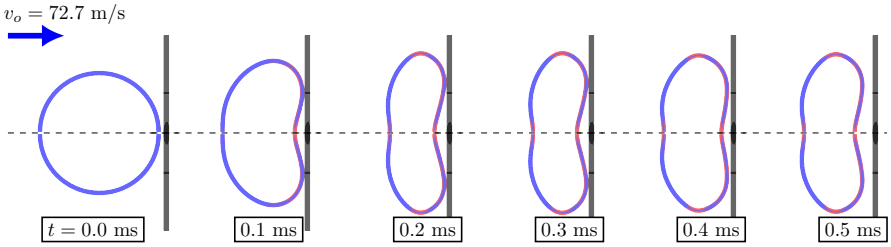


Figure 86: Snapshots of the simulations carried out for high velocity impact of thin walled tube on rigid wall [Sinaie et al., 2018b].

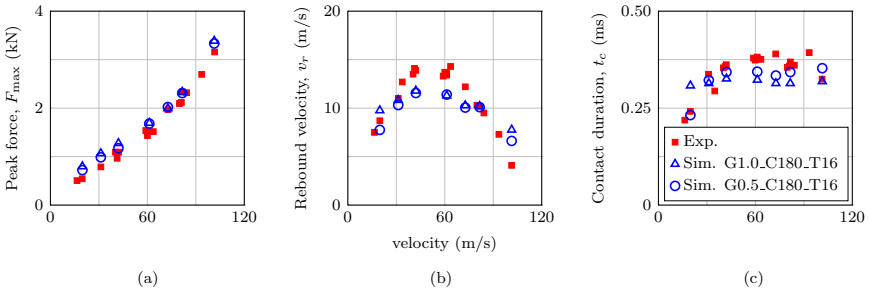


Figure 87: Comparison of simulation and experimental impact responses. Experimental data from Xu et al. [2015]. Grid spacing (in mm) and number of CPDI elements along the circumference and across the thickness are the values following G, C and T, respectively.

ring, was measured to be 11.5 μs. This is in good agreement with the experimental measurement of 11.4 μs, as was given by Shim et al. [2007].

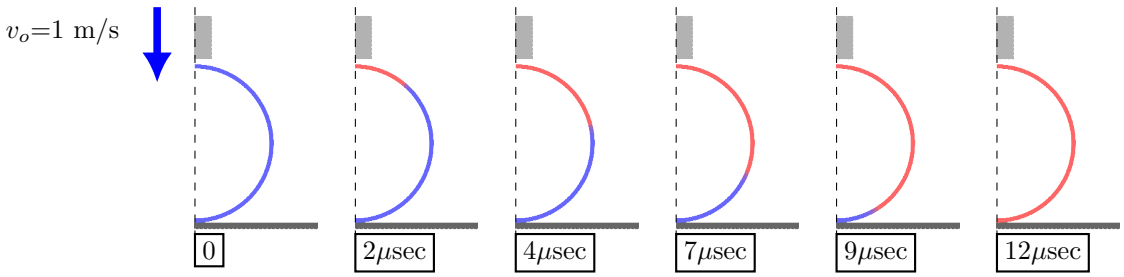


Figure 88: Snapshots of the simulations carried out on thin walled tube for stress-wave travel, Sinaie et al. [2018b].

Remark 40 *We emphasize that the FEM (with one-point quadrature shell elements) performs better than the MPM for the simulations presented in this section. Our aim was to validate the MPM for simulations of thin-walled structures subjected to various excitation conditions.*

6.3.4 Cellular structures under lateral loading

Building up on their previous work, Sinaie et al. [2019] continued to evaluate the performance of the MPM, this time for cellular structures. They carried out 3D simulations of the experiments of Langrand et al. [2017] and Shim and Stronge [1986]. Schematics of these experiments are illustrated in Fig. 89. These are signature applications of the MPM: simulations involving many contacts.

The MPM model is shown in Fig. 90. This is a 3D model but with only a single layer of particles in the z-direction. Fig. 91 shows the two parameters used to create models with different levels of refinement. The include the spacing of grid points l_g (equal in both directions) and the number of material points within each grid cell n_m (along each axis). Material parameters and geometry dimensions of both tests are given in Table 11. Again, contacts between the sample and the load platens and contacts between the structure walls are no-slip. The standard MPM (with hat functions) and the MUSL algorithm were used for all simulations.

Fig. 92 shows the simulation results of Sinaie et al. [2019] in relation to the experimental setup of Langrand et al. [2017]. These include force-deformation curves, or more precisely equivalent stress versus equivalent strain, for different values of the grid spacing l_g and particle density n_m. The results indicate a continuous convergence of simulations towards experimental measurements as a result of increased model refinement. These also indicate the contribution of l_g and n_m towards the improvement of the simulation results.

Fig. 93 shows simulated force-deformation curves obtained using a large-strain and a small-strain elasto-plastic constitutive model. Details on the two models can be found in Sinaie et al. [2019]. As seen from this figure, both formulations yield the same curve, indicating that although the overall deformations are large, there is no need to use a large-strain formulation. The authors attribute this to the small time steps used for the simulations. As a side effect, these time steps turn out to be small

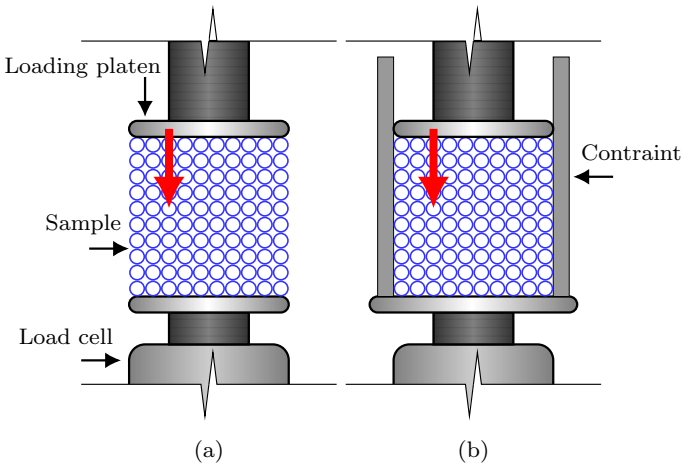


Figure 89: Schematics of the samples and test setup used by (a) Langrand et al. [2017] and (b) by Shim and Stronge [1986].

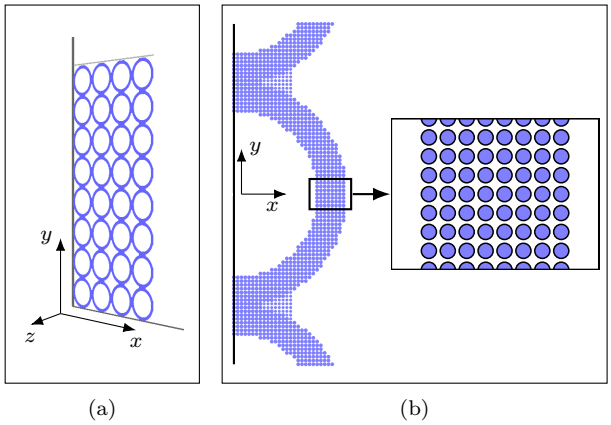


Figure 90: Layout of the material points in the x-y-z coordinate system. This numerical setup is used for the samples tested by Langrand et al. [2017]. The tubes are welded together with a braze length l_b.

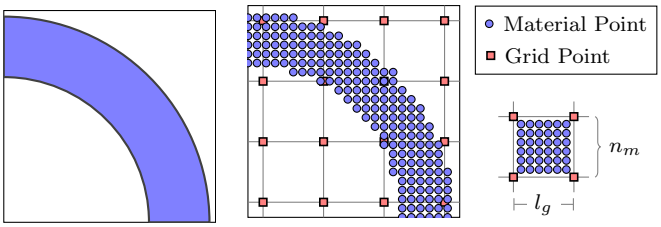


Figure 91: Discretization of bodies into distinct material points which lie within a background grid. Grid spacing is denoted by l_g and the offset between material points is equal to $ds = l_g/n_m$. The particle volume is ds^3. This is a regular particle distribution discussed in Section 4.1.1.

Table 11: Geometric and material properties of the tests carried out by Langrand et al. [2017].

Variable	meaning	Experiments	
		Langrand et al. [2017]	Shim and Stronge [1986]
D	tube outer diameter	5.0 mm	12.69 mm
t	tube thickness	0.5 mm	0.71 mm
l	sample length	40.0 mm	12.76 mm
b	sample width	40.0 mm	126.9 mm
h	sample height	42.0 mm	126.9 mm
l_b	braze length	1.5 mm	
Material		Inconel 600	Aluminum
ρ	density	8250.0 kg/m^3	2700 kg/m^3
E	elastic modulus	197.6 GPa	70.0 GPa
σ_y	yield stress	280.0 MPa	70.0 MPa
b	hardening parameter	15.0	35.0
Q	hardening parameter	420.0 (MPa)	55.0 (MPa)

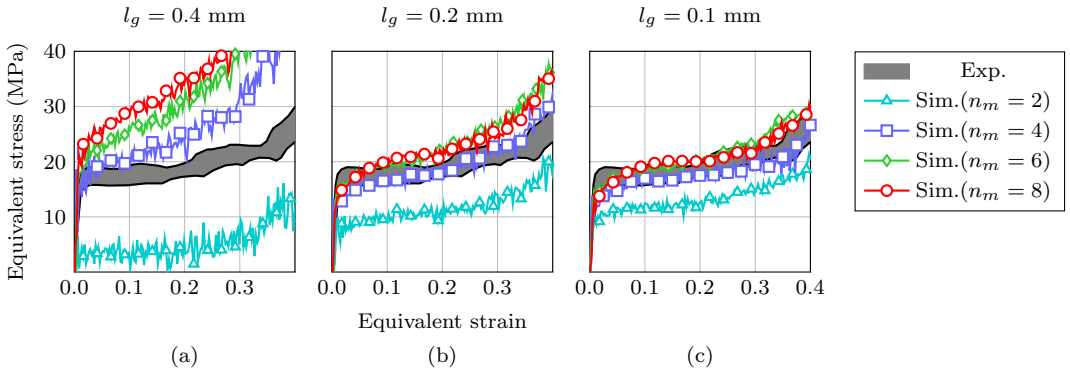


Figure 92: Force-deformation curves at different levels of refinement of the numerical model. Experimental curves are extracted from Langrand et al. [2017].

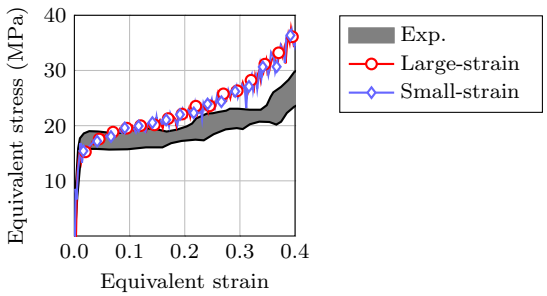


Figure 93: Comparison of large-strain and small-strain plasticity formulations using the sample tested in Langrand et al. [2017]. The particle assembly is created with $l_g = 0.2$ mm and $n_m = 6$.

enough to eliminate the need for a large-strain model. Section 6.3.4 presents the deformed configurations of the structure at different levels of compression.

Fig. 95 shows the simulation results of Sinaie et al. [2019] in relation to the experimental setup of Shim and Stronge [1986]. These include force-deformation curves (or more precisely equivalent stress versus equivalent strain) for different values of the grid spacing l_g and particle density n_m. Note the main difference between the setup used by Shim and Stronge [1986] and that of Langrand et al. [2017] is the existence of lateral constraints (Fig. 89). The results indicate a continuous convergence of simulations towards experimental measurements as a result of increased model refinement (l_g and n_m).

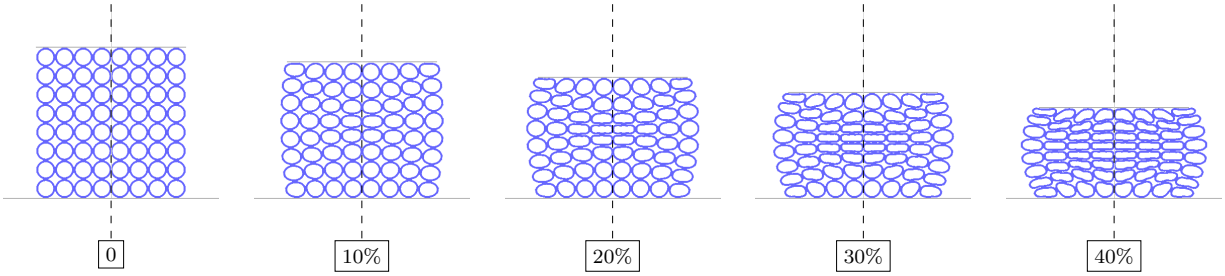


Figure 94: Simulation snapshots of the experiment of Langrand et al. [2017] taken at different levels of compression.

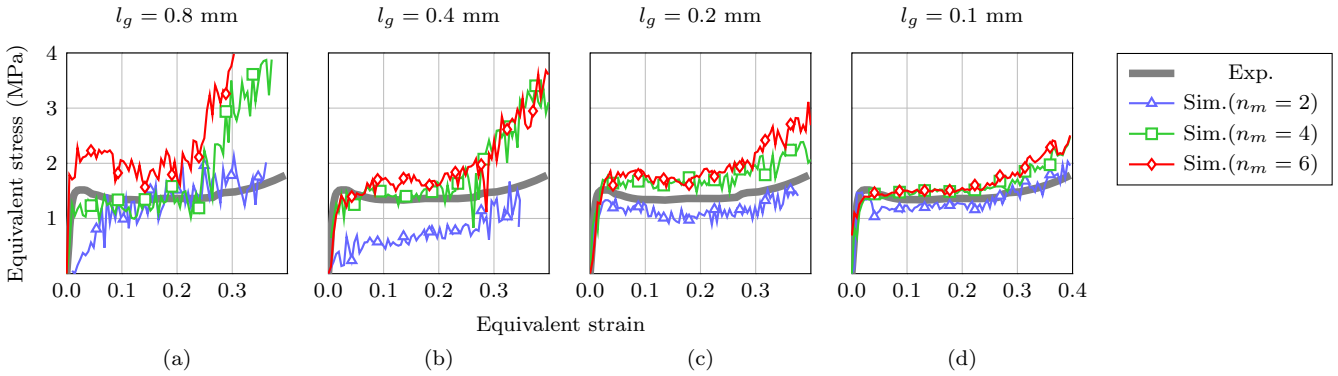


Figure 95: Force-deformation curves at different levels of refinement of the numerical model. Experimental curves are extracted from Shim and Stronge [1986].

7 Conclusions

As Prof Sulsky once said 'the MPM is a work in progress'. The result of this progress made over the last two decades is a simple and efficient platform for simulating challenging solid mechanics, fluid mechanics and fluid-structure interaction problems. This paper has presented a unified picture of the MPM covering its theory, implementation and applications. Both the updated Lagrangian and the total Lagrangian MPM formulations have been discussed. Various weighting functions employed in those MPM formulations including low order weighting functions such as the well known hat functions, high order functions such as GIMP, B-splines and Bernstein functions, as well as CPDI have been presented. A weakly compressible MPM for fluid and gases as well as algorithms for coupled thermo-mechanical problems were also outlined.

The computational algorithms presented in this contribution were implemented in various in-house codes. These codes were then used to produce all the materials reported herein. Therefore, all the results and figures are original. Furthermore, all simulations were verified either against exact solutions or experimental findings. Our discussion on the MPM is thus trust-able. For reproducible research outcomes, our code `Karamelo` is made open source.

In the remaining of this section, we briefly summarize what the MPM is good at and present aspects that need further improvements.

7.1 Application domain of the MPM

From our literature survey and our own experiences, the MPM is very suitable for problems exhibiting very large deformation and contacts. Indeed, it has been demonstrated that the MPM is much faster than the FEM implemented in commercial FE packages for such problems. Particularly, for the simulation of full densification of foam materials [Bardenhagen et al., 2005], the MPM seems to be the only tool that can do the job. For large deformation thermo-mechanical problems typically encountered in manufacturing processes, Leroch et al. [2018] also mentioned that the MPM is faster than the FEM.

Another area where the MPM is promising is image-based simulations. This is due to the fact that converting an image to a set of material points is much easier than building a body-fitted finite element mesh. Note that, this attribute applies to any meshfree method, not just the MPM.

The MPM is certainly not the best method to model fluids and gases. Nonetheless, it could be an efficient method for fluid-structure interaction (FSI) problems as it provides a unified FSI solver. Alternatively, the MPM can be used only for the solids and a more efficient method (e.g., finite volume method) is used for the fluid, if the solid undergoes very large deformation.

7.2 Improvements and future works

As with any numerical method, the MPM has its own set of shortcomings. In particular, the following items

- large memory footprint since both grid and particles are needed;
- formal analysis (convergence, error and stability) of the MPM is extremely difficult;
- enforcement of boundary conditions is difficult compared with FEM;
- lack of structural elements; and
- damage/fracture modeling.

The first item was not discussed in details in the engineering community. On the other hand, computer graphics researchers started using Sparse Paged Grid (SPGrid) to reduce the storage footprint [Gao, 2018]. The formal analysis of the MPM is extremely difficult due to the irregular distribution of the particles but also due to their relative motion with respect to the grid. The method of manufactured solution has been proved to be useful for convergence analysis of the MPM. However, only manufactured solutions for time-independent hyperelastic materials were developed. We need manufactured solutions for rate-dependent and history-dependent materials. Furthermore, no MPM variant performs well for the generalized vortex problem. For this problem, the TLMPM constitutes a good candidate as it eliminates almost all deficiencies of the MPM (cell crossing instability and quadrature error). And yet, its error plateaus for very fine meshes.

The difficult enforcement of boundary conditions is due to the lack of an explicit representation of boundaries. But, it is still easier than some other methods (e.g., SPH).

A note on efficiency and complexity is in order. Efficiency decreases with greater complexity of the numerical technique; nevertheless, the accuracy of the numerical solution is generally increased with increasing complexity. A compromise must be reached between efficiency and accuracy. Based on findings on hydrocodes, second-order accurate methods appear to be computationally more efficient than high-order methods, see e.g., Anderson Jr [1987]. That is why Sulsky and Gong [2016] aimed for second-order accuracy with their improved MPM. It is worthy noting that the standard MPM (with hat functions) when used with the modified update stress last algorithm is very robust and seems to be locking free.

We observe a lack of MPM formulations for structural elements such as beams and shells. And we do not know why. Note that bars and membranes were modeled within the MPM in Gilmanov and Acharya [2008b] (membranes) and Lian et al. [2011c] (bars). However, these formulations are actually a FEM-MPM technique. A pure MPM formulation for structural elements is not yet available.

Also, damage and fracture is not studied as much as with the FEM and meshfree methods such as EFG and SPH. A majority of works on fracture simulations using the MPM do not perform mesh sensitivity analyses. Even though post-processing simulation results generated by a particle method can be done with softwares such as `Ovito` or `VisIt`, pre-processing tools for particle methods, in particular the MPM, have not yet been developed.

As can be seen from this review, the MPM is in a state of rapid evolution. It has captured the interests of a broader community of researchers whose different and sometimes divergent skills and backgrounds can be used tackle pending limitations of the MPM. Furthermore we have witnessed collaboration between researchers from different fields such as the work presented in Gaume et al. [2018] which is a collaboration between engineers and computer graphics researchers.

Acknowledgments

The first author gratefully acknowledges the financial support of the Australian Research Council (ARC) Training Centre in Alloy Innovation for Mining Efficiency (IC160100036). The second author (V.P. Nguyen) thanks the funding support from the Australian Research Council via DECRA project DE160100577. He also acknowledges the helpful discussions with Dr Rebecca Brannon at University of Utah and Dr Debora Sulsky at University of New Mexico who was kindly to read not exactly this paper but a similar note written by Nguyen. The support from the National Natural Science Foundation of China (51678246; 51878294) to the third author (J.Y. Wu) is acknowledged.

A The method of manufactured solutions (MMS)

The method of manufactured solutions (MMS) provides a framework to verify nonlinear codes. In the MMS, the solution (i.e., the displacement field) of the model equations is assumed *a priori* i.e., being manufactured. Given the assumed prescribed

displacements, the constitutive model can be evaluated to determine the corresponding stress field. Next, the divergence of the stress and the acceleration are evaluated, and the required body forces to achieve these solutions are analytically determined from the equations of motion. Note that boundary conditions (e.g., for the velocities) and initial conditions (e.g., for the velocities and stresses) are also determined from the manufactured solutions. One then execute the code (an MPM code in our context) with this body force vector and boundary/initial conditions. The numerical solutions are then compared with the manufactured ones [Knupp and Salari, 2003]. An excellent report on the MMS can be found at http://prod.sandia.gov/techlib/access-control.cgi/2000/001444.pdf.

Wallstedt and Guilkey [2008] was the first to use the MMS to check the convergence of the MPM. Since then, the MMS has been used more. For example, a suite of code verification tests for solid mechanics problems is presented in Kamojjala et al. [2015] for rate-independent constitutive models.

For readers unfamiliar with the method, Section A.1 presents a step-by-step derivation of the body force for a 1D problem. This is followed by a discussion on error norms in Section A.2. Appendix A.3 provides a procedure to obtain the so-called convergence curve and convergence rate for a chosen manufactured solution.

A.1 An one dimensional manufactured solution

To demonstrate the MMS, in what follows we present the method in one dimension. It should be noted that the solutions are typically manufactured in the total Lagrangian form i.e., with respect to the initial configuration in the MMS as it is more convenient.

The manufactured displacement is assumed to be

$$u(X,t) = G\sin(\pi X)\sin(c\pi t) \tag{A.1}$$

where G is the maximum amplitude of the displacement; $c = \sqrt{E/\rho}$ and E denotes the Young's modulus. The period is thus given by $T = \frac{2\pi}{c\pi}$; X denotes the material coordinates i.e., coordinates in the reference configuration. The spatial domain is $0 \leq X \leq 1$ and the time domain is $0 \leq t \leq T$ i.e., one period of oscillation is considered. As can be seen, the manufactured displacements expressed in terms of sine and cosine functions i.e., smooth functions are commonly used, see e.g., Wallstedt and Guilkey [2008] even though they are not representative of general material deformations.

The velocity and acceleration are thus given by

$$\begin{aligned} v(X,t) &= \pi c G\sin(\pi X)\cos(c\pi t) \\ a(X,t) &= -\pi^2 c^2 G\sin(\pi X)\sin(c\pi t) = -\pi^2 c^2 u(X,t) \end{aligned} \tag{A.2}$$

where Equation (A.1) was used.

One has to choose a constitutive model so that the stress can be determined analytically. We use a Neo-Hookean material where the 1$^{\text{st}}$ PK stress P is given by

$$P = \lambda \ln(J)F^{-1} + \mu F^{-1}(FF - 1) \tag{A.3}$$

with λ, μ being the Lamé constants and $J = F$ is the Jacobian of the deformation. The deformation gradient F is written by using the displacement given in Equation (A.1)

$$F(X,t) = 1 + \frac{\partial u}{\partial X} = 1 + \pi G\cos(\pi X)\sin(c\pi t) \tag{A.4}$$

which results in the following expression for the spatial derivative of F

$$\frac{\partial F}{\partial X} = -\pi^2 G\sin(\pi X)\sin(c\pi t) = -\pi^2 u(X,t) \tag{A.5}$$

where use was made of Equation (A.1). The divergence of the stress, appearing in the linear momentum balance equation, is thus given by

$$\frac{\partial P}{\partial X} = \frac{\partial F}{\partial X}\left[\frac{\lambda}{F^2}(1 - \ln(F)) + \mu\left(1 + \frac{1}{F^2}\right)\right] \tag{A.6}$$

From the momentum equation given as follows

$$\rho_0 a(X,t) = \frac{\partial P}{\partial X} + \rho_0 b(X,t) \tag{A.7}$$

one can solve for the body force

$$b(X,t) = \frac{\pi^2 u(X,t)}{\rho_0}\left[\frac{\lambda}{F^2}\left(1-\ln(F)\right)+\mu\left(1+\frac{1}{F^2}\right)-E\right] \tag{A.8}$$

Besides, initial conditions are given by

$$\begin{aligned} v(X,0) &= \pi c G \sin(\pi X) \\ \sigma(X,0) &= \frac{1}{J}\left(\lambda \ln(F(X,0)) + \mu(F(X,0)F(X,0)-1)\right) = 0 \end{aligned} \tag{A.9}$$

and boundary conditions are written as

$$v(0,t) = 0, \quad v(1,t) = 0 \tag{A.10}$$

Now, one has a well defined problem, which can be solved using an MPM code. The obtained numerical displacement, denoted by $u^h(X,t)$, is then compared it with the manufactured (exact) displacement in Equation (A.1) to assess the performance of the MPM. One needs a norm for this comparison, which is discussed in what follows.

A.2 Norms

In order to assess the accuracy of the numerical solution, one needs a measure of the error made compared to the analytical solution (i.e., the manufactured solutions). However, what error measure to be adopted is confusing as different authors used different definitions. Even worse, some researchers adopted different error measures in different related works. It seems to the authors that their aim was to get convergence forcefully. Therefore, in this work, we used two different errors, the first is un-normalized while the second is. The first error is based on the "distance" measure between the numerical and analytical solution and integrated in space. This measure is a function of time, and at time step t^n it is defined as:

$$e(t^n) = \sqrt{\frac{\sum_{p=1}^{n_p} V_p^{t^n} \left|\left|\mathbf{u}_p^h(t^n) - \mathbf{u}^{\text{exact}}(\mathbf{X}_p, t^n)\right|\right|^2}{V_{\text{tot}}}} \tag{A.11}$$

where $\mathbf{u}_p^h(t^n) = \mathbf{x}_p^{t^n} - \mathbf{X}_p$ is the numerical displacement at particle p, $\mathbf{u}^{\text{exact}}(\mathbf{X}_p, t^n)$ is the exact (manufactured) displacement at p, n_p is the total number of particles in the solid and V_{tot} its total volume. In the above equation, $||\mathbf{a}||$ denotes the Euclidean norm of vector $\mathbf{a}$. However, to not have to compare the error at every time step, the first error measure is taken as the overall maximum of the error function:

$$e_1 := \max_n(e(t^n)) \tag{A.12}$$

This measure of error is not normalized and has the dimension of length. It is therefore dependent on the maximum amplitude of the displacement. One can normalize the error to make it dimensionless. The normalized error measure is defined as

$$e_2 := \sqrt{\frac{\sum_{t^n=0}^{t_f}\sum_{p=1}^{n_p} V_p^{t^n} \left|\left|\mathbf{u}_p^h(t^n) - \mathbf{u}^{\text{exact}}(\mathbf{X}_p, t^n)\right|\right|^2}{\sum_{t^n=0}^{T}\sum_{p=1}^{n_p} V_p^{t^n} \left|\left|\mathbf{u}^{\text{exact}}(\mathbf{X}_p, t^n)\right|\right|^2}} \tag{A.13}$$

For the TLMPM, as the spatial integration is carried over the initial configuration, one just simply replaces $V_p^{t^n}$ by V_p^0 in Equations (A.12) and (A.13).

A.3 Convergence rate

It is well known that the error of the numerical solution measured in some norms is related to the element size h by the following equation

$$e(h) \approx C h^{k(p)} \tag{A.14}$$

where C is a positive constant, $k(p)$ is a positive integer and p denotes the order of the basis functions – $p = 1$ for linear elements, $p = 2$ for quadratic elements etc. Equation (A.14) furnishes a practical way to check the convergence of a numerical method by taking the logarithm of both sides of the above equation

$$e(h) \approx C h^k \rightarrow \log(E(h)) \approx \log(C) + k\log(h) \tag{A.15}$$

which indicates that on a log-log plot the relation between the error and the mesh size is a line with a slope k; k is also called the *convergence rate*.

For time dependent problems, to manifest the expected theoretical (or optimal) convergence rates in space, the time step and spatial mesh size must be selected such that the space discretization errors dominate the time discretization errors. Consequently, rather small time steps must be taken. We refer to Strang and Fix [1973], Ciarlet and Lions [1991] for a formal treatment of the mathematical analysis of the FEM which can serve as a basis for the analysis of the MPM.

A practical procedure to verify the convergence rate of a dynamic MPM code is as follows

- Manufacturing a solution and determining the corresponding body forces and initial/boundary conditions;
- Using an MPM code with body forces, initial/boundary conditions determined from the first step. Repeating this step for different mesh sizes from large to very small;
- Defining a proper error norm and compute these norms for every considered meshes;
- Plotting the errors and the mesh sizes h on a log-log scale.

One convergence data is given in Table 12. As can be seen three meshes are used with a starting mesh size h equals 0.125, and the second mesh size was halved from the first mesh and so on. A convergence curve is obtained by plotting this data and the convergence rate is the slope of the convergence curve in the log-log space and this rate can be conveniently determined in Matlab using the **polyfit** command. Alternatively this rate can be obtained by computing the ratios between successive errors.

Mesh size h	Error e	ratio
0.1250000	7.281254249028449e-06	
0.0625000	3.777609954416918e-06	1.93
0.0312500	1.916379665584172e-06	1.97

Table 12: An example convergence data produced by a second-order accurate (in space) method.

B Constitutive models

This appendix presents some commonly used material models for solids. A model for isotropic linear elastic materials is given in Section B.1. A nonlinear elastic model suitable for solids undergoing large elastic deformation such as rubbers and polymers is considered in Section B.2. Appendix B.3 presents the widely used Johnson-Cook elasto-plastic model in conjunction with the Mie-Grüneisen equation of state. We choose not to give a full exposition of the model and its surrounding theory but, instead, direct the reader to appropriate references where further details can be found. In our implementation, symmetric strain and stress tensors are stored as $3x3$ matrices which is different from the common Voigt notation, often presented in FEM textbooks, where they are stored as column vectors. This is to make easy for matrix operations such as the polar decomposition of the deformation gradient tensor.

B.1 Linear elastic isotropic material

For isotropic linear elastic materials, the stress tensor is given by

$$\sigma_{ij} = \lambda \epsilon_{kk} \delta_{ij} + 2\mu \epsilon_{ij}, \boldsymbol{\sigma} = (\lambda \mathrm{tr}\boldsymbol{\epsilon})\mathbf{I} + 2\mu\boldsymbol{\epsilon} \tag{B.1}$$

where λ and μ are the Lamé's constants. The stress tensor is also often written in terms of a hydrostatic and a deviatoric part

$$\boldsymbol{\sigma} = (K \mathrm{tr}\boldsymbol{\epsilon})\mathbf{I} + 2G\boldsymbol{\epsilon}' \tag{B.2}$$

where $\boldsymbol{\epsilon}'$ denotes the deviatoric strain tensor, and the bulk modulus K and shear modulus G are given by

$$K := \frac{3\lambda + 2\mu}{3}, \quad G := \mu \tag{B.3}$$

which are related to the Young's modulus E and Poisson's ratio ν by the following equation

$$E = \frac{9KG}{3K+G}, \quad \nu = \frac{3K-2G}{2(3K+G)} \tag{B.4}$$

The stress update is written as

$$\boldsymbol{\sigma}_p^{t+\Delta t} = \boldsymbol{\sigma}_p^t + (\lambda \mathrm{tr}\Delta\boldsymbol{\epsilon}_p)\mathbf{I} + 2\mu\Delta\boldsymbol{\epsilon}_p, \quad \Delta\boldsymbol{\epsilon}_p = \Delta t \mathbf{D}_p^{t+\Delta t}, \; \mathbf{D}_p^{t+\Delta t} = \frac{1}{2}(\mathbf{L}_p^{t+\Delta t} + (\mathbf{L}_p^{t+\Delta t})^{\mathrm{T}}) \tag{B.5}$$

B.2 Neo-Hookean

The Neo-Hookean model is an isotropic hyperelastic material model which exhibits characteristics that can be identified with the familiar material parameters found in linear elastic analysis [Bonet and Wood, 1997]. For this material model, the 1^{st} Piola-Kirchhoff stress tensor $\mathbf{P}$ is expressed as a function of the deformation matrix $\mathbf{F}$ and the Lamé constants μ and λ as follows:

$$\mathbf{P} = \mu(\mathbf{F} - \mathbf{F}^{-\mathrm{T}}) + \lambda \ln J \mathbf{F}^{-\mathrm{T}} \tag{B.6}$$

where $J = \det \mathbf{F}$.

Such formulation is convenient to be used with the total Lagrangian scheme, but for other schemes, the use of the Cauchy stress tensor $\boldsymbol{\sigma}$ might be more appropriate. Since $\boldsymbol{\sigma} = \frac{1}{J}\mathbf{P}\mathbf{F}^T$, Equation (B.6) becomes:

$$\boldsymbol{\sigma} = \frac{1}{J}\left[\mu(\mathbf{F}\mathbf{F}^{\mathrm{T}} - \mathbf{I}) + \lambda \ln J \mathbf{I}\right] \tag{B.7}$$

B.3 Elasto-plastic materials

Material modeling can be divided Into three areas: volumetric response, or resistance to compressibility (equation of state), the resistance to distortion (constitutive); and the reduction in ability to carry stress as damage accumulates (failure). This section presents a temperature dependent hypoelastic-damage-plastic material model. The model is applicable to large strain and large rotation problems and suitable for problems when the elastic deformation is negligible compared with the plastic one.

To deal with the non-invariance of the stress rate under rigid body rotation [Bonet and Wood, 1997], the rigid body motion is eliminated from the strain rate, and thus from the stress rate. This is achieved by first performing a polar decomposing of $\mathbf{F}$ in rotation $\mathbf{R}$ and stretch $\mathbf{U}$ parts, *i.e.* $\mathbf{F} = \mathbf{R}\mathbf{U}$, using the singular value decomposition. Then, the rigid body rotations are subtracted from the strain rate tensor $\mathbf{D}$ to obtain the un-rotated strain rate tensor $\mathbf{d} = \mathbf{R}^{\mathrm{T}}\mathbf{D}\mathbf{R}$. Once the un-rotated stress rate is integrated into the un-rotated stress $\boldsymbol{\sigma}'$ using Algorithm 12, the later is rotated back to the current configuration: $\boldsymbol{\sigma} = \mathbf{R}\boldsymbol{\sigma}'\mathbf{R}^{\mathrm{T}}$. Note that this is an alternative to objective stress rates presented in Equation (2.15).

The Cauchy stress tensor $\boldsymbol{\sigma}$ is expressed as the sum of its isotropic part, i.e., the hydrostatic pressure ($\hat{p}$), and the traceless symmetric deviatoric stress $\boldsymbol{\sigma}^d$. An equation of state (EOS) is used to determine the hydrostatic pressure, see Appendix B.3.1. The deviatoric response is determined using a plastic flow rule in combination with a yield condition. The von Mises yield condition is adopted here. Moreover, when fracture is taken into account, it is modeled using the classic continuum damage mechanics approach. That is, the stress tensor scales linearly with a damage variable D [Lemaitre, 1985]. In summary, the model equations are

$$\begin{aligned}
&\boldsymbol{\sigma} = -\hat{p}\mathbf{I} + \boldsymbol{\sigma}^d && \text{(stress decomposition)}\\
&\hat{p} = \mathrm{EOS}(\rho, e, D, \cdots) && \text{(equation of state)}\\
&\mathbf{d}^d = \mathbf{d}^{d,e} + \mathbf{d}^{d,p} && \text{(strain rate decomposition)}\\
&\dot{\boldsymbol{\sigma}}^d = (1-D)2G(\mathbf{d}^d - \mathbf{d}^{d,p}) && \text{(isotropic hypoelastic)}\\
&f := \sigma_{\mathrm{eq}} - \sigma_f \leq 0 && \text{(von Mises yield condition)}\\
&\mathbf{d}^{d,p} = \dot{\lambda}\frac{\partial f}{\partial \boldsymbol{\sigma}^d} && \text{(associated plastic flow)}\\
&f \leq 0, \;\; \dot{\lambda} \geq 0, \;\; \dot{\lambda} f = 0 && \text{(Karush-Kuhn-Tucker conditions)}
\end{aligned} \tag{B.8}$$

where G is the shear modulus; λ is the plastic multiplier, $\mathbf{d}^d$ is the un-rotated deviatoric strain rate with $\mathbf{d}^{d,e}$ and $\mathbf{d}^{d,p}$ are the elastic and plastic parts, respectively; σ_f is the flow stress to be discussed in Appendix B.3.2, and $\sigma_{\mathrm{eq}} = \sqrt{\frac{3}{2}\boldsymbol{\sigma}^d : \boldsymbol{\sigma}^d}$ is the equivalent von Mises stress.

B.3.1 Equation of state

The hydrostatic pressure is determined using the Mie-Grüneisen EOS modified to account for damage [Wilkins, 1999]:

$$\begin{cases} \hat{p} &= \dfrac{\rho_0(1-D)c_0^2(\eta-1)\left[\eta-\dfrac{\Gamma_0}{2}(\eta-1)\right]}{[\eta-S_\alpha(\eta-1)]^2}+\Gamma_0 e; \quad \eta=\dfrac{\rho(1-D)}{\rho_0} \text{ if } \hat{p}>0 \\ \hat{p} &= \dfrac{\rho_0 c_0^2(\eta-1)[\eta-\frac{\Gamma_0}{2}(\eta-1)]}{[\eta-S_\alpha(\eta-1)]^2}+\Gamma_0 e; \quad \eta=\dfrac{\rho}{\rho_0} \text{ otherwise} \end{cases} \tag{B.9}$$

where c_0 is the bulk speed of sound, Γ_0 the Grüneisen Gamma in the reference state. S_α is the linear Hugoniot slope coefficient. Note that positive pressure is compression.

The internal energy e is written as

$$e = C_v \rho_0 (T - T_r) \tag{B.10}$$

where C_v denotes the specific heat at constant volume.

B.3.2 Johnson-Cook flow model

According to the Johnson-Cook's flow stress model scaled with damage [Johnson and Cook, 1985], the equivalent von Mises flow stress is written as

$$\sigma_f(\varepsilon_p, \dot{\varepsilon}_p, T) = \left[A + B\left(\varepsilon_p\right)^n\right]\left[1 + C\ln\dot{\varepsilon}_p^*\right]\left[1-(T^*)^m\right](1-D) \tag{B.11}$$

where ε_p is the equivalent plastic strain, ε_p^* is the normalized plastic strain rate, A the yield stress, B and n the strain hardening parameters, C the strain rate parameter, and m a temperature coefficient. This model has five experimentally determined parameters that describe quite well the response of a number of metals.

The normalized plastic strain rate and the homologous temperature T^* are given by:

$$\dot{\varepsilon}_p^* = \dot{\varepsilon}_p/\dot{\varepsilon}_0, \quad T^* = \frac{T - T_r}{T_m - T_r} \tag{B.12}$$

where $\dot{\varepsilon}_p$ and $\dot{\varepsilon}_0$ are the plastic strain rate, and the user-defined reference plastic strain rate, respectively; T_r denotes the reference temperature and T_m is the reference melting temperature. Unless otherwise stated, $\dot{\varepsilon}_0 = 1.0\ \mathrm{s}^{-1}$.

Remark 41 *As can be seen from Equation (B.11), the effect of plastic strain, its rate, temperature and damage are coupled by being multiplied by each other. In case that damage is not interested, its bracket is simply omitted. Similarly, if thermal softening is not needed, the corresponding bracket should be skipped. The temperature T can be computed solving a heat diffusion equation or it can be simply obtained from the plastic work. For high strain rate deformation, there is not sufficient time for heat conduction, and thus adiabatic condition prevails, the temperature increase can be computed as follows*

$$\Delta T = \frac{\chi}{\rho C_p}\sigma_f \Delta\varepsilon_p \tag{B.13}$$

where $0 < \chi \leq 1$ is the Taylor-Quinney coefficient that determines how much the plastic work is converted into heat. For metals, $\chi = 0.9$ is often used.

B.3.3 Damage

The amount of damage (D) in each particle is determined using the Johnson-Cook damage model, widely used for engineering applications [Johnson and Cook, 1985]. It is a strain rate dependent phenomenological model based on the local accumulation of plastic strain. According to this model, damage initiates when the accumulated equivalent plastic strain reaches the equivalent strain at failure ε_f (see Fig. 96):

$$D_{\text{init}} := \sum \frac{\Delta\varepsilon_p}{\varepsilon_f} = 1 \tag{B.14}$$

where $\Delta\varepsilon_p$ is the equivalent plastic strain increment. The equivalent strain at failure ε_f is given by Johnson-Cook's empirical equation:

$$\varepsilon_f = \left[D_1 + D_2\exp(D_3\sigma^*)\right]\left[1 + D_4\ln(\dot{\varepsilon}_p^*)\right]\left[1 + D_5 T^*\right] \tag{B.15}$$

and where $D_1, \ldots, D_5$ are five material constants, $\sigma^* = -\hat{p}/\sigma_{\text{eq}}$ is the stress triaxiality.

As this model only describes damage initiation, in order to have a complete model of the fracture phenomenon, a damage evolution model is required. Here, it was assumed that the damage variable is given by:

$$D = \begin{cases} 0 & \text{when } 0 \leq D_{\text{init}} < 1 \\ 10\,(D_{\text{init}} - 1) & \text{when } D_{\text{init}} \geq 1 \end{cases} \tag{B.16}$$

Even if this assumption affects the details of the damage propagation, it does not change the fundamentals of the implementation. Other forms for the damage evolution can be used.

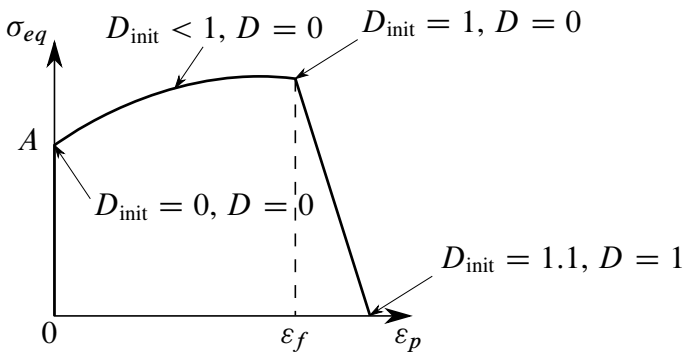


Figure 96: Schematic of a typical equivalent stress-plastic strain curve showing the evolution of both the damage initiation variable D_{init} and the damage variable D. Three points are highlighted: the yield stress A, the point at which damage initiates $D_{\text{init}} = 1$, and the point of total failure $D = 1$.

B.3.4 Algorithm

The complete stress update algorithm is given in Algorithm 11. The equivalent plastic strain and the the deviatoric stress tensor are calculated jointly and incrementally according to the stress return algorithm developed by Leroch et al. [2016] and shown in Algorithm 12. The damage is updated using Algorithm 13. As can be seen, the damage update is updated after the deviatoric stress update. This is mainly for efficiency and easy implementation. Furthermore, the flow stress is assumed to be constant during the stress update.

Algorithm 11 Stress update algorithm.

1: Inputs: ε_p^t (equivalent plastic strain), $\boldsymbol{\sigma}_t'^{\,d}$ (un-rotated deviatoric stress), $\mathbf{d}^d$ (un-rotated deviatoric strain rate)
2: Outputs: $\varepsilon_p^{t+\Delta t}$, $\boldsymbol{\sigma}_{t+\Delta t}$
3: $\mathbf{D} = 0.5(\mathbf{L} + \mathbf{L}^{\mathrm{T}})$ ▷ strain rate
4: Polar decomposition for $\mathbf{F}$ to get $\mathbf{R}$ and $\mathbf{U}$
5: $\mathbf{d} = \mathbf{R}^{\mathrm{T}}\mathbf{D}\mathbf{R}$ ▷ un-rotated strain rate
6: $\mathbf{d}^d = \mathbf{d} - (1/3)\mathrm{tr}(\mathbf{d})\mathbf{I}$ ▷ un-rotated deviatoric strain rate
7: Compute $\boldsymbol{\sigma}'^{\,d}_{t+\Delta t}(\varepsilon_p^t, D^t)$ ▷ Using Algorithm 11
8: Compute pressure $p_{t+\Delta t}(D^t)$ ▷ using an EOS
9: Compute $\boldsymbol{\sigma}'_{t+\Delta t} = \boldsymbol{\sigma}'^{\,d}_{t+\Delta t} + p_{t+\Delta t}\mathbf{I}$ ▷ un-rotated stress
10: Compute $\boldsymbol{\sigma}_{t+\Delta t} = \mathbf{R}\boldsymbol{\sigma}'_{t+\Delta t}\mathbf{R}^{\mathrm{T}}$ ▷ final stress
11: Compute damage $D^{t+\Delta t}$ ▷ using Algorithm 12

Algorithm 12 Plasticity algorithm proposed by Leroch *et al.*

Inputs: ε_p^t (equivalent plastic strain), $\boldsymbol{\sigma}_t'^d$ (un-rotated deviatoric stress), $\mathbf{d}^d$, damage D^t
Outputs: $\varepsilon_p^{t+\Delta t}$ (equivalent plastic strain), $\boldsymbol{\sigma}_{t+\Delta t}'^d$
Compute $G' = (1 - D^t)G$
$\boldsymbol{\sigma}_{\text{trial}}'^d = \boldsymbol{\sigma}_t'^d + 2G'\Delta t\mathbf{d}^d$ ▷ purely elastic stress deviator update
$\sigma_{\text{trial}}'^{\text{eq}} = \sqrt{\frac{3}{2}\boldsymbol{\sigma}_{\text{trial}}'^d : \boldsymbol{\sigma}_{\text{trial}}'^d}$ ▷ equivalent von Mises trial stress
$\sigma_f = \left[A + B\left(\varepsilon_p^t\right)^n\right]\left[1 + C\ln\dot{\varepsilon}_p^*\right]\left[1 - (T^*)^m\right](1 - D^t)$ ▷ JC flow stress
if $\sigma_{\text{trial}}'^{\text{eq}} < \sigma_f$ **then** ▷ yielding did not occur, purely elastic step
    $\boldsymbol{\sigma}_{n+1}'^d = \boldsymbol{\sigma}_{\text{trial}}'^d$ ▷ keep trial deviatoric stress
else ▷ yielding has occurred
    $\Delta\varepsilon_p = (\sigma_{\text{trial}}'^{\text{eq}} - \sigma_f)/(3G')$ ▷ compute the equivalent plastic strain increment
    $\varepsilon_p^{t+\Delta t} = \varepsilon_p^t + \Delta\varepsilon_p$ ▷ update the undamaged matrix plastic strain
    $\boldsymbol{\sigma}_{t+\Delta t}'^d = \frac{\sigma_f}{\sigma_{\text{trial}}'^{\text{eq}}}\boldsymbol{\sigma}_{\text{trial}}'^d$ ▷ scale deviatoric stress back to yield surface
end if

Algorithm 13 Damage algorithm.

Inputs: $\Delta\varepsilon_p^{t+\Delta t}$ (incremental equivalent plastic strain), $\boldsymbol{\sigma}_{t+\Delta t}$, D_{init}^t (damage initiation variable)
Outputs: $D_{\text{init}}^{t+\Delta t}$ (updated damage initiation variable), $D^{t+\Delta t}$ (updated damage)
$\sigma^* = \sigma_m/\sigma_{\text{eq}}$ ▷ Compute stress triaxiality
$\varepsilon_f = [D_1 + D_2\exp(D_3\sigma^*)]\,[1 + D_4\ln(\dot{\varepsilon}_p^*)][1 + D_5T^*]$ ▷ Strain at failure
$D_{\text{init}}^{t+\Delta t} = D_{\text{init}}^t + \Delta\varepsilon_p^{t+\Delta t}/\varepsilon_f$
if $D_{\text{init}}^{t+\Delta t} \geq 1$ **then** ▷ Damage has initiated
    $D_{t+\Delta t} = \left(D_{\text{init}}^{t+\Delta t} - 1\right)$
else ▷ Damage has not initiated
    $D_{t+\Delta t} = 0$
end if

C An alternative CPDI derivation

In this section we present an alternative view of CPDI formulations following Nguyen et al. [2017]. This is to provide another explanation of the CPDI approximation. And it also illustrates how one can couple the FEM with the MPM. We refer to Section 1.2.2 for a discussion on this coupled FEM-MPM.

In the FEM-MPM, one first computes the internal force vectors at the nodes of the particle mesh and then map those forces to the background grid nodes. Note that this technique of using a Lagrangian mesh to compute the forces and projecting these forces to a Eulerian grid to resolve collisions is very common in computer graphics to simulate cloth, strands, and hair. It is going to be shown that this procedure will yield internal force vectors which are identical to the ones obtained by using the original CPDI. It is also shown that by lumping the mass at the particle corners and then project the corner masses to the grid nodes, one obtains the same expression for the nodal mass m_I as with CPDI. The discussion is, for sake of simplicity, confined to CPDI-Q4, i.e., the particle domains are four-node quadrilaterals.

In order to demonstrate the equivalence of CPDI and FEM-MPM in two dimensions, we refer to Fig. 97. Recall that the nodal internal force vector in the MPM is given by, cf. Equation (2.36)

$$\begin{aligned} f_{xI}^{\text{int}} &= -V_p\left[(\sigma_{xx})_p\frac{\partial\phi_I}{\partial x}(\mathbf{x}_p) + (\sigma_{xy})_p\frac{\partial\phi_I}{\partial y}(\mathbf{x}_p)\right] \\ f_{yI}^{\text{int}} &= -V_p\left[(\sigma_{xy})_p\frac{\partial\phi_I}{\partial x}(\mathbf{x}_p) + (\sigma_{yy})_p\frac{\partial\phi_I}{\partial y}(\mathbf{x}_p)\right] \end{aligned} \tag{C.1}$$

And the CPDI-Q4 derivatives are written as (from Equation (3.37))

$$\begin{bmatrix}\phi_{I,x}(\mathbf{x}_p)\\ \phi_{I,y}(\mathbf{x}_p)\end{bmatrix} = \frac{1}{2V_p}\left\{N_I(\mathbf{x}_1)\begin{bmatrix}y_{24}\\ x_{42}\end{bmatrix} + N_I(\mathbf{x}_2)\begin{bmatrix}y_{31}\\ x_{13}\end{bmatrix} + N_I(\mathbf{x}_3)\begin{bmatrix}y_{42}\\ x_{24}\end{bmatrix} + N_I(\mathbf{x}_4)\begin{bmatrix}y_{13}\\ x_{31}\end{bmatrix}\right\} \tag{C.2}$$

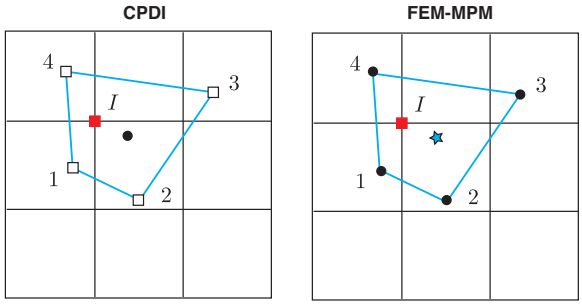


Figure 97: Equivalence between CPDI and FEM-MPM in two dimension. In the CPDI the particles (black dots) are truly particles as they carry all the state. In the FEM-MPM, the particles do not carry stresses, instead they store the internal forces. The stresses and internal variables are stored at the Gauss points (blue star) of the particle elements. The two approaches are equivalent only for constant stress elements.

Substituting Equation (C.2) into Equation (C.1) results in the following explicit expression of the grid internal force vector for the x component

$$\begin{aligned} f_{xI}^{\text{int}} = N_I(\mathbf{x}_1)\left(\frac{y_{42}}{2}\sigma_{xx} + \frac{x_{24}}{2}\sigma_{xy}\right) + N_I(\mathbf{x}_2)\left(\frac{y_{13}}{2}\sigma_{xx} + \frac{x_{31}}{2}\sigma_{xy}\right) \\ + N_I(\mathbf{x}_3)\left(\frac{y_{24}}{2}\sigma_{xx} + \frac{x_{42}}{2}\sigma_{xy}\right) + N_I(\mathbf{x}_4)\left(\frac{y_{31}}{2}\sigma_{xx} + \frac{x_{13}}{2}\sigma_{xy}\right) \end{aligned} \tag{C.3}$$

where subscript p was omitted to avoid clutter.

We focus now on the computation of the internal forces in FEM-MPM. Firstly, the internal force vector at node J, $J = 1, 2, 3, 4$, of the particle element is given by

$$\mathbf{f}_J^{\text{int}} = -\int_\Omega \begin{bmatrix} N_{J,x} & 0 & N_{J,y} \\ 0 & N_{J,y} & N_{J,x} \end{bmatrix} \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix} \mathrm{d}\Omega = -\sum_{g=1} \begin{bmatrix} N_{J,x} & 0 & N_{J,y} \\ 0 & N_{J,y} & N_{J,x} \end{bmatrix}_g \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix}_g w_g |J|_g \tag{C.4}$$

where g denotes the integration points, w_g is the integration weight and $|J|$ is the determinant of the transformation from the parent domain to the physical domain (see e.g., Hughes [2000] for details). Assuming that the stress field within the particle element is uniform to be consistent with the CPDI, the internal force can be evaluated using one single quadrature point positioned at the center of the element ($w_g = 4$ and $|J|_g = V_p/4$) yielding the x-component of the internal forces at the four corners:

$$\begin{aligned} f_{x1}^{\text{int}} &= \frac{y_{42}}{2}\sigma_{xx} + \frac{x_{24}}{2}\sigma_{xy}, \quad f_{x2}^{\text{int}} = \frac{y_{13}}{2}\sigma_{xx} + \frac{x_{31}}{2}\sigma_{xy} \\ f_{x3}^{\text{int}} &= \frac{y_{24}}{2}\sigma_{xx} + \frac{x_{42}}{2}\sigma_{xy}, \quad f_{x4}^{\text{int}} = \frac{y_{31}}{2}\sigma_{xx} + \frac{x_{13}}{2}\sigma_{xy} \end{aligned} \tag{C.5}$$

Now these internal forces at the nodes of the particle element are mapped to the background grid. At node I, we obtain

$$f_{xI}^{\text{int}} = N_I(\mathbf{x}_1) f_{x1}^{\text{int}} + N_I(\mathbf{x}_2) f_{x2}^{\text{int}} + N_I(\mathbf{x}_3) f_{x3}^{\text{int}} + N_I(\mathbf{x}_4) f_{x4}^{\text{int}} \tag{C.6}$$

which results in an internal force identical to that of the CPDI given in Equation (C.3). One can repeat the same procedure for the y component of the internal force vector which concludes the proof of the equivalence between CPDI and FEM-MPM.

Next, we prove that the nodal mass m_I obtained using CPDI or FEM-MPM is identical. Recall that the lumped mass, obtained by the row sum technique, at node c of a quadrilateral is given by

$$M_c = \int_\Omega \rho N_c \mathrm{d}\Omega = \rho\left(\int_\Omega N_c \mathrm{d}\Omega\right) = M_p\left(\frac{1}{V_p}\int_\Omega N_c \mathrm{d}\Omega\right) \tag{C.7}$$

Projecting these masses to the background grid, one obtains the nodal mass:

$$m_I = \sum_{c=1}^{4} M_c N_I(\mathbf{x}_c) = m_p\left(\sum_{c=1}^{4}\left(\frac{1}{V_p}\int_\Omega N_c \mathrm{d}\Omega\right) N_I(\mathbf{x}_c)\right) \tag{C.8}$$

where Equation (C.7) was used. Noting the CPDI weighting function ϕ_{Ip} given in Equation(3.30), one sees that the above nodal mass can also be written as $m_I = m_p\phi_{Ip}$. That concludes the proof.

D Moving least square approximations

Moving least squares (MLS) was introduced in Shepard [1968] in the late 1960s for constructing smooth approximations to fit a specified cloud of points. It was then extended in Lancaster [1981] for general surface generation problems. The most famous application of MLS approximation is probably within the diffuse element method (DEM) and element-free Galerkin (EFG) method. In an MLS scheme the fitting is done locally i.e., the fitting is different from points to points. We first present the one dimensional MLS. For a point $\bar{x}$ the approximation reads

$$u^h(x, \bar{x}) = \mathbf{p}^{\mathrm{T}}(x)\mathbf{a}(\bar{x}) \tag{D.1}$$

where the coefficients $\mathbf{a}$ vary in space, and $\mathbf{p}(x)$ is a complete polynomial of order m. They are defined as

$$\mathbf{a}(\bar{x}) = \begin{bmatrix} a_0(\bar{x}) & a_1(\bar{x}) & \dots & a_m(\bar{x}) \end{bmatrix}^{\mathrm{T}}, \quad \mathbf{p}(x) = \begin{bmatrix} 1 & x & x^2 & \dots & x^m \end{bmatrix}^{\mathrm{T}} \tag{D.2}$$

The MLS approximation is written as [Belytschko et al., 1994]

$$u^h(x) = \phi_I^{\mathrm{MLS}}(x)u_I, \quad \phi_I^{\mathrm{MLS}}(x) = \mathbf{p}^{\mathrm{T}}(x)\mathbf{A}^{-1}(x)w(x - x_I)\mathbf{p}(x_I), \quad \mathbf{A}(x) = \sum_I w(x - x_I)\mathbf{p}(x_I)\mathbf{p}^{\mathrm{T}}(x_I) \tag{D.3}$$

where $\phi_I^{\mathrm{MLS}}(x)$ is the MLS basis function, $\mathbf{A}(x)$ is the moment matrix, and $w(x - x_I)$ denotes the weight function (or kernel function). The kernel functions have compact support, thus MLS approximation is local. The support size is defined by the so called dilatation parameter or smoothing length.

D.1 Shepard approximation

If $\mathbf{p} = [1]$, the MLS approximation becomes the Shepard approximation which is given by

$$u^h(x) = \frac{\sum_{I=1}^n w(x - x_I)}{\sum_{I=1}^n w(x - x_I)} u_I \tag{D.4}$$

which can reproduce exactly a constant function i.e., if $u_I = c$ then $u^h(x) = c$. If we recall the projection of the particle velocity to the grid nodes in the MPM (for 1D problems)

$$v_I = \frac{\sum_p \phi_I(x_p) m_p v_p}{\sum_p \phi_I(x_p) m_p} = \frac{\sum_p \phi_I(x_p) m_p}{\sum_p \phi_I(x_p) m_p} v_p \tag{D.5}$$

which is similar to the Shepard approximation. That is why only a constant particle velocity field can be exactly projected to the grid using the conventional MPM.

Weight functions Some commonly-used weight functions are

- the cubic spline weight function

$$w(r) = \begin{cases} 2/3 - 4r^2 + 4r^3 & r \leq 1/2 \\ 4/3 - 4r + 4r^2 - 4/3r^3 & 1/2 < r \leq 1 \\ 0 & r > 1 \end{cases} \tag{D.6}$$

- the quartic spline weight function

$$w(r) = \begin{cases} 1 - 6r^2 + 8r^3 - 3r^4 & r \leq 1 \\ 0 & r > 1 \end{cases} \tag{D.7}$$

with

$$r = \frac{|x_I - x|}{d_I} \tag{D.8}$$

where d_I is the support size of node I.

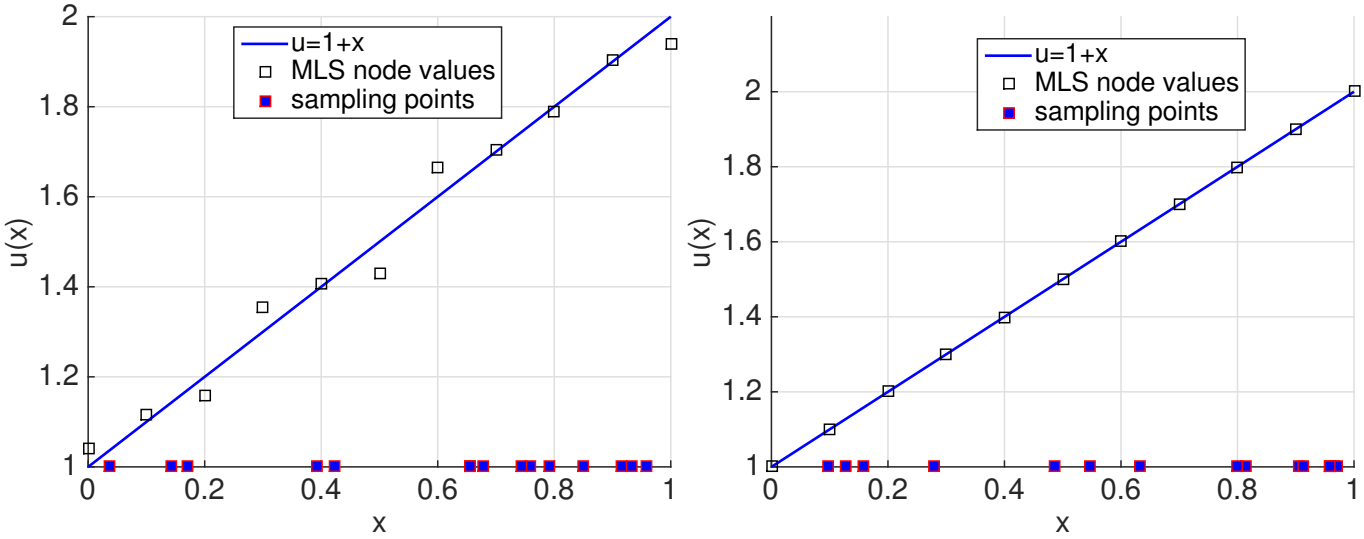


Figure 98: MLS approximation of a linear function with Shepard function (left) and linear basis $\mathbf{p} = [1 \;\; x]^{\mathrm{T}}$ (right).

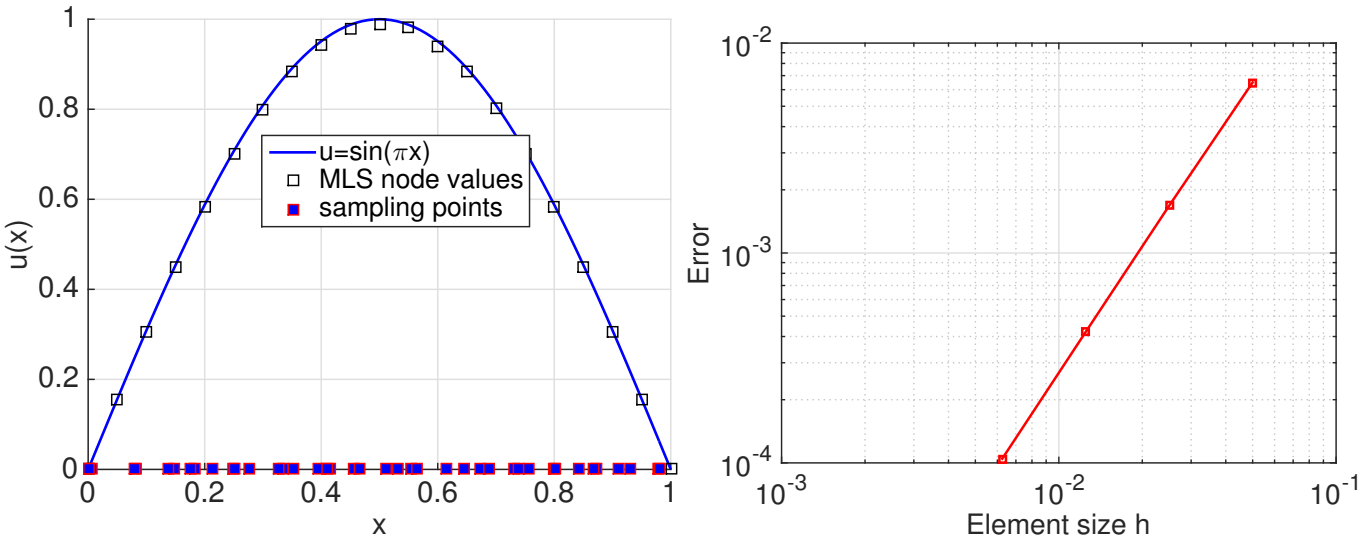


Figure 99: MLS approximation of $\sin(\pi x)$ with linear basis $\mathbf{p} = [1 \; x]^{\mathrm{T}}$. Second order convergence verified with sampling points unequally spaced within the grid elements.

D.2 Examples

We test the one dimensional MLS approximations with constant and linear basis (i.e., a Shepard approximation and a MLS with $\mathbf{p} = [1 \; x]^{\mathrm{T}}$. A unit interval is discretized by 11 equally spaced grid nodes (mesh size is thus $h = 1/10$). Fifteen sampling points are randomly distributed on the interval, denoted by x_p with $p = 1, 2, \ldots, 15$. These sampling points play the role of material points in the MPM. A linear function $u(x) = 1 + x$ is considered. At the sampling point x_p, one has $u_p = 1 + x_p$. The aim is to use MLS approximation to determine the functions at the grid nodes i.e., u_I with $I = 1, 2, \ldots, 11$. We use the cubic spline given in Equation (D.6) with $d_I = 2h$. The result given in Fig. 98 shows that the Shepard approximation is unable to exactly reproduce a linear function while the MLS with a linear basis is.

Next we study the convergence property of MLS. To this end we consider the function $\sin(\pi x)$ on a unit interval. This interval is discretized by 20, 40, 80 and 160 elements. For each element two material points are randomly distributed so that we have a distribution of material points that mimics a MPM simulation. Results are depicted in Fig. 99 which confirms the second order accuracy of linear MLS. Although not shown here the linear MLS converges for very fine meshes.

D.3 Higher dimensions

Extension of MLS to higher dimensions is straightforward. For example, in 2D the linear basis is $\mathbf{p} = [1 \; x \; y]^{\mathrm{T}}$ and thus the moment matrix is given by

$$\mathbf{A} = \sum_{I=1}^{n} w(\mathbf{x} - \mathbf{x}_I) \begin{bmatrix} 1 & x_I & y_I \\ x_I & x_I^2 & x_I y_I \\ y_I & x_I y_I & y_I^2 \end{bmatrix} \tag{D.9}$$

and due to the compact support of the weight function, the sum $\sum_I$ operates only over the points within the circle centered at $\mathbf{x}$ of radius being the smooth length. The argument to the weight functions is now written as $r = ||\mathbf{x}_I - \mathbf{x}|| / d_I$ where $||\cdot||$ is the usual Euclidean norm. Note that we are using a circular domain of influence.

E Utilities

This section presents some utilities that can be useful for junior computational mechanicians. First, Matlab codes for the visualization of one/two dimensional functions (such as the shape functions employed in the FEM and the MPM) are given in Appendix E.1. Second, utilization of SageMath – an open source computer algebra system – is discussed in in Appendix E.2. SageMath is an open source CAS that aims to be an alternative to Magma, Maple, Mathematica, and MATLAB. Furthermore, it supports technical writing using LaTeX. This system can be used to carry out lengthy and tedious derivations (such as the derivation of CPDI functions and of MMS body forces).

E.1 Scripts to plot basis functions

This section presents Matlab scripts to plot various MPM basis functions. Listing 1 is used to plot one dimensional GIMP functions on a grid of 4 cells and 5 nodes. The resulting plot was given in Fig. 17. Listing 2 presents commands to create plots of two dimensional GIMP functions given in Fig. 18. Since the codes are quite self explanatory we do not provide any explanations.

Listing 1: Matlab script to plot 1D GIMP basis functions

```
L          = 4;           % physical domain
elemCount  = 4;           % no of elements
nodes      = linspace(0,L,elemCount+1); % nodes
dx         = L/elemCount;% grid spacing
n          = 2;
ptsCount    = 250;
x           = 0:L/ptsCount:L;% sampling points where phi_I evaluated;
shapeFunc  = zeros(elemCount+1,length(x));
dshapeFunc = zeros(elemCount+1,length(x));
% compute phi_I and dphi_I, I=1:elemCount+1 at x(j): j=1:ptsCount
for I=1:elemCount+1
    for i=1:length(x)
        pts = x(i) - nodes(I);
        [shapeFunc(I,i), dshapeFunc(I,i)] = getGIMP(pts,dx,1.5);
    end
end
%%
figure(1)
hold on
plot(x,shapeFunc(1,:),'black-','LineWidth',1.4);
plot(x,shapeFunc(2,:),'red-'  ,'LineWidth',1.4);
%plot(x,sum(dshapeFunc,1),'black--' ,'LineWidth',1.4);
axis equal, axis([0 4 0 1])
```

Listing 2: Matlab script to plot 2D GIMP basis functions

```
% X,Y: grid of 200x200 points on a square 4x4
[X,Y] = meshgrid (linspace(0,4,200));
R     = zeros(size(X,1),size(X,1));
xI    = [2 2]; % node I with phi_I
lp    = 1;     % particle size
h     = 1;     % grid space
% compute phi_I at 200x200 points
for i=1:size(X,1)
    for j=1:size(X,1)
        x             = X(1,i);
        y             = Y(j,1);
        pts           = x - xI(1);
        [Nx, dshape] = getGIMP(pts,h,lp);
        pts           = y - xI(2);
        [Ny, dshape] = getGIMP(pts,h,lp);
        R(i,j) = Nx*Ny;
    end
end
% 3D plot
figure
surf (X,Y,R,'EdgeColor','none','LineStyle','none','FaceLighting','phong')
hold off
% contour plot
figure
surf (X,Y,R,'EdgeColor','none','LineStyle','none')
hold off
axis equal, view([0 90]), colorbar
```

E.2 Symbolic calculus

A computer algebra system (CAS) is a software program that allows computation over mathematical expressions in a way that is similar to the traditional hand computations. CAS have been shown extremely useful, among other things, for deriving complex expressions such as consistent material tangent matrices of complex constitutive models in the FEM community. There exists excellent commercial CAS such as Maple, Mathematica and Matlab. In this section we present the derivation of CPDI basis functions using SageMath. Other application of a CAS in the MPM is to derive the body forces corresponding to a manufactured solution and get an optimized code for that force.

Listing 3: SageMath script to derive CPDI-L3 basis functions

```
var('xi x1 x2 x3')              # define variables
N1=-0.5*(1-xi)*xi
N2= 0.5*(1+xi)*xi
N3= 1-xi^2
x = N1*x1 + N2*x2 + N3*x3
J = x.diff(xi)                  # Jacobian
N1J = N1*J
N2J = N2*J
N3J = N3*J
wf1 = N1J.integral(xi,-1,1)  # 1st func. weight without 1/Vp
wf2 = N2J.integral(xi,-1,1)
wf3 = N3J.integral(xi,-1,1)
view(wf1)
latex(wf1)
```

Listing 3 presents the SageMath scripts used to derive the CPDI-L3 basis functions (CPDI for 1D problems with each particle represented by three-node line element). Note that the final command **latex(wf1)** is to export the SageMath object **wf1** to LaTeX which was copied to our LaTeX document to generate the following equation

$$-\frac{1}{2}x_1 - \frac{1}{6}x_2 + \frac{2}{3}x_3 \tag{E.1}$$

Listing 4 presents the SageMath scripts used to derive the CPDI-Q4 basis functions. Line 13 is used to carry out double integrations.

Listing 4: SageMath script to derive CPDI-Q4 basis functions

```
1  var('xi eta x1 y1 x2 y2 x3 y3 x4 y4')
2  N1=0.25*(1-xi)*(1-eta)
3  N2=0.25*(1+xi)*(1-eta)
4  N3=0.25*(1+xi)*(1+eta)
5  N4=0.25*(1-xi)*(1+eta)
6  x = N1*x1 + N2*x2 + N3*x3 + N4 * x4
7  y = N1*y1 + N2*y2 + N3*y3 + N4 * y4
8  J = x.diff(xi)*y.diff(eta) - x.diff(eta)*y.diff(xi)
9  N1J = N1*J
10 N2J = N2*J
11 N3J = N3*J
12 N4J = N4*J
13 wf1 = integral(integral(N1J,eta,-1,1),xi,-1,1)
14 wf2 = integral(integral(N2J,eta,-1,1),xi,-1,1)
15 wf3 = integral(integral(N3J,eta,-1,1),xi,-1,1)
16 wf4 = integral(integral(N4J,eta,-1,1),xi,-1,1)
17 show(wf1.simplify_full())
18 show((wf1+wf2+wf3+wf4).simplify_full())
```

F Explicit Lagrangian finite elements

It is a common practice to verify a new method with the FEM. Therefore, it is convenient to have a FEM code in hand. Due to this reason, we present the implementation of explicit dynamic Lagrangian finite elements for large deformation solid mechanics problems. Readers are referred to textbooks e.g., Belytschko et al. [2000], Wu and Wu [2012] for derivations and a comprehensive treatment of the topic. The focus is on implementation aspects to help post-graduate students. Our discussion is confined to nonlinear elastic materials in the framework of hyperelasticity. We believe that this type of materials is sufficient to demonstrate finite element formulations for large deformation problems without delving too much into the stress update of more complex materials such as hyperelastic-plastic ones.

Finite elements using Lagrangian meshes are commonly classified as total Lagrangian formulation and updated Lagrangian formulation. In both formulations, the independent variables are the material coordinates $\mathbf{X}$ and time t. In the total Lagrangian formulation, the stress and strain are Lagrangian, i.e., they are defined with respect to the reference configuration (for example, the nominal stress is employed), spatial derivatives are computed with respect to the material coordinates. The corresponding weak form therefore involves integrals over the reference configuration. On the orther hand, the updated Lagrangian formulation uses the Eulerian strain and stress measures e.g., the Cauchy stress, spatial derivatives are computed with respect to the spatial coordinates $\mathbf{x}$. The corresponding weak form therefore involves integrals over the current (deformed) configuration.

F.1 Updated Lagrangian finite elements

The computational domain is discretized by continuum elements and the solution is advanced in time using the central difference scheme (or the leapfrog scheme). We pay attention to the flowchart and the computation of the internal force vector.

F.1.1 General flowchart

The flowchart of an updated Lagrangian FE code is given in Algorithm 14. As the mass matrix is constant, it is computed once.

Algorithm 14 Updated Lagrangian finite elements.

Initialization
  Initialize nodal velocities/displacements and stresses at Gauss points;
  Compute the consistent mass matrix, diagonalize it;
end
while $t < t_f$ **do**
  Compute nodal forces (refer to Algorithm 15)
    Compute external/internal forces $\mathbf{f}_I^{\text{ext},t}$, $\mathbf{f}_I^{\text{int},t}$
    Compute nodal force $\mathbf{f}_I^t = \mathbf{f}_I^{\text{ext},t} + \mathbf{f}_I^{\text{int},t}$
  end
  Update the velocity $\mathbf{v}_I^{t+\Delta t} = \mathbf{v}_I^t + (\mathbf{f}_I^t/m_I)\Delta t$
  Fix Dirichlet nodes I e.g., $\mathbf{v}_I^{t+\Delta t} = \bar{\mathbf{v}}$
  Update displacements $\mathbf{u}_I^{t+\Delta t} = \mathbf{u}_I^t + \Delta t\,\mathbf{v}_I^{t+\Delta t}$
  Update mesh $\mathbf{x}_I^{t+\Delta t} = \mathbf{x}_I^t + \Delta t\,\mathbf{v}_I^{t+\Delta t}$
end while

F.1.2 Computation of internal force

For simplicity we confine our discussion to two dimensions. The internal force vector at a generic node I is given by

$$\begin{bmatrix} f_{xI}^{\text{int}} \\ f_{yI}^{\text{int}} \end{bmatrix} = \int_\Omega \begin{bmatrix} N_{I,x} & 0 & N_{I,y} \\ 0 & N_{I,y} & N_{I,x} \end{bmatrix} \begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \end{bmatrix} \mathrm{d}\Omega \tag{F.1}$$

where derivatives are with respect to the current configuration. For an element with n nodes, we can collectively gather the internal force vectors of all n nodes to get the so-called element internal force vector[13]

$$\mathbf{f}_e^{\text{int}} = \int_{\Omega_e} \mathbf{B}^{\mathrm{T}}\boldsymbol{\sigma}\,\mathrm{d}\Omega \tag{F.2}$$

where Ω_e denotes the element domain and $\mathbf{B}$ is written as

$$\mathbf{B} = \begin{bmatrix} N_{1,x} & 0 & N_{2,x} & 0 & \dots & N_{n,x} & 0 \\ 0 & N_{1,x} & 0 & N_{2,x} & \dots & 0 & N_{n,y} \\ N_{1,y} & N_{I,x} & N_{2,y} & N_{2,x} & \dots & N_{n,y} & N_{n,x} \end{bmatrix} \tag{F.3}$$

which is a $3 \times 2n$ matrix.

Algorithm 15 Updated Lagrangian: computation of internal forces at time t.

for each element e **do**
  get element connectivity of e, $esctr$
  get element coordinates of e, $\mathbf{x}_e$
  initialize element internal force $\mathbf{f}_e^{\text{int}} = \mathbf{0}$
  for each Gauss point g **do**
    compute the shape functions/derivatives $\mathbf{N}(\boldsymbol{\xi}_g)$ and $\mathbf{dNdxi}(\boldsymbol{\xi}_g)$
    compute the Jacobian $\mathbf{J} = (\mathbf{x}_e)^{\mathrm{T}}\mathbf{dNdxi}$
    compute the shape function derivatives $\mathbf{dNdx} = \mathbf{dNdxiJ}^{-1}$
    compute $\mathbf{B}$ matrix
    compute stresses
    compute internal force $\mathbf{f}_e^{\text{int}} = \mathbf{f}_e^{\text{int}} + \mathbf{B}^{\mathrm{T}}\boldsymbol{\sigma}_g w_g \det\mathbf{J}$
  end for
  Assemble $\mathbf{f}_e^{\text{int}}$ to the global internal force $\mathbf{f}^{\text{int}}$ using $esctr$
end for

[13] For finite elements it is more common to work on an element basis rather than on a node basis. Therefore we do not compute the internal force node by node but rather we compute it element by element. This is not a requirement but we decided to do so to be consistent with the FEM practices.

If the Cauchy stresses are stored in a full matrix rather than in a vector (using the Voigt notation), then the element internal force is given by

$$\begin{bmatrix} f_{x1}^{\text{int}} & f_{y1}^{\text{int}} \\ f_{x2}^{\text{int}} & f_{y2}^{\text{int}} \\ \vdots & \vdots \\ f_{xn}^{\text{int}} & f_{yn}^{\text{int}} \end{bmatrix} = \int_{\Omega^e} \begin{bmatrix} \frac{\partial N_1}{\partial x} & \frac{\partial N_1}{\partial y} \\ \frac{\partial N_2}{\partial x} & \frac{\partial N_2}{\partial y} \\ \vdots & \vdots \\ \frac{\partial N_n}{\partial x} & \frac{\partial N_n}{\partial y} \end{bmatrix} \begin{bmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{yx} & \sigma_{yy} \end{bmatrix} \mathrm{d}\Omega = \int_{\Omega^e} \boldsymbol{\mathcal{B}}^{\mathrm{T}} [\boldsymbol{\sigma}] \mathrm{d}\Omega \tag{F.4}$$

which is implemented in our Matlab code. We find it convenient in Matlab to work this way when we assemble the element force into the global force.

One can compute the gradient deformation from the gradient velocity $\mathbf{L}$ as done in the MPM. Alternatively one can compute $\mathbf{F}$ directly as follows

$$\mathbf{F} = \frac{\partial \mathbf{x}}{\partial \mathbf{X}} = \left(\frac{\partial \mathbf{X}}{\partial \mathbf{x}} \right)^{-1} = \left(\mathbf{I} - \frac{\partial \mathbf{u}}{\partial \mathbf{x}} \right)^{-1} = \left(\mathbf{I} - \mathbf{u}^e \boldsymbol{\mathcal{B}}^{\mathrm{T}} \right)^{-1} \tag{F.5}$$

where $\mathbf{u}^e$ denotes the element displacement matrix of which explicit expression can be found in Equation (F.11).

F.2 Total Lagrangian finite elements

The general flowchart of a total Lagrangian FE code is similar to the UL finite element code except that one does not update the mesh i.e., the node coordinates are not updated, as we are always working on the initial reference configuration.

The TL form of the internal force vector is given by [Belytschko et al., 2000]

$$f_{iI}^{\text{int}} = \int_{\Omega_0^e} \frac{\partial N_I}{\partial X_k} P_{ki} \mathrm{d}\Omega \tag{F.6}$$

which can be rewritten explicitly as in 2D

$$\begin{bmatrix} f_{xI}^{\text{int}} & f_{yI}^{\text{int}} \end{bmatrix} = \int_{\Omega_0} \begin{bmatrix} \frac{\partial N_I}{\partial X} & \frac{\partial N_I}{\partial Y} \end{bmatrix} \begin{bmatrix} P_{xx} & P_{xy} \\ P_{yx} & P_{yy} \end{bmatrix} \mathrm{d}\Omega \tag{F.7}$$

And for an element with n nodes, Equation (F.7) yields the following expression for the elemental internal force matrix

$$\begin{bmatrix} f_{x1}^{\text{int}} & f_{y1}^{\text{int}} \\ f_{x2}^{\text{int}} & f_{y2}^{\text{int}} \\ \vdots & \vdots \\ f_{xn}^{\text{int}} & f_{yn}^{\text{int}} \end{bmatrix} = \int_{\Omega_0^e} \begin{bmatrix} \frac{\partial N_1}{\partial X} & \frac{\partial N_1}{\partial Y} \\ \frac{\partial N_2}{\partial X} & \frac{\partial N_2}{\partial Y} \\ \vdots & \vdots \\ \frac{\partial N_n}{\partial X} & \frac{\partial N_n}{\partial Y} \end{bmatrix} \begin{bmatrix} P_{xx} & P_{xy} \\ P_{yx} & P_{yy} \end{bmatrix} \mathrm{d}\Omega \tag{F.8}$$

Or in a more compact form as

$$\left[\mathbf{f}_e^{\text{int}} \right] = \int_{\Omega_0^e} \boldsymbol{\mathcal{B}}_0^{\mathrm{T}} \left[\mathbf{P} \right] \mathrm{d}\Omega, \quad \boldsymbol{\mathcal{B}}_0 = \begin{bmatrix} \boldsymbol{\mathcal{B}}_1 & \boldsymbol{\mathcal{B}}_2 & \dots & \boldsymbol{\mathcal{B}}_n \end{bmatrix}, \quad \boldsymbol{\mathcal{B}}_I = \begin{bmatrix} N_{I,X} \\ N_{I,Y} \end{bmatrix} \tag{F.9}$$

which is very similar to the corresponding UL form of the internal force. It should be noted that, when using the nominal stress the stress is now stored as a matrix (because it is a non-symmetric tensor) not as a vector, and thus the internal force is now a matrix. We use square brackets to differentiate an internal force matrix from an internal force vector.

Next we present how to compute the deformation gradient (which is needed to determine the stresses). There are different ways to achieve this goal. For example, one can compute $\mathbf{F}$ using the displacements via the following equation

$$F_{ij} = \delta_{ij} + \frac{\partial u_i}{\partial X_j} = \delta_{ij} + \frac{\partial N_I}{\partial X_j} u_{iI} \tag{F.10}$$

which can be explicitly written for an element with n nodes

$$\mathbf{F} = \mathbf{I} + \begin{bmatrix} \frac{\partial N_I}{\partial X} u_{xI} & \frac{\partial N_I}{\partial Y} u_{xI} \\ \frac{\partial N_I}{\partial X} u_{yI} & \frac{\partial N_I}{\partial Y} u_{yI} \end{bmatrix} = \mathbf{I} + \begin{bmatrix} u_{x1} & u_{x2} & \dots & u_{xn} \\ u_{y1} & u_{y2} & \dots & u_{yn} \end{bmatrix} \begin{bmatrix} \frac{\partial N_1}{\partial X} & \frac{\partial N_1}{\partial Y} \\ \frac{\partial N_2}{\partial X} & \frac{\partial N_2}{\partial Y} \\ \vdots & \vdots \\ \frac{\partial N_n}{\partial X} & \frac{\partial N_n}{\partial Y} \end{bmatrix} \tag{F.11}$$

or compactly as $\mathbf{F} = \mathbf{I} + \mathbf{u}_e \boldsymbol{\mathcal{B}}_0^{\mathrm{T}}$ with $\mathbf{u}_e$ is often referred to as the element displacement matrix. We are now ready to compute the TL internal force. The algorithm is given in Algorithm 16. Note that there are different implementations that employ the second Piola-Kirchhoff stress tensor $\mathbf{S}$. We refer to the textbook [Belytschko et al., 2000] for details.

Algorithm 16 Total Lagrangian: computation of internal forces at time t.

for each element e **do**
  get element connectivity of e, $esctr$
  get element coordinates of e, $\mathbf{X}_e$
  get element displacements of e, $\mathbf{u}_e$
  initialize element internal force $[\mathbf{f}_e^{\mathrm{int}}] = \mathbf{0}$
  for each Gauss point g **do**
    compute the shape functions/derivatives $\mathbf{N}(\boldsymbol{\xi}_g)$ and $\mathbf{dNdxi}(\boldsymbol{\xi}_g)$
    compute the Jacobian $\mathbf{J} = (\mathbf{X}_e)^{\mathrm{T}}\mathbf{dNdxi}$
    compute the shape function derivatives $\mathbf{dNdx} = \mathbf{dNdxiJ}^{-1}$
    compute $\boldsymbol{\mathcal{B}}_0$ matrix
    compute $\mathbf{F} = \mathbf{I} + \mathbf{u}_e \boldsymbol{\mathcal{B}}_0^{\mathrm{T}}$
    compute stresses $[\mathbf{P}]$ using $\mathbf{F}$
    compute internal force $[\mathbf{f}_e^{\mathrm{int}}] = [\mathbf{f}_e^{\mathrm{int}}] + \boldsymbol{\mathcal{B}}_0^{\mathrm{T}}[\mathbf{P}]_g w_g \det \mathbf{J}$
  end for
  Assemble $[\mathbf{f}_e^{\mathrm{int}}]$ to the global internal force $\mathbf{f}^{\mathrm{int}}$ using $esctr$
end for

G Axi-symmetric MPM

Structures of revolution (SOR) subject to loading which is symmetric about the axis of revolution can be effectively modeled using the so-called two dimensional axi-symmetric formulations. As can be seen from Fig. 100, a SOR is obtained by revolving a generating cross section around the axis of revolution 360°. The spatial discretization is thus only performed on the 2D cross section. The resulting axi-symmetric formulation is quite similar to the 2D one. The major modeling difference is the appearance of the circumferential (or hoop) strain and stress ($\epsilon_{\theta\theta}$ and $\sigma_{\theta\theta}$). Axi-symmetric MPM formulations have been presented in Sulsky and Schreyer [1996], Ma et al. [2013], Nairn and Guilkey [2015].

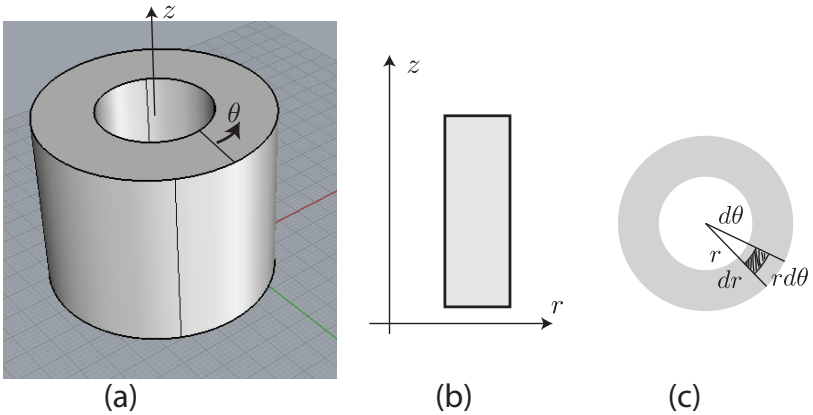


Figure 100: A solid of revolution (a) and its two dimensional representation (b). We consider a cylindrical coordinate system with coordinates (r, θ, z), and with z along the axis of symmetry.

For the ULMPM, the following are modifications for axi-symmetric problems:

- Particle mass per radian (which varies from particle to particle):

$$m_p := \int_{\Omega_p} \rho r \, \mathrm{d}\Omega = \rho A_p r_p^0 \tag{G.1}$$

where the coordinates of particle p are $\mathbf{x}_p^t := (r_p^t, z_p^t)$.

- The nodal internal force vector is given by

$$
\begin{aligned}
f_{rI}^{\text{int}} &= -\sum_{p=1}^{n_p} V_p \left[(\sigma_{rr})_p \frac{\partial \phi_I}{\partial r}(\mathbf{x}_p) + (\sigma_{rz})_p \frac{\partial \phi_I}{\partial z}(\mathbf{x}_p) + (\sigma_{\theta\theta})_p \frac{\phi_I(\mathbf{x}_p)}{r_p} \right] \\
f_{zI}^{\text{int}} &= -\sum_{p=1}^{n_p} V_p \left[(\sigma_{rz})_p \frac{\partial \phi_I}{\partial r}(\mathbf{x}_p) + (\sigma_{zz})_p \frac{\partial \phi_I}{\partial z}(\mathbf{x}_p) \right]
\end{aligned}
\tag{G.2}
$$

  which is only slightly different from the 2D case – the only modification to f_{rI}^{int} is the third term which comes from the contribution of the energy $\sigma_{\theta\theta}\epsilon_{\theta\theta}$. In the above equation, V_p is the particle volume per radian.

- The initial particle volume per radian is given by

$$
V_p^0 = m_p/\rho^0 \tag{G.3}
$$

- The velocity gradient matrix is written as

$$
\mathbf{L} = \begin{bmatrix} L_{rr} & L_{rz} & 0 \\ L_{zr} & L_{zz} & 0 \\ 0 & 0 & L_{\theta\theta} \end{bmatrix}, \quad L_{\theta\theta} = \sum_I \frac{N_I(\mathbf{x}_p)}{r_p} v_{rI} \tag{G.4}
$$

Remark 42 *If GIMP or CPDI are used, one needs to use the axi-symmetric forms of the GIMP/CPDI weighting functions. Details can be found in [Nairn and Guilkey, 2015]. For other weighting functions, the expressions for 2D plane and 3D problems can be directly used.*

Figure 101 shows a simple verification test of billet upsetting [Sulsky and Schreyer, 1996]. The test was simulated using the 3D standard MPM and the axi-symmetric MPM. The 3D simulation involves 47 744 particles whereas the axi-symmetric one contains only 696 particles. The results are very similar to that of Sulsky and Kaul [2004].

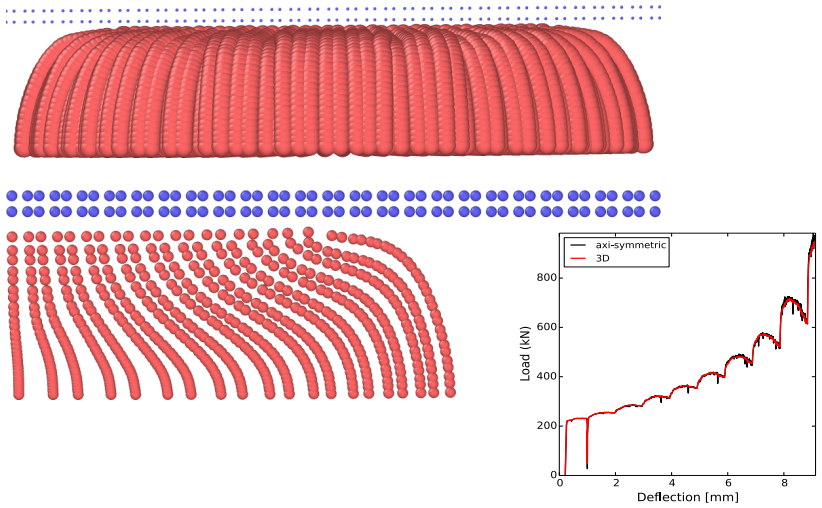


Figure 101: Simulations of billet upsetting using a 3D and an axi-symmetric MPM.

References

K. Abe, K. Soga, and S. Bandara. Material point method for coupled hydromechanical problems. *Journal of Geotechnical and Geoenvironmental Engineering*, 140(3):04013033, 2014. [Cited on page 16]

S. Agarwal, C. Senatore, T. Zhang, M. Kingsbury, K. Iagnemma, D. I. Goldman, and K. Kamrin. Modeling of the interaction of rigid wheels with dry granular media. *Journal of Terramechanics*, 85:1–14, 2019. [Cited on page 16]

I. K. Al-Kafaji. *Formulation of a Dynamic Material Point Method (MPM) for Geomechanical Problems*. PhD thesis, University of Stuttgart, 2013. [Cited on pages 12, 19, 36, and 64]

E. Alonso and F. Zabala. Progressive failure of Aznalcóllar dam using the material point method. *Géotechnique*, 61(9): 795–808, 2011. [Cited on pages 14 and 42]

R. Ambati, X. Pan, H. Yuan, and X. Zhang. Application of material point methods for cutting process simulations. *Computational Materials Science*, 57:102–110, 2012. [Cited on page 19]

S. Andersen and L. Andersen. Analysis of spatial interpolation in the material-point method. *Computers & Structures*, 88 (7-8):506–518, 2010a. [Cited on pages 40, 65, and 89]

S. Andersen and L. Andersen. Modelling of landslides with the material-point method. *Computational Geosciences*, 14(1): 137–147, 2010b. [Cited on pages 16 and 65]

S. M. Andersen. *Material-Point Analysis of Large-Strain Problems: Modelling of Landslides*. PhD thesis, Aalborg University, 2009. [Cited on pages 19 and 44]

C. E. Anderson Jr. An overview of the theory of hydrocodes. *International journal of impact engineering*, 5(1-4):33–59, 1987. [Cited on pages 39 and 108]

M. Arroyo and M. Ortiz. Local maximum-entropy approximation schemes: a seamless bridge between finite elements and meshfree methods. *International Journal for Numerical Methods in Engineering*, 65(13):2167–2202, 2006. [Cited on page 5]

S. Atluri and T. Zhu. A new meshless local petrov-galerkin (mlpg) approach in computational mechanics. *Computational Mechanics*, 22:117–127, 1998. [Cited on page 5]

F. Aurenhammer. Voronoi diagrams–a survey of a fundamental geometric data structure. *ACM Computing Survey*, 23(3): 345–405, 1991. [Cited on page 55]

G. Ayton, S. G. Bardenhagen, P. McMurtry, D. Sulsky, and G. A. Voth. Interfacing continuum and molecular dynamics: An application to lipid bilayers. *The Journal of Chemical Physics*, 114(15):6913–6924, 2001. [Cited on page 16]

I. Babuška, U. Banerjee, and J. E. Osborn. Meshless and generalized finite element methods: A survey of some major results. In M. Griebel and M. A. Schweitzer, editors, *Meshfree methods for partial differential equations*, volume 26 of *Lecture Notes in Computational Science and Engineering*, pages 1–20. Springer-Verlag, Berlin, 2002. [Cited on page 5]

B. Banerjee. Material point method simulations of fragmenting cylinders. In *17th ASCE Engineering Mechanics Conference*, University of Delaware, Newark, DE, 2004. [Cited on page 19]

S. Bardenhagen. Energy Conservation Error in the Material Point Method for Solid Mechanics. *Journal of Computational Physics*, 180(1):383–403, 2002. [Cited on pages 36 and 86]

S. Bardenhagen and E. Kober. The generalized interpolation material point method. *Computer Modeling in Engineering and Sciences*, 5(6):477–495, 2004. [Cited on pages 6, 10, 43, and 83]

S. Bardenhagen, J. Brackbill, and D. Sulsky. The material-point method for granular materials. *Computer Methods in Applied Mechanics and Engineering*, 187(3-4):529–541, 2000. [Cited on pages 12, 62, 68, and 70]

S. Bardenhagen, J. Guilkey, K. Roessig, J. Brackbill, W. Witzel, and J. Foster. Improved contact algorithm for the material point method and application to stress propagation in granular material. *Computer Modeling in Engineering and Sciences*, 2(4):509–522, 2001. [Cited on pages 12, 70, 72, and 74]

S. Bardenhagen, A. Brydon, and J. Guilkey. Insight into the physics of foam densification via numerical simulation. *Journal of the Mechanics and Physics of Solids*, 53(3):597 – 617, 2005. [Cited on pages 12, 18, 59, and 107]

S. Bardenhagen, J. Nairn, and H. Lu. Simulation of dynamic fracture with the Material Point Method using a mixed J-integral and cohesive law approach. *International Journal of Fracture*, 170(1):49–66, 2011. [Cited on page 13]

Y. Bazilevs and K. Takizawa. *Advances in Computational Fluid-Structure Interaction and Flow Simulation*. Springer, 2017. [Cited on page 17]

T. Belytschko, Y. Y. Lu, and L. Gu. Element-free Galerkin methods. *International Journal of Numerical Methods in Engineering*, 37:229–256, 1994. [Cited on pages 5 and 117]

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

T. Belytschko, W. K. Liu, and B. Moran. *Nonlinear Finite Elements for Continua and Structures*. John Wiley & Sons, Chichester, England, 2000. [Cited on pages 24, 27, 29, 35, 36, 40, 61, 121, 123, and 124]

D. J. Benson. Computational methods in Lagrangian and Eulerian hydrocodes. *Computational Methods Applied Mechanics and Engineering*, 99(2-3):235–394, 1992. [Cited on pages 12, 25, and 30]

M. Berzins. Nonlinear stability and time step selection for the MPM method. *Computational Particle Mechanics*, 5(4): 455–466, 2018. [Cited on page 19]

L. Beuth. *Formulation and Application of a Quasi-Static Material Point Method*. PhD thesis, University of Stuttgart, 2012. [Cited on pages 12 and 19]

L. Beuth, Z. Wieckowski, and P. A. Vermeer. Solution of quasi-static large-strain problems by the material point method. *International Journal for Numerical and Analytical Methods in Geomechanics*, 35(13):1451–1465, 2011. [Cited on pages 11, 16, 41, 64, and 83]

Y. Bing, M. Cortis, T. Charlton, W. Coombs, and C. Augarde. B-spline based boundary conditions in the material point method. *Computers & Structures*, 212:257–274, 2019. [Cited on page 60]

J. Bonet and R. D. Wood. *Nonlinear continuum mechanics for finite element analysis*. Cambridge university press, 1997. [Cited on page 112]

B. Bourdin, G. Francfort, and J. Marigo. The variational approach to fracture. *Journal of Elasticity*, 91(1-3):5–148, 2008. [Cited on page 14]

B. Boyce, S. Kramer, T. Bosiljevac, E. Corona, J. Moore, K. Elkhodary, C. Simha, B. Williams, A. Cerrone, A. Nonn, et al. The second sandia fracture challenge: predictions of ductile failure under quasi-static and moderate-rate dynamic loading. *International Journal of Fracture*, 198(1-2):5–100, 2016. [Cited on page 13]

B. L. Boyce, S. L. Kramer, H. E. Fang, T. E. Cordova, M. K. Neilsen, K. Dion, A. K. Kaczmarowski, E. Karasz, L. Xue, A. J. Gross, et al. The sandia fracture challenge: blind round robin predictions of ductile tearing. *International Journal of Fracture*, 186(1-2):5–68, 2014. [Cited on page 13]

J. Brackbill and H. Ruppel. FLIP: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *Journal of Computational Physics*, 65(2):314–343, 1986. [Cited on pages 6 and 32]

J. Brackbill, D. Kothe, and H. Ruppel. Flip: A low-dissipation, particle-in-cell method for fluid flow. *Computer Physics Communications*, 48(1):25 – 38, 1988. [Cited on page 6]

R. Brannon, K. Kamojjala, and A. Sadeghirad. Establishing credibility of particle methods through verification testing. *Particle-Based Methods II - Fundamentals and Applications*, pages 685–696, 2011. [Cited on pages 94 and 95]

A. Brydon, S. Bardenhagen, E. Miller, and G. Seidler. Simulation of the densification of real open-celled foam microstructures. *Journal of the Mechanics and Physics of Solids*, 53(12):2638 – 2660, 2005. [Cited on pages 12 and 18]

D. Burgess, D. Sulsky, and J. Brackbill. Mass matrix formulation of the FLIP particle-in-cell method. *Journal of Computational Physics*, 103(1):1 – 15, 1992. [Cited on page 31]

J. Burghardt, B. Leavy, J. Guilkey, Z. Xue, and R. Brannon. Application of Uintah-MPM to shaped charge jet penetration of aluminum. *IOP Conference Series: Materials Science and Engineering*, 10(1):012223, 2010. [Cited on page 16]

J. Burghardt, R. Brannon, and J. Guilkey. A nonlocal plasticity formulation for the material point method. *Computer Methods in Applied Mechanics and Engineering*, 225-228(0):55 – 64, 2012. [Cited on page 14]

O. Buzzi, D. Pedroso, and A. Giacomini. Caveats on the implementation of the generalized material point method. *Computer Modeling in Engineering and Sciences*, 1(1):1–21, 2008. [Cited on page 87]

F. Ceccato, I. Redaelli, C. di Prisco, and P. Simonini. Impact forces of granular flows on rigid structures: comparison between discontinuous (DEM) and continuous (MPM) numerical approaches. *Computers and Geotechnics*, 103:201–217, 2018. [Cited on page 15]

J.-S. Chen, M. Hillman, and S.-W. Chi. Meshfree methods: progress made after 20 years. *Journal of Engineering Mechanics*, 143(4):04017001, 2017. [Cited on page 5]

Z. Chen, W. Hu, L. Shen, X. Xin, and R. Brannon. An evaluation of the MPM for simulating dynamic failure with damage diffusion. *Engineering Fracture Mechanics*, 69(17):1873–1890, 2002. [Cited on page 14]

Z. Chen, Y. Gan, and J. Chen. A coupled thermo-mechanical model for simulating the material failure evolution due to localized heating. *COMPUTER MODELING IN ENGINEERING AND SCIENCES*, 26(2):123, 2008. [Cited on pages 12 and 76]

Z. Chen, Y. Han, S. Jiang, Y. Gan, and T. D. Sewell. A multiscale material point method for impact simulation. *Theoretical and Applied Mechanics Letters*, 2(5):051003, 2012. [Cited on page 16]

Z. Chen, X. Qiu, X. Zhang, and Y. Lian. Improved coupling of finite element method with material point method based on a particle-to-surface contact algorithm. *Computer Methods in Applied Mechanics and Engineering*, 293:1 – 19, 2015. [Cited on page 16]

Y.-J. Cheon and H.-G. Kim. An adaptive material point method coupled with a phase-field fracture model for brittle materials. *International Journal for Numerical Methods in Engineering*, 2019. [Cited on pages 14 and 85]

H. Childs, E. Brugger, B. Whitlock, J. Meredith, S. Ahern, D. Pugmire, K. Biagas, M. Miller, C. Harrison, G. H. Weber, H. Krishnan, T. Fogal, A. Sanderson, C. Garth, E. W. Bethel, D. Camp, O. Rübel, M. Durant, J. M. Favre, and P. Navrátil. VisIt: An End-User Tool For Visualizing and Analyzing Very Large Data. In *High Performance Visualization–Enabling Extreme-Scale Scientific Insight*, pages 357–372. 2012. [Cited on page 65]

A. J. Chorin. Numerical solution of the navier-stokes equations. *Mathematics of computation*, 22(104):745–762, 1968. [Cited on page 15]

A. Choudhury, M. Steffen, J. Guilkey, and S. Parker. Enchanced understanding of particle simulations through deformation-based visualization. *Computer Methods in Engineering and Sciences*, 63:117 – 136, 2010. [Cited on page 65]

P. Ciarlet and J. Lions. *Handbook of Numerical Analysis*. North-Holland, Amsterdam, 1991. [Cited on page 111]

C. Coetzee, A. Basson, and P. Vermeer. Discrete and continuum modelling of excavator bucket filling. *Journal of Terramechanics*, 44(2):177 – 186, 2007. [Cited on pages 15 and 16]

C. J. Coetzee. *The Modelling of Granular Flow Using the Particle-in-Cell Method*. PhD thesis, University of Stellenbosch, 2003. [Cited on pages 15, 19, 70, 87, and 89]

C. J. Coetzee, P. A. Vermeer, and A. H. Basson. The modelling of anchors using the material point method. *International Journal for Numerical and Analytical Methods in Geomechanics*, 29(9):879–895, 2005. [Cited on page 16]

W. M. Coombs, T. J. Charlton, M. Cortis, and C. E. Augarde. Overcoming volumetric locking in material point methods. *Computer Methods in Applied Mechanics and Engineering*, 333:1–21, 2018. [Cited on page 74]

M. Cortis, W. Coombs, C. Augarde, M. Brown, A. Brennan, and S. Robinson. Imposition of essential boundary conditions in the material point method. *International Journal for Numerical Methods in Engineering*, 113(1):130–152, 2018. [Cited on page 60]

R. Courant. Variational methods for the solution of problems of equilibrium and vibrations. *Bulletin of the American Mathematical Society*, 49(1):1–23, 1943. [Cited on page 5]

L. Cueto-Felgueroso, I. Colominas, G. Mosqueira, F. Navarrina, and M. Casteleiro. On the Galerkin formulation of the smoothed particle hydrodynamics method. *International Journal for Numerical Methods in Engineering*, 60(9):1475–1512, 2004. [Cited on page 80]

S. Cummins and J. Brackbill. An Implicit Particle-in-Cell Method for Granular Materials. *Journal of Computational Physics*, 180(2):506–548, 2002. [Cited on pages 10 and 11]

P. A. Cundall and O. D. L. Strack. A discrete numerical model for granular assemblies. *geotechnique*, 29(1):47–65, 1979. [Cited on page 15]

N. P. Daphalapurkar, H. Lu, D. Coker, and R. Komanduri. Simulation of dynamic crack growth using the generalized interpolation material point (GIMP) method. *International Journal of Fracture*, 143(1):79–102, 2007. [Cited on page 13]

G. Daviet and F. Bertails-Descoubes. A semi-implicit material point method for the continuum simulation of granular materials. *ACM Transactions on Graphics (TOG)*, 35(4):102, 2016. [Cited on page 18]

P. de Koster, R. Tielen, E. Wobbes, and M. Moller. Extension of B-spline Material Point Method for unstructured triangular grids using powell-sabin splines. *Comput. Mech.*, 2019. [Cited on pages 11 and 65]

E. de Souza Neto, D. Perić, M. Dutko, and D. Owen. Design of simple low order finite elements for large strain analysis of nearly incompressible solids. *International Journal of Solids and Structures*, 33(20–22):3277 – 3296, 1996. [Cited on page 74]

E. A. de Souza Neto, D. Peric, and D. R. Owen. *Computational methods for plasticity: theory and applications*. John Wiley & Sons, 2011. [Cited on page 33]

A. de Vaucorbeil and C. R. Hutchinson. A new Total-Lagrangian Smooth Particle Hydrodynamics approximation for the simulation of damage and fracture of ductile materials. Submitted to International Journal for Numerical Methods in Engineering, 2019. [Cited on page 100]

A. de Vaucorbeil and V. P. Nguyen. Karamelo: an open source parallel C++ package for the matertial point method. Submitted to Advances in Engineering Software, 2020. [Cited on page 66]

A. de Vaucorbeil, V. P. Nguyen, and C. R. Hutchinson. A Total-Lagrangian Material Point Method for solid mechanics problems involving large deformations. Submitted to Computer Methods in Applied Mechanics and Engineering, 2019. [Cited on pages 6, 7, 37, 92, and 95]

S. Dey, T. Børvik, O. Hopperstad, J. Leinum, and M. Langseth. The effect of target strength on the perforation of steel plates using three different projectile nose shapes. *International Journal of Impact Engineering*, 30(8):1005 – 1038, 2004. [Cited on page 100]

S. Dey, T. Børvik, O. Hopperstad, and M. Langseth. On the influence of fracture criterion in projectile impact of steel plates. *Computational Materials Science*, 38(1):176 – 191, 2006. [Cited on pages 100 and 101]

M. Doblaré, E. Cueto, B. Calvo, M. Martínez, J. Garcia, and J. Cegonino. On the employ of meshless methods in biomechanics. *Computer Methods in Applied Mechanics and Engineering*, 194(6–8):801 – 821, 2005. [Cited on page 5]

J. Dolbow and T. Belytschko. Numerical integration of the galerkin weak form in meshfree methods. *Computational mechanics*, 23(3):219–230, 1999. [Cited on page 30]

Y. Dong and J. Grabe. Large scale parallelisation of the material point method with multiple gpus. *Computers and Geotechnics*, 101:149–158, 2018. [Cited on page 66]

Q. Du, V. Faber, and M. Gunzburger. Centroidal voronoi tessellations: Applications and algorithms. *SIAM Review*, 41(4): 637–676, 1999. [Cited on page 55]

C. Duarte and J. Oden. H-p clouds- an h-p meshless method. *Numer. Methods Partial Differential Equations*, 1996. [Cited on page 5]

S. Dunatunga and K. Kamrin. Continuum modeling and simulation of granular flows through their many phases. *Journal of Fluids Mechanics*, 2015. [Cited on pages 15 and 65]

C. Dyka and R. Ingel. An approach for tension instability in smoothed particle hydrodynamics (SPH). *Computers & structures*, 57(4):573–580, 1995. [Cited on page 84]

E. Edwards and R. Bridson. A high-order accurate particle-in-cell method. *International Journal for Numerical Methods in Engineering*, 90(9):1073–1088, 2012. [Cited on page 11]

M. U. Espluga. *Analysis of Meshfree Methods for Lagrangian Fluid-Structure Interaction*. PhD thesis, Universidad Politécnica de Madrid, 2014. [Cited on page 15]

T. Fagan, V. Lemiale, J. Nairn, Y. Ahuja, R. Ibrahim, and Y. Estrin. Detailed thermal and material flow analyses of friction stir forming using a three-dimensional particle based model. *Journal of Materials Processing Technology*, 231:422–430, 2016. [Cited on page 12]

G. Fasshauer. *Meshfree Approximation Methods with MATLAB*. Interdisciplinary Mathematical Sciences (Book 6). World Scientific Publishing Company, 2007. [Cited on page 5]

J. Fern, A. Rohe, K. Soga, and E. Alonso. *The material point method for geotechnical engineering: a practical guide*. CRC Press, 2019. [Cited on pages 5, 6, and 16]

C. Fu, Q. Guo, T. Gast, C. Jiang, and J. Teran. A polynomial particle-in-cell method. *ACM Trans. Graph.*, 36(6):222:1–222:12, Nov. 2017. [Cited on page 18]

V. Galavi, L. Beuth, B. Z. Coelho, F. S. Tehrani, P. Hölscher, and F. Van Tol. Numerical simulation of pile installation in saturated sand using material point method. *Procedia Engineering*, 175:72–79, 2017. [Cited on page 16]

Y. Gan, Z. Chen, and S. Montgomery-Smith. Improved material point method for simulating the zona failure response in piezo-assisted intracytoplasmic sperm injection. *Computer Modeling in Engineering and Sciences*, pages 1–24, 2011. [Cited on pages 17 and 80]

Y. Gan, Z. Sun, Z. Chen, X. Zhang, and Y. Liu. Enhancement of the material point method using B-spline basis functions. *International Journal for Numerical Methods in Engineering*, 113(3):411–431, 2018. [Cited on pages 10 and 49]

G. C. Ganzenmüller. Smooth-Mach-Dynamics package for LAMMPS. *Fraunhofer Ernst-Mach Institute for High-Speed Dynamics*, 2014. [Cited on page 66]

M. Gao. *Sparse Paged Grid and its Applications to Adaptivity and Material Point Method in Physics Based Simulations*. PhD thesis, University of Wisconsin–Madison, 2018. [Cited on page 108]

M. Gao, A. P. Tampubolon, C. Jiang, and E. Sifakis. An adaptive generalized interpolation material point method for simulating elastoplastic materials. *ACM Transactions on Graphics (TOG)*, 36(6):223, 2017. [Cited on page 85]

J. Gaume, T. Gast, J. Teran, A. van Herwijnen, and C. Jiang. Dynamic anticrack propagation in snow. *Nature communications*, 9(1):3047, 2018. [Cited on pages 19 and 108]

M. Geers, V. Kouznetsova, and W. Brekelmans. Multi-scale computational homogenization: Trends and challenges. *Journal of Computational and Applied Mathematics*, 234(7):2175–2182, 2010. [Cited on page 19]

C. Geuzaine and J. F. Remacle. Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331, 2009. [Cited on page 95]

F. A. Gilabert, V. Cantavella, E. Sánchez, and G. Mallol. Modelling fracture process in ceramic materials using the material point method. *EPL (Europhysics Letters)*, 96(2):24002, 2011. [Cited on page 13]

A. Gilmanov and S. Acharya. A computational strategy for simulating heat transfer and flow past deformable objects. *International Journal of Heat and Mass Transfer*, 51(17-18):4415–4426, 2008a. [Cited on page 17]

A. Gilmanov and S. Acharya. A hybrid immersed boundary and material point method for simulating 3D fluid–structure interaction problems. *International Journal for Numerical Methods in Fluids*, 56(12):2151–2177, 2008b. [Cited on pages 17, 18, and 108]

R. A. Gingold and J. J. Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Mon. Notices Royal Astro. Soc.*, 181:375–389, 1977. [Cited on page 5]

M. Gong. *Improving the Material Point Method*. PhD thesis, The University of New Mexico, Albuquerque, 2015. [Cited on pages 83, 90, and 92]

F. Gracia, P. Villard, and V. Richefeu. Comparison of two numerical approaches (DEM and MPM) applied to unsteady flow. *Computational Particle Mechanics*, pages 1–19, 2019. [Cited on pages 15 and 85]

A. A. Griffith. The phenomena of rupture and flow in solids. *Philosophical Transactions of the Royal Society of Londres*, 221:163–198, 1920. [Cited on page 13]

C. Gritton and M. Berzins. Improving accuracy in the MPM method using a null space filter. *Computational Particle Mechanics*, 4(1):131–142, 2017. [Cited on pages 11 and 19]

C. Gritton, J. Guilkey, J. Hooper, D. Bedrov, R. M. Kirby, and M. Berzins. Using the material point method to model chemical/mechanical coupling in the deformation of a silicon anode. *Modelling and Simulation in Materials Science and Engineering*, 25(4):045005, 2017. [Cited on page 12]

J. Guilkey, T. Harman, and B. Banerjee. An Eulerian-Lagrangian approach for simulating explosions of energetic devices. *Computers & Structures*, 85(11-14):660–674, 2007. [Cited on page 17]

J. E. Guilkey and J. A. Weiss. Implicit time integration for the material point method: Quantitative and algorithmic comparisons with the finite element method. *International Journal for Numerical Methods in Engineering*, 57(9):1323–1338, 2003. [Cited on page 11]

J. E. Guilkey, J. B. Hoying, and J. A. Weiss. Computational modeling of multicellular constructs with the material point method. *Journal of Biomechanics*, 39(11):2074 – 2086, 2006. [Cited on pages 9, 18, 41, and 59]

Q. Guo, X. Han, C. Fu, T. Gast, R. Tamstorf, and J. Teran. A material point method for thin shells with frictional contact. *ACM Transactions on Graphics (TOG)*, 37(4):147, 2018. [Cited on page 17]

Y. Guo and J. A. Nairn. Calculation of j-integral and stress intensity factors using the material point method. *Computer Modeling in Engineering & Sciences*, 6:295–308, 2004. [Cited on page 13]

V. Gupta, S. Rajagopal, and N. Gupta. A comparative study of meshfree methods for fracture. *International Journal of Damage Mechanics*, 20(5):729–751, 2011. [Cited on page 9]

M. E. Gurtin. *An Introduction to Continuum Mechanics*. Academic Press, New York, 1981. [Cited on pages 23 and 26]

F. Hamad, D. Stolle, and P. Vermeer. Modelling of membranes in the material point method with applications. *International Journal for Numerical and Analytical Methods in Geomechanics*, 39(8):833–853, 2015. [Cited on pages 16, 17, and 18]

F. M. Hamad. *Formulation of a Dynamic Material Point Method and Applications to Soil–Water–Geotextile Systems*. PhD thesis, University of Stuttgart, 2014. [Cited on page 19]

X. Han, T. F. Gast, Q. Guo, S. Wang, C. Jiang, and J. Teran. A hybrid material point method for frictional contact with diverse materials. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 2(2):17, 2019. [Cited on page 18]

F. Harlow. The particle-in-cell computing method for fluid dynamics. *Methods for Computational Physics*, 3:319–343, 1964. [Cited on page 6]

F. H. Harlow. Fluid dynamics in Group T-3 Los Alamos National Laboratory: (LA-UR-03-3852). *Journal of Computational Physics*, 195(2):414 – 433, 2004. [Cited on page 6]

L. He and Z. Chen. Study on one-dimensional softening with localization via integrated mpm and sph. *Computational Particle Mechanics*, pages 1–8, 2019. [Cited on page 15]

L. He, Y. Gan, and Z. Chen. Preliminary effort in developing the smoothed material point method for impact. *Computational Particle Mechanics*, 6(1):45–53, 2019. [Cited on page 15]

G. A. Holzapfel. *Nonlinear Solid Mechanics*. John Wiley & Sons, New York, 2000. [Cited on page 23]

M. Homel and E. Herbold. Mesoscale modeling of porous materials using new methodology for fracture and frictional contact in the material point method. In *Dynamic Behavior of Materials, Volume 1*, pages 97–102. Springer, 2018. [Cited on page 14]

M. A. Homel and E. B. Herbold. Field-gradient partitioning for fracture and frictional contact in the material point method. *International Journal for Numerical Methods in Engineering*, 109(7):1013–1044, 2017. [Cited on pages 13, 14, 18, 68, and 71]

M. A. Homel, J. E. Guilkey, and R. M. Brannon. Continuum effective-stress approach for high-rate plastic deformation of fluid-saturated geomaterials with application to shaped-charge jet penetration. *Acta Mechanica*, pages 1–32, 2015. [Cited on page 16]

M. A. Homel, R. M. Brannon, and J. Guilkey. Controlling the onset of numerical fracture in parallelized implementations of the material point method (MPM) with convective particle domain interpolation (CPDI) domain scaling. *International Journal for Numerical Methods in Engineering*, 107(1):31–48, 2016. [Cited on pages 10, 57, 85, and 100]

P. Hu, L. Xue, K. Qu, K. Ni, and M. Brenner. *Unified Solver for Modeling and Simulation of Nonlinear Aeroelasticity and Fluid-Structure Interactions*. American Institute of Aeronautics and Astronautics, 2009. [Cited on page 17]

P. Hu, L. Xue, R. Kamakoti, Q. Li, Z. Wang, and M. Brenner. Particle-based methods with least squares technique for nonlinear aeroelasticity and fluid-structure interactions in aste-p toolset. In *52nd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference 19th AIAA/ASME/AHS Adaptive Structures Conference 13t*, page 2062, 2011. [Cited on page 17]

W. Hu and Z. Chen. Model-based simulation of the synergistic effects of blast and fragmentation on a concrete wall using the MPM. *International Journal of Impact Engineering*, 32(12):2066 – 2096, 2006. [Cited on pages 19 and 80]

Y. Hu and Y. Fang. An asynchronous material point method. In *ACM SIGGRAPH 2017 Posters*, page 60. ACM, 2017. [Cited on page 40]

Y. Hu, Y. Fang, Z. Ge, Z. Qu, Y. Zhu, A. Pradhana, and C. Jiang. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Transactions on Graphics (TOG)*, 37(4):150, 2018. [Cited on page 18]

Y. Hu, J. Liu, A. Spielberg, J. B. Tenenbaum, W. T. Freeman, J. Wu, D. Rus, and W. Matusik. Chainqueen: A real-time differentiable physical simulator for soft robotics. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6265–6271. IEEE, 2019. [Cited on page 18]

P. Huang, X. Zhang, S. Ma, and H. Wang. Shared memory OpenMP parallelization of explicit MPM and its application to hypervelocity impact. *Computer Modeling in Engineering and Sciences*, 38(2):119–147, 2008. [Cited on page 66]

P. Huang, X. Zhang, S. Ma, and X. Huang. Contact algorithms for the material point method in impact and penetration simulation. *International Journal for Numerical Methods in Engineering*, 85(4):498–517, 2011. [Cited on pages 14 and 71]

T. Hughes, J. Cottrell, and Y. Bazilevs. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering*, 194(39-41):4135–4195, 2005. [Cited on pages 47, 49, and 57]

T. J. R. Hughes. *The finite element method: linear static and dynamic finite element analysis*. Dover Publications Inc., New York, 2000. ISBN 0-486-41181-8. Corrected reprint of the 1987 original [Prentice-Hall Inc., Englewood Cliffs, N.J.]. [Cited on pages 40, 60, and 116]

I. Iaconeta, A. Larese, R. Rossi, and Z. Guo. Comparison of a material point method and a Galerkin meshfree method for the simulation of cohesive-frictional materials. *Materials*, 10(10):1150, 2017. [Cited on pages 9, 11, 15, and 36]

I. Iaconeta, A. Larese, R. Rossi, and E. Oñate. A stabilized mixed implicit material point method for non-linear incompressible solid mechanics. *Computational Mechanics*, 63(6):1243–1260, 2019. [Cited on page 74]

S. Idelsohn, E. Onate, F. D. Pin, and N. Calvo. Fluid-structure interaction using the particle finite element method. *Computer Methods in Applied Mechanics and Engineering*, 195(17-18):2100 – 2123, 2006. [Cited on page 5]

I. Ionescu, J. E. Guilkey, M. Berzins, R. M. Kirby, and J. A. Weiss. Simulation of soft tissue failure using the Material Point Method. *Journal of Biomechanical Engineering*, 128(6):917–924, 2006. [Cited on page 14]

G. R. Irwin. Analysis of stresses and strains near the end of a crack traversing a plate. *Journal of Applied Mechanics*, 24: 361–364, 1957. [Cited on page 13]

T. Jacquemin, S. Tomar, K. Agathos, S. Mohseni-Mofidi, and S. P. Bordas. Taylor-series expansion based numerical methods: A primer, performance benchmarking and new approaches for problems with non-smooth solutions. *Archives of Computational Methods in Engineering*, pages 1–49, 2019. [Cited on page 5]

I. Jassim, D. Stolle, and P. Vermeer. Two-phase dynamic analysis by material point method. *International Journal for Numerical and Analytical Methods in Geomechanics*, 37(15):2502–2522, 2013. [Cited on pages 11, 16, 41, and 64]

C. Jiang. *The Material Point Method for the Physics-Based Simulation of Solids and Fluids*. PhD thesis, University of California, Los Angles, 2015. [Cited on page 19]

C. Jiang, C. Schroeder, A. Selle, J. Teran, and A. Stomakhin. The affine particle-in-cell method. *ACM Transactions on Graphics*, 34(4):51:1–51:10, 2015a. [Cited on page 82]

C. Jiang, C. Schroeder, A. Selle, J. Teran, and A. Stomakhin. The affine particle-in-cell method. *ACM Transactions on Graphics*, 34(4):51:1–51:10, 2015b. [Cited on page 18]

C. Jiang, C. Schroeder, J. Teran, A. Stomakhin, and A. Selle. The material point method for simulating continuum materials. In *ACM SIGGRAPH 2016 Courses*, page 24. ACM, 2016. [Cited on pages 5, 18, and 19]

C. Jiang, T. Gast, and J. Teran. Anisotropic elastoplasticity for cloth, knit and hair frictional contact. *ACM Transactions on Graphics (TOG)*, 36(4):152, 2017. [Cited on page 18]

G. R. Johnson and W. H. Cook. Fracture characteristics of three metals subjected to various strains, strain rates, temperatures and pressures. *Engineering Fracture Mechanics*, 21(1):31 – 48, 1985. [Cited on page 113]

G. R. Johnson and T. J. Holmquist. Evaluation of cylinder-impact test data for constitutive model constants. *Journal of Applied Physics*, 64(8):3901–3910, 1988. [Cited on page 97]

S. G. Johnson. PyPlot module for Julia. https://github.com/stevengj/PyPlot.jl, 2012. [Cited on pages 66, 88, and 96]

L. Kachanov. Time rupture process under creep conditions. *Izv. A Rad. Nauk. SSSR otd Tekh. Nauk*, 8:26–31, 1958. [Cited on page 13]

E. Kakouris and S. P. Triantafyllou. Phase-field material point method for brittle fracture. *International Journal for Numerical Methods in Engineering*, 112(12):1750–1776, 2017a. [Cited on page 14]

E. G. Kakouris and S. P. Triantafyllou. Material point method for crack propagation in anisotropic media: a phase field approach. *Archive of Applied Mechanics*, 2017b. [Cited on page 14]

K. Kamojjala, R. Brannon, A. Sadeghirad, and J. Guilkey. Verification tests in solid mechanics. *Engineering with Computers*, 31(2):193–213, 2015. [Cited on pages 90, 94, 95, and 109]

P. Knupp and K. Salari. *Verification of Computer Codes in Computational Science and Engineering*. Chapman and Hall/CRC, 2003. [Cited on pages 19 and 109]

K. Konagai and J. Johansson. Two dimensional lagrangian particle finite-difference method for modeling large soil deformations. *Structural Engineering/Earthquake Engineering, JSCE*, 18(2):105s–110s, 2001. [Cited on page 16]

S. Kularathna and K. Soga. Comparison of two projection methods for modeling incompressible flows in mpm. *Journal of Hydrodynamics, Ser. B*, 29(3):405–412, 2017. [Cited on page 15]

G. M. Lancaster. Surfaces generated by moving least squares methods. *Math. Comput.*, 3(37):141–158, 1981. [Cited on page 117]

B. Langrand, F. Casadei, V. Marcadon, G. Portemont, and S. Kruch. Experimental and finite element analysis of cellular materials under large compaction levels. *International Journal of Solids and Structures*, 128:99–116, 2017. [Cited on pages 104, 105, 106, and 107]

R. Leavy, J. Guilkey, B. Phung, A. Spear, and R. Brannon. A convected-particle tetrahedron interpolation technique in the material-point method for the mesoscale modeling of ceramics. *Computational Mechanics*, pages 1–21, 2019. [Cited on page 55]

J. H. Lee and D. Huang. Material point method modeling of porous semi-brittle materials. *IOP Conference Series: Materials Science and Engineering*, 10(1):012093, 2010. [Cited on page 18]

N. Lelong and D. Rochais. Influence of microstructure on the dynamic behavior of a polyurethane foam with the material point method. *Materialia*, 5:100199, 2019. [Cited on page 18]

J. Lemaitre. A continuous damage mechanics model for ductile fracture. *Journal of Engineering Materials and Technology*, 107(1):83–89, jan 1985. doi: 10.1115/1.3225775. [Cited on page 112]

J. Lemaitre and J.-L. Chaboche. *Mechanics of Solid Materials*. Cambridge University Press, 1994. [Cited on page 13]

V. Lemiale, J. Nairn, and A. Hurmane. Material point method simulation of equal channel angular pressing involving large plastic strain and contact through sharp corners. *Computer Modeling in Engineering & Science*, 70(1):41–66, 2010. [Cited on pages 12, 70, 71, and 74]

S. Leroch, M. Varga, S. Eder, A. Vernes, M. R. Ripoll, and G. Ganzenmüller. Smooth particle hydrodynamics simulation of damage induced by a spherical indenter scratching a viscoplastic material. *International Journal of Solids and Structures*, 81(Supplement C):188 – 202, 2016. [Cited on page 114]

S. Leroch, S. J. Eder, G. Ganzenmüller, L. Murillo, and M. R. Ripoll. Development and validation of a meshless 3D material point method for simulating the micro-milling process. *Journal of Materials Processing Technology*, 262:449–458, 2018. [Cited on pages 12, 15, 32, and 107]

B. Li, F. Habbal, and M. Ortiz. Optimal transportation meshfree approximation schemes for fluid and plastic flows. *International journal for numerical methods in engineering*, 83(12):1541–1579, 2010. [Cited on pages 5 and 15]

F. Li, J. Pan, and C. Sinka. Modelling brittle impact failure of disc particles using material point method. *International Journal of Impact Engineering*, 38(7):653 – 660, 2011. [Cited on page 19]

J. Li, Y. Hamamoto, Y. Liu, and X. Zhang. Sloshing impact simulation with material point method and its experimental validations. *Computers & Fluids*, 103:86–99, 2014. [Cited on page 14]

S. Li and W. Liu. *Meshfree Particle Methods [Hardcover]*. Springer, 2007. ISBN 3540222561. [Cited on page 5]

X. Li and D. Sulsky. A parallel material-point method with application to solid mechanics. In C. B. M. Ingber, H. Power, editor, *Computational Science–ICCS 2002*, volume 2331 of *Applications of High-Performance Computing in Engineering VI*. WIT Press, Southampton, 2000. [Cited on page 66]

Y. Lian, X. Zhang, and Y. Liu. Coupling of finite element method with material point method by local multi-mesh contact method. *Computer Methods in Applied Mechanics and Engineering*, 200(47-48):3482–3494, 2011a. [Cited on page 16]

Y. Lian, X. Zhang, X. Zhou, S. Ma, and Y. Zhao. Numerical simulation of explosively driven metal by material point method. *International Journal of Impact Engineering*, 38(4):238–246, 2011b. [Cited on page 19]

Y. Lian, X. Zhang, X. Zhou, and Z. Ma. A FEMP method and its application in modeling dynamic response of reinforced concrete subjected to impact loading. *Computer Methods in Applied Mechanics and Engineering*, 200(17–20):1659 – 1670, 2011c. [Cited on pages 17, 18, and 108]

Y. Lian, X. Zhang, and Y. Liu. Coupling between finite element method and material point method for problems with extreme deformation. *Theoretical and Applied Mechanics Letters*, 021003:2–5, 2012. [Cited on page 16]

Y.-P. Lian, Y. Liu, and X. Zhang. Coupling of membrane element with material point method for fluid-membrane interaction problems. *International Journal of Mechanics and Materials in Design*, pages 1–13, 2014. [Cited on page 17]

Y. Liang, X. Zhang, and Y. Liu. An efficient staggered grid material point method. *Computer Methods in Applied Mechanics and Engineering*, 352:85–109, 2019. [Cited on pages 11 and 97]

K. Liew, Y. Cheng, and S. Kitipornchai. Boundary element-free method (BEFM) for two-dimensional elastodynamic analysis using Laplace transform. *International Journal for Numerical Methods in Engineering*, 64(12):1610–1627, 2005. [Cited on page 85]

L. J. Lim, A. Andreykiv, and R. B. J. Brinkgreve. Pile penetration simulation with Material Point Method. In *Installation Effects in Geotechnical Engineering*, pages 24–30. CRC Press, 2013. [Cited on page 16]

T. Liszka and J. Orkisz. The finite difference method at arbitrary irregular grids and its application in applied mechanics. *Computers & Structures*, 11:83–95, 1980. [Cited on page 5]

C. Liu and W. Sun. Shift boundary material point method: an image-to-simulation workflow for solids of complex geometries undergoing large deformation. *Computational Particle Mechanics*, pages 1–18, 2019. [Cited on pages 18 and 60]

G. Liu. *Mesh Free Methods: Moving Beyond the Finite Element Method*. CRC Press, 2002. [Cited on page 5]

G. Liu and M. Liu. *Smoothed Particle Hydrodynamics: A Meshfree Particle Method*. World Scientific, 2003. [Cited on page 5]

P. Liu, Y. Liu, and X. Zhang. Internal-structure-model based simulation research of shielding properties of honeycomb sandwich panel subjected to high-velocity impact. *International Journal of Impact Engineering*, 77:120 – 133, 2015a. [Cited on page 12]

W. K. Liu, S. Jun, and Y. F. Zhang. Reproducing kernel particle methods. *International Journal of Numerical Methods in Engineering*, 20:1081–1106, 1995. [Cited on page 5]

Y. Liu, H.-K. Wang, and X. Zhang. A multiscale framework for high-velocity impact process with combined material point method and molecular dynamics. *International Journal of Mechanics and Materials in Design*, 9(2):127–139, 2013. [Cited on page 16]

Y. Liu, X. Qiu, X. Zhang, and T. Yu. Response of woodpecker's head during pecking process simulated by material point method. *PloS one*, 10(4):e0122677, 2015b. [Cited on page 18]

M. A. Llano-Serna, M. M. Farias, and D. M. Pedroso. An assessment of the material point method for modelling large scale run-out processes in landslides. *Landslides*, 13(5):1057–1066, 2016. [Cited on page 16]

E. Love and D. Sulsky. An unconditionally stable, energy-momentum consistent implementation of the material-point method. *Computer Methods in Applied Mechanics and Engineering*, 195(33-36):3903–3925, 2006a. [Cited on pages 10 and 74]

E. Love and D. L. Sulsky. An energy-consistent material-point method for dynamic finite deformation plasticity. *International Journal for Numerical Methods in Engineering*, 65(10):1608–1638, 2006b. [Cited on pages 10 and 74]

H. Lu, N. P. Daphalapurkar, B. Wang, S. Roy, and R. Komanduri. Multiscale simulation from atomistic to continuum-coupling molecular dynamics (MD) with the material point method (MPM). *Philosophical Magazine*, 86(20):2971–2994, 2006. [Cited on page 16]

L. Lucy. A numerical approach to the testing of the fission hypothesis. *Astron. J.*, 82:1013–1024, 1977. [Cited on page 5]

J. Ma, H. Lu, and R. Komanduri. Structured mesh refinement in generalized interpolation material point (GIMP) method for simulation of dynamic problems. *Computer Modeling in Engineering and Sciences*, 12:213–227, 2006. [Cited on page 85]

J. Ma, D. Wang, and M. Randolph. A new contact algorithm in the material point method for geotechnical simulations. *International Journal for Numerical and Analytical Methods in Geomechanics*, 38(11):1197–1210, 2014. [Cited on pages 12 and 16]

S. Ma, X. Zhang, Y. Lian, and X. Zhou. Simulation of high explosive explosion using adaptive material point method. *Computer Modeling in Engineering and Sciences (CMES)*, 39(2):101, 2009a. [Cited on pages 14, 15, 19, 85, 98, and 99]

S. Ma, X. Zhang, and X. M. Qiu. Comparison study of MPM and SPH in modeling hypervelocity impact problems. *International Journal of Impact Engineering*, 36(2):272–282, 2009b. [Cited on pages 15 and 36]

X. Ma, D. Z. Zhang, P. T. Giguere, and C. Liu. Axisymmetric computation of taylor cylinder impacts of ductile and brittle materials using original and dual domain material point methods. *International Journal of Impact Engineering*, 54:96 – 104, 2013. [Cited on page 124]

Z. T. Ma, X. Zhang, and P. Huang. An object-oriented MPM framework for simulation of large deformation and contact of numerous grains. *Computer Modeling in Engineering and Sciences*, 55(1):61–87, 2010. [Cited on page 66]

L. E. Malvern. *Introduction to the Mechanics of a Continuous Medium*. Prentice-Hall International, Englewood Cliffs, New Jersey, 1969. [Cited on page 23]

S. Mao. Material point method and adaptive meshing applied to fluid-structure interaction (FSI) problems. In *ASME 2013 Fluids Engineering Division Summer Meeting*, pages V01BT13A004–V01BT13A004. American Society of Mechanical Engineers, 2013. [Cited on pages 17 and 80]

S. Mao, Q. Chen, D. Li, and Z. Feng. Modeling of free surface flows using improved material point method and dynamic adaptive mesh refinement. *Journal of Engineering Mechanics*, 142(2):04015069, 2015. [Cited on page 85]

J. E. Marsden and T. J. R. Hughes. *Mathematical Foundations of Elasticity*. Prentice-Hall, Englewood Cliffs, New Jersey, 1983. [Cited on pages 23 and 26]

C. Mast, P. Mackenzie-Helnwein, P. Arduino, G. Miller, and W. Shin. Mitigating kinematic locking in the material point method. *Journal of Computational Physics*, 231(16):5351–5373, 2012. [Cited on pages 14, 74, and 81]

C. M. Mast. *Modeling Landslide-Induced Flow Interactions with Structures using the Material Point Method*. PhD thesis, University of Washington, 2013. [Cited on page 19]

S. May, J. Vignollet, and R. d. Borst. Powell–sabin b-splines and unstructured standard t-splines for the solution of the kirchhoff–love plate theory exploiting bézier extraction. *International Journal for Numerical Methods in Engineering*, 107(3):205–233, 2016. [Cited on page 65]

C. Miehe, F. Welschinger, and M. Hofacker. Thermodynamically consistent phase-field models of fracture: Variational principles and multi-field fe implementations. *Int. J. Numer. Meth. Engng.*, 83:1273–1311, 2010. [Cited on page 14]

T. Mishra, G. C. Ganzenmüller, M. de Rooij, M. Shisode, J. Hazrati, and D. J. Schipper. Modelling of ploughing in a single-asperity sliding contact using material point method. *Wear*, 418:180–190, 2019. [Cited on page 19]

J. J. Monaghan. Simulating free surface flows with sph. *Journal of computational physics*, 110(2):399–406, 1994. [Cited on page 80]

L. Moresi, F. Dufour, and H.-B. Mühlhaus. A Lagrangian integration point finite element method for large deformation modeling of viscoelastic geomaterials. *Journal of Computational Physics*, 184(2):476 – 497, 2003. [Cited on pages 18 and 30]

L. Moresi, S. Quenette, V. Lemiale, C. Mériaux, B. Appelbe, and H.-B. Mühlhaus. Computational approaches to studying nonlinear dynamics of the crust and mantle. *Physics of the Earth and Planetary Interiors*, 163(1-4):69–82, 2007. [Cited on pages 18 and 30]

G. Moutsanidis, D. Kamensky, D. Z. Zhang, Y. Bazilevs, and C. C. Long. Modeling strong discontinuities in the material point method using a single velocity field. *Computer Methods in Applied Mechanics and Engineering*, 345:584–601, 2019a. [Cited on page 13]

G. Moutsanidis, J. J. Koester, M. R. Tupek, J.-S. Chen, and Y. Bazilevs. Treatment of near-incompressibility in meshfree and immersed-particle methods. *Computational Particle Mechanics*, pages 1–19, 2019b. [Cited on page 75]

H.-B. Mühlhaus, H. Sakaguchi, L. Moresi, and M. Fahey. Discrete and continuum modelling of granular materials. In P. Vermeer, H. Herrmann, S. Luding, W. Ehlers, S. Diebels, and E. Ramm, editors, *Continuous and Discontinuous Modelling of Cohesive-Frictional Materials*, volume 568 of *Lecture Notes in Physics*, pages 185–204. Springer Berlin Heidelberg, 2001. [Cited on page 16]

A. Nair and S. Roy. Implicit Time Integration in the Generalized Interpolation Material Point Method for Finite Deformation Hyperelasticity. *Mechanics of Advanced Materials and Structures*, 19(6):465–473, 2012. [Cited on page 10]

J. Nairn. Numerical simulations of transverse compression and densification in wood. *Wood and Fiber Science*, 38(4): 576–591, 2006. [Cited on pages 12 and 18]

J. Nairn. Material point method simulations of transverse fracture in wood with realistic morphologies. *Holzforschung*, 61 (4):375–381, 2007a. [Cited on pages 13 and 59]

J. Nairn. Analytical and numerical modeling of R curves for cracks with bridging zones. *International Journal of Fracture*, 155(2):167–181, 2009. [Cited on page 13]

J. A. Nairn. Material Point Method Calculations with Explicit Cracks. *Computer Modeling in Engineering and Sciences*, 4 (6):649–663, 2003. [Cited on pages 13 and 36]

J. A. Nairn. Numerical implementation of imperfect interfaces. *Computational Materials Science*, 40(4):525 – 536, 2007b. [Cited on pages 13 and 70]

J. A. Nairn. Modeling imperfect interfaces in the material point method using multimaterial methods. *Computer Methods in Applied Mechanics and Engineering*, 1(1):1–15, 2013. [Cited on page 12]

J. A. Nairn and J. E. Guilkey. Axisymmetric form of the generalized interpolation material point method. *International journal for numerical methods in engineering*, 101(2):127–147, 2015. [Cited on pages 12, 56, 76, 124, and 125]

J. A. Nairn, S. Bardenhagen, and G. Smith. Generalized contact and improved frictional heating in the material point method. *Computational Particle Mechanics*, 5(3):285–296, 2018. [Cited on pages 12 and 74]

B. Nayroles, G. Touzot, and P. Villon. Generalizing the finite element method: Diffuse approximation and diffuse elements. *Computational Mechanics*, 10:307–318, 1992. [Cited on page 5]

E. A. S. Neto, F. M. A. Pires, and D. R. J. Owen. 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, 2005. [Cited on pages 74 and 75]

T. Nguyen, A. Van Tol, A. Elkadi, and A. Rohe. Numerical investigation of pile installation effects in sand using material point method. *Computers and Geotechnics*, 73:58–71, 2016. [Cited on page 16]

V. P. Nguyen. Discontinuous Galerkin/Extrinsic cohesive zone modeling: implementation caveats and applications in computational fracture mechanics. *Engineering Fracture Mechanics*, 128:37–68, 2014a. [Cited on page 13]

V. P. Nguyen. Material point method: basics and applications. https://www.researchgate.net/publication/262415477_Material_point_method_basics_and_applications_Contents, 2014b. Online. [Cited on page 19]

V. P. Nguyen, T. Rabczuk, S. Bordas, and M. Duflot. Meshless methods: A review and computer implementation aspects. *Mathematics and Computers in Simulation*, 79(3):763–813, 2008. ISSN 0378-4754. [Cited on page 5]

V. P. Nguyen, M. Stroeven, and L. J. Sluys. Multiscale continuous and discontinuous modelling of heterogeneous materials: A review on recent developments. *Journal of Multiscale Modelling*, 3(4):1–42, 2012. [Cited on page 19]

V. P. Nguyen, C. T. Nguyen, T. Rabczuk, and S. Natarajan. On a family of convected particle domain interpolations in the material point method. *Finite Elements in Analysis and Design*, 126:50–64, 2017. [Cited on pages 17, 50, 54, 55, and 115]

R. W. Ogden. *Non-Linear Elastic Deformations*. Ellis Harwood Ltd, Chichester, England, 1984. [Cited on pages 23 and 26]

X. F. Pan, A. Xu, G. Zhang, and J. Zhu. Generalized interpolation material point approach to high melting explosive with cavities under shock. *Journal of Physics D: Applied Physics*, 41(1):015401, 2008. [Cited on page 19]

S. Parker. A component-based architecture for parallel multi-physics pde simulation. In P. Sloot, A. Hoekstra, C. Tan, and J. Dongarra, editors, *Computational Science – ICCS 2002*, volume 2331 of *Lecture Notes in Computer Science*, pages 719–734. Springer Berlin Heidelberg, 2002. [Cited on page 66]

S. Parker, J. Guilkey, and T. Harman. A component-based parallel infrastructure for the simulation of fluid-structure interaction. *Engineering with Computers*, 22(3-4):277–292, 2006. [Cited on pages 6 and 66]

C. S. Peskin. The immersed boundary method. *Acta numerica*, 11:479–517, 2002. [Cited on page 17]

L. A. Piegl and W. Tiller. *The NURBS Book*. Springer, 1996. ISBN 3540615458. [Cited on pages 47 and 49]

N. Pinyol, M. Alvarado, E. Alonso, and F. Zabala. Thermal effects in landslide mobility. *Géotechnique*, 68(6):528–545, 2017. [Cited on page 16]

S. Plimpton. Fast parallel algorithms for short-range molecular dynamics. *Journal of computational physics*, 117(1):1–19, 1995. [Cited on pages 6 and 66]

M. J. Powell and M. A. Sabin. Piecewise quadratic approximations on triangles. *ACM Transactions on Mathematical Software (TOMS)*, 3(4):316–325, 1977. [Cited on pages 11 and 65]

W. W. Predebon, C. E. Anderson, and J. D. Walker. Inclusion of evolutionary damage measures in Eulerian wavecodes. *Computational Mechanics*, 7(4):221–236, 1991. [Cited on page 97]

N. S. Pruijn. The improvement of the material point method by increasing efciency and accuracy. Master's thesis, TU Delft, 2016. [Cited on page 64]

T. Rabczuk. Computational methods for fracture in brittle and quasi-brittle solids: State-of-the-art review and future perspective s. *ISRN Applied Mathematics*, 2013. doi: 10.1155/2013/849231. Article ID 849231, 38 pages. [Cited on page 13]

P. W. Randles and L. D. Libersky. Smoothed particle hydrodynamics: some recent improvements and applications. *Comput. Methods Appl. Mech. Eng.*, 139(1–4):375–408, 1996. ISSN 0045-7825. [Cited on page 71]

S. J. Raymond, B. Jones, and J. R. Williams. A strategy to couple the material point method (mpm) and smoothed particle hydrodynamics (sph) computational techniques. *Computational Particle Mechanics*, 5(1):49–58, 2018. [Cited on pages 9 and 15]

S. J. Raymond, B. D. Jones, and J. R. Williams. Modeling damage and plasticity in aggregates with the material point method (mpm). *Computational Particle Mechanics*, 6(3):371–382, 2019. [Cited on page 14]

G. Remmerswaal. Development and implementation of moving boundary conditions in the material point method. Master's thesis, TU Delft, 2017. [Cited on page 60]

A. Renaud, T. Heuzé, and L. Stainier. A discontinuous galerkin material point method for the solution of impact problems in solid dynamics. *Journal of computational physics*, 369:80–102, 2018. [Cited on page 7]

A. Renaud, T. Heuzé, and L. Stainier. Stability properties of the discontinuous galerkin material point method for hyperbolic problems in one and two space dimensions. *International Journal for Numerical Methods in Engineering*, 2019. [Cited on page 7]

J. Rots, P. Nauta, G. Kusters, and J. Blaauwendraad. Smeared crack approach and fracture localization in concrete. *Heron*, 30:1–47, 1985. [Cited on page 14]

J. G. Rots. Smeared and discrete representations of localized fracture. *International Journal of Fracture*, 51(1):45–59, 1991. [Cited on page 14]

K. P. Ruggirello and S. C. Schumacher. A comparison of parallelization strategies for the Material Point Method. In *11th World Congress on Computational Mechanics*, pages 20–25, 2014. [Cited on page 67]

M. Sabel, C. Sator, and R. Müller. A particle finite element method for machining simulations. *Computational Mechanics*, 54(1):123–131, 2014. [Cited on page 5]

A. Sadeghirad, R. M. Brannon, and J. Burghardt. A convected particle domain interpolation technique to extend applicability of the material point method for problems involving massive deformations. *International Journal for Numerical Methods in Engineering*, 86(12):1435–1456, 2011. [Cited on pages 6, 10, 43, 44, 50, and 52]

A. Sadeghirad, R. Brannon, and J. Guilkey. Second-order convected particle domain interpolation (CPDI2) with enrichment for weak discontinuities at material interfaces. *International Journal for Numerical Methods in Engineering*, 95(11): 928–952, 2013. [Cited on pages 10, 44, 50, 53, and 54]

J. Sanchez. *A Critical Evaluation of Computational Fracture Using a Smeared Crack Approach in MPM*. PhD thesis, University of New Mexico, 2011. [Cited on page 14]

J. Sanchez, H. Schreyer, D. Sulsky, and P. Wallstedt. Solving quasi-static equations with the material-point method. *International Journal for Numerical Methods in Engineering*, 103(1):60–78, 2015. [Cited on page 10]

L. Scholtès and F. V. Donzé. Modelling progressive failure in fractured rock masses using a 3D discrete element method. *International Journal of Rock Mechanics and Mining Sciences*, 52:18–30, 2012. [Cited on page 15]

H. Schreyer, D. Sulsky, and S.-J. Zhou. Modeling delamination as a strong discontinuity with the material point method. *Computer Methods in Applied Mechanics and Engineering*, 191(23–24):2483–2507, 2002. [Cited on page 14]

J. Sethian. *Level set methods and fast marching methods: Evolving interfaces in computational geometry, fluid mechanics, computer vision and materials science*. Cambridge University Press, Cambridge, U.K., 1999. [Cited on page 58]

L. Shen. A rate-dependent damage/decohesion model for simulating glass fragmentation under impact using the material point method. *Computer Modeling in Engineering and Sciences*, 49(1):23–45, 2009. [Cited on page 14]

L. Shen and Z. Chen. A multi-scale simulation of tungsten film delamination from silicon substrate. *International Journal of Solids and Structures*, 42(18–19):5036–5056, 2005. [Cited on page 14]

D. Shepard. A two-dimensional function for irregularly spaced points. In *23rd ACM National Conference*, pages 517–524, 1968. [Cited on page 117]

V. Shim, R. Lan, Y. Guo, and L. Yang. Elastic wave propagation in cellular systems–experiments on single rings and ring systems. *International journal of impact engineering*, 34(10):1565–1584, 2007. [Cited on pages 101, 102, and 104]

V.-W. Shim and W. Stronge. Lateral crushing in tightly packed arrays of thin-walled metal tubes. *International Journal of Mechanical Sciences*, 28(10):709–728, 1986. [Cited on pages 104, 105, 106, and 107]

S. A. Silling. Reformulation of elasticity theory for discontinuities and long-range forces. *Journal of the Mechanics and Physics of Solids*, 48(1):175–209, 2000. [Cited on page 66]

J. C. Simo and T. J. R. Hughes. *Computational Inelasticity*. Springer, London, 1998. [Cited on pages 33 and 87]

S. Sinaie, V. P. Nguyen, C. T. Nguyen, and S. Bordas. Programming the material point method in julia. *Advances in Engineering Software*, 105:17–29, Mar 2017. [Cited on pages 19 and 66]

S. Sinaie, T. D. Ngo, and V. P. Nguyen. A discrete element model of concrete for cyclic loading. *Computers & Structures*, 196:173 – 185, 2018a. [Cited on page 15]

S. Sinaie, T. D. Ngo, V. P. Nguyen, and T. Rabczuk. Validation of the material point method for the simulation of thin-walled tubes under lateral compression. *Thin-Walled Structures*, 130:32–46, May 2018b. [Cited on pages 55, 100, 101, 103, and 104]

S. Sinaie, T. D. Ngo, A. Kashani, and A. S. Whittaker. Simulation of cellular structures under large deformations using the material point method. *International Journal of Impact Engineering*, 134:103385, 2019. [Cited on pages 12, 104, and 106]

G. A. Sod. A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws. *Journal of Computational Physics*, 27(1):1 – 31, 1978. [Cited on page 89]

K. Soga, E. Alonso, A. Yerro, K. Kumar, and S. Bandara. Trends in large-deformation analysis of landslide mass movements with particular emphasis on the material point method. *Géotechnique*, 66(3):248–273, 2015. [Cited on page 16]

Y. Song, Y. Liu, and X. Zhang. A transport point method for complex flow problems with free surface. *Computational Particle Mechanics*, Sep 2019. [Cited on page 83]

M. Steffen. *Analysis-Guided Improvements of the Material Point Method (MPM)*. PhD thesis, University of Utah, 2009. [Cited on page 19]

M. Steffen, R. M. Kirby, and M. Berzins. Analysis and reduction of quadrature errors in the material point method (MPM). *International Journal for Numerical Methods in Engineering*, 76(6):922–948, 2008a. doi: 10.1002/nme.2360. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.2360. [Cited on page 6]

M. Steffen, R. M. Kirby, and M. Berzins. Analysis and reduction of quadrature errors in the material point method (MPM). *International Journal for Numerical Methods in Engineering*, 76(6):922–948, 2008b. [Cited on pages 19, 30, and 47]

M. Steffen, P. Wallstedt, J. Guilkey, R. Kirby, and M. Berzins. Examination and analysis of implementation choices within the material point method (MPM). *Computer Modeling in Engineering and Sciences*, 31(2):107–127, 2008c. [Cited on pages 10, 19, 43, 45, and 47]

M. Steffen, R. M. Kirby, and M. Berzins. Decoupling and balancing of space and time errors in the material point method (MPM). *International Journal for Numerical Methods in Engineering*, 82(10):1207–1243, 2010. [Cited on page 19]

A. Stomakhin, C. Schroeder, L. Chai, J. Teran, and A. Selle. A material point method for snow simulation. *ACM Transactions on Graphics*, 32(4):1, 2013. [Cited on pages 32 and 47]

A. Stomakhin, C. Schroeder, C. Jiang, L. Chai, J. Teran, and A. Selle. Augmented MPM for phase-change and varied materials. *ACM Transactions on Graphics*, 33(4):138:1–138:11, 2014a. [Cited on pages 10 and 18]

A. Stomakhin, C. Schroeder, C. Jiang, L. Chai, J. Teran, and A. Selle. Augmented MPM for phase-change and varied materials. *ACM Transactions on Graphics*, 33(4):138:1–138:11, 2014b. [Cited on page 15]

W. G. Strang and G. J. Fix. *An Analysis of the Finite Element Method*. Prentice-Hall, Englewood Cliffs, 1973. [Cited on page 111]

A. Stukowski. Visualization and analysis of atomistic simulation data with ovito–the open visualization tool. *Modelling and Simulation in Materials Science and Engineering*, 18(1):015012, 2009. [Cited on pages 65, 96, 98, 99, and 101]

Y.-C. Su, J. Tao, S. Jiang, Z. Chen, and J.-M. Lu. Study on the fully coupled thermodynamic fluid–structure interaction with the material point method. *Computational Particle Mechanics*, pages 1–16, 2019. [Cited on pages 17 and 80]

N. Sukumar, B. Moran, and T. Belytschko. The natural element method in solid mechanics. *International Journal of Numerical Methods in Engineering*, 43:839–887, 1998. [Cited on page 5]

D. Sulsky and J. Brackbill. A numerical method for suspension flow. *Journal of Computational Physics*, 96(2):339 – 368, 1991. [Cited on page 70]

D. Sulsky and M. Gong. Improving the material-point method. In *Innovative Numerical Approaches for Multi-Field and Multi-Scale Problems*, pages 217–240. Springer, 2016. [Cited on pages 6, 11, 19, 43, 82, 83, 96, and 108]

D. Sulsky and A. Kaul. Implicit dynamics in the material-point method. *Computer Methods in Applied Mechanics and Engineering*, 193(12-14):1137–1170, 2004. [Cited on pages 10, 11, and 125]

D. Sulsky and H. Schreyer. Axisymmetric form of the material point method with applications to upsetting and Taylor impact problems. *Computer Methods in Applied Mechanics and Engineering*, 139:409–429, 1996. [Cited on pages 6, 97, 98, 124, and 125]

D. Sulsky and L. Schreyer. MPM simulation of dynamic material failure with a decohesion constitutive model. *European Journal of Mechanics - A/Solids*, 23(3):423 – 445, 2004. [Cited on page 14]

D. Sulsky, Z. Chen, and H. Schreyer. A particle method for history-dependent materials. *Computer Methods in Applied Mechanics and Engineering*, 5:179–196, 1994. [Cited on pages 5, 6, 7, 27, 29, 86, and 87]

D. Sulsky, S. Zhou, and H. L. Schreyer. Application of a particle-in-cell method to solid mechanics. *Computer Physics Communications*, 87(1-2):236–252, 1995. [Cited on pages 6, 34, 35, 36, and 89]

D. Sulsky, H. Schreyer, K. Peterson, R. Kwok, and M. Coon. Using the material-point method to model sea ice dynamics. *Journal of Geophysical Research*, 112(C2):C02S90, 2007. [Cited on page 19]

L. Sun, S. R. Mathur, and J. Y. Murthy. An unstructured finite-volume method for incompressible flows with complex immersed boundaries. *Numerical Heat Transfer, Part B: Fundamentals*, 58(4):217–241, 2010. [Cited on page 17]

Z. Sun, H. Li, Y. Gan, H. Liu, Z. Huang, and L. He. Material point method and smoothed particle hydrodynamics simulations of fluid flow problems: a comparative study. *Progress in Computational Fluid Dynamics, An International Journal (PCFD)*, 18(1):1–18, 2018. [Cited on pages 15 and 86]

Z. Sun, Z. Huang, and X. Zhou. Benchmarking the material point method for interaction problems between the free surface flow and elastic structure. *Progress in Computational Fluid Dynamics, an International Journal*, 19(1):1–11, 2019. [Cited on page 17]

H. Tan and J. A. Nairn. Hierarchical, adaptive, material point method for dynamic energy release rate calculations. *Computer Methods in Applied Mechanics and Engineering*, 191(19-20):2123–2137, 2002. [Cited on pages 13 and 85]

J. Tao, Y. Zheng, Z. Chen, and H. Zhang. Generalized interpolation material point method for coupled thermo-mechanical processes. *International Journal of Mechanics and Materials in Design*, 12(4):577–595, 2016. [Cited on pages 12, 76, 77, and 79]

J. Tao, H. Zhang, Y. Zheng, and Z. Chen. Development of generalized interpolation material point method for simulating fully coupled thermomechanical failure evolution. *Computer Methods in Applied Mechanics and Engineering*, 332:325–342, 2018. [Cited on page 12]

R. Tielen, E. Wobbes, M. Möller, and L. Beuth. A high order material point method. *Procedia Engineering*, 175:265–272, 2017. [Cited on page 10]

L. T. Tran, J. Kim, and M. Berzins. Solving time-dependent pdes using the material point method, a case study from gas dynamics. *International Journal for Numerical Methods in Fluids*, 62(7):709–732, 2010. [Cited on pages 14 and 19]

Q.-A. Tran and W. Sołowski. Temporal and null-space filter for the material point method. *International Journal for Numerical Methods in Engineering*, 2019. [Cited on page 11]

Q.-A. Tran, M. Berzins, and W. Sołowski. An improved moving least squares method for the material point method. In *2nd International Conference on the Material Point Method for Modelling Soil-Water-Structure Interaction*, 2019. [Cited on page 85]

M. Vargas, E. Nascimento, G. Nascimento, M. Hotta, and M. Almeida. Comparative study of the material point method and smoothed particle hydrodynamics applied to the numerical simulation of a dam-break flow in the presence of geometric obstacles. *Current Journal of Applied Science and Technology*, 2018. [Cited on pages 15 and 86]

P. Wallstedt and J. Guilkey. An evaluation of explicit time integration schemes for use with the generalized interpolation material point method. *Journal of Computational Physics*, 227(22):9628 – 9642, 2008. [Cited on pages 19, 35, 36, 90, and 109]

P. C. Wallstedt and J. E. Guilkey. Improved velocity projection for the material point method. *Computer Modeling in Engineering & Science*, 19(3):223–232, 2007. [Cited on page 82]

P. C. Wallstedt and J. E. Guilkey. A weighted least squares particle-in-cell method for solid mechanics. *International Journal for Numerical Methods in Engineering*, 85(13):1687–1704, 2011. [Cited on page 11]

B. Wang, V. Karuppiah, H. Lu, R. Komanduri, and S. Roy. Two-Dimensional Mixed Mode Crack Simulation Using the Material Point Method. *Mechanics of Advanced Materials and Structures*, 12(6):471–484, 2005. [Cited on pages 13 and 64]

B. Wang, P. J. Vardon, M. A. Hicks, and Z. Chen. Development of an implicit material point method for geotechnical applications. *Computers and Geotechnics*, 71:159 – 167, 2016. [Cited on page 11]

L. Wang, W. Coombs, C. Augarde, M. Cortis, T. Charlton, M. Brown, J. Knappett, A. Brennan, C. Davidson, D. Richards, et al. On the use of domain-based material point methods for problems involving large distortion. *Computer Methods in Applied Mechanics and Engineering*, 355:1003–1025, 2019. [Cited on pages 10 and 94]

Y. Wang, H. Beom, M. Sun, and S. Lin. Numerical simulation of explosive welding using the material point method. *International Journal of Impact Engineering*, 38(1):51–60, 2011. [Cited on page 19]

C. Weißenfels and P. Wriggers. Stabilization algorithm for the optimal transportation meshfree approximation scheme. *Computer Methods in Applied Mechanics and Engineering*, 329:421–443, 2018. [Cited on page 15]

Z. Wieçkowski. The material point method in large strain engineering problems. *Computer Methods in Applied Mechanics and Engineering*, 193(39-41):4417–4438, 2004. [Cited on pages 11, 16, 60, and 64]

Z. Wieçkowski, S.-k. Y., and J.-h. Y. A Particle-in-cell solution to the silo discharging problem. *International Journal For Numerical Methods in Engineering*, 45:1203–1225, 1999. [Cited on pages 11, 12, 16, and 64]

M. L. Wilkins. *Computer Simulation of Dynamic Phenomena*. Springer Berlin Heidelberg, 1999. ISBN 3540630708. URL https://www.ebook.de/de/product/6699393/mark_l_wilkins_computer_simulation_of_dynamic_phenomena.html. [Cited on page 113]

M. L. Wilkins and M. W. Guinan. Impact of cylinders on a rigid boundary. *Journal of Applied Physics*, 44(3):1200–1206, 1973. [Cited on page 97]

E. Wobbes, M. Möller, V. Galavi, and C. Vuik. Conservative taylor least squares reconstruction with application to material point methods. *International Journal for Numerical Methods in Engineering*, 117(3):271–290, 2019. [Cited on pages 6, 11, and 82]

J. Wolper, Y. Fang, M. Li, J. Lu, M. Gao, and C. Jiang. Cd-mpm: continuum damage material point methods for dynamic fracture animation. *ACM Transactions on Graphics (TOG)*, 38(4):119, 2019. [Cited on pages 14 and 18]

B. Wu, Z. Chen, X. Zhang, Y. Liu, and Y. Lian. Coupled shell-material point method for bird strike simulation. *Acta Mechanica Solida Sinica*, 31(1):1–18, 2018. [Cited on page 17]

J. Y. Wu. A unified phase-field theory for the mechanics of damage and quasi-brittle failure in solids. *Journal of the Mechanics and Physics of Solids*, 103:72–99, 2017. [Cited on page 14]

J. Y. Wu, V. P. Nguyen, C. T. Nguyen, D. Sutula, S. Sinaie, and S. Bordas. Phase field modeling of fracture. *Advances in Applied Mechancis: Fracture Mechanics: Recent Developments and Trends*, 53:submitted, 2019. [Cited on page 14]

S. Wu and L. Wu. *Introduction to the Explicit Finite Element Method for Nonlinear Transient Dynamics*. John Wiley and Sons, 2012. [Cited on page 121]

X. Xiang, G. Lu, Z.-X. Li, and Y. Lv. Large deformation of tubes under oblique lateral crushing. *International Journal of Impact Engineering*, 2017. [Cited on pages 101, 102, and 103]

S. Xu, D. Ruan, G. Lu, and T. Yu. Collision and rebounding of circular rings on rigid target. *International Journal of Impact Engineering*, 79:14–21, 2015. [Cited on pages 101, 102, and 104]

L. Xue, O. Borodin, and G. D. Smith. Modeling of enhanced penetrant diffusion in nanoparticle-polymer composite membranes. *Journal of Membrane Science*, 286(1–2):293 – 300, 2006a. [Cited on page 18]

L. Xue, O. Borodin, G. D. Smith, and J. Nairn. Micromechanics simulations of the viscoelastic properties of highly filled composites by the material point method (MPM). *Modelling and Simulation in Materials Science and Engineering*, 14(4): 703, 2006b. [Cited on page 18]

P. Yang, Y. Liu, X. Zhang, X. Zhou, and Y. Zhao. Simulation of fragmentation with material point method based on Gurson model and random failure. *Computer Modeling in Engineering and Sciences*, 85(3):207–236, 2012. [Cited on page 14]

P. Yang, Y. Gan, X. Zhang, Z. Chen, W. Qi, and P. Liu. Improved decohesion modeling with the material point method for simulating crack evolution. *International Journal of Fracture*, 186(1-2):177–184, 2014. [Cited on pages 13 and 14]

W.-C. Yang, P. Arduino, G. R. Miller, and P. Mackenzie-Helnwein. Smoothing algorithm for stabilization of the material point method for fluid–solid interaction problems. *Computer Methods in Applied Mechanics and Engineering*, 342:177–199, 2018. [Cited on pages 17, 74, and 80]

Z. Ye, X. Zhang, G. Zheng, and G. Jia. A material point method model and ballistic limit equation for hyper velocity impact of multi-layer fabric coated aluminum plate. *International Journal of Mechanics and Materials in Design*, 14(4):511–526, 2018. [Cited on page 19]

A. Yerro, E. Alonso, and N. Pinyol. The material point method for unsaturated soils. *Géotechnique*, 65:201–217(16), 2015. [Cited on page 16]

A. Yerro, K. Soga, and J. Bray. Runout evaluation of oso landslide with the material point method. *Canadian Geotechnical Journal*, (999):1–14, 2018. [Cited on page 16]

A. York, D. Sulsky, and H. Schreyer. The material point method for simulation of thin membranes. *International Journal for Numerical Methods in Engineering*, 44(10):1429–1456, 1999. [Cited on pages 9, 12, 14, 17, and 80]

A. York, D. Sulsky, and H. Schreyer. Fluid-membrane interaction based on the material point method. *International Journal for Numerical Methods in Engineering*, pages 901–924, 2000. [Cited on pages 12, 14, 17, and 80]

A. R. York. *Development of Modifications to the Material Point Method for the Simulation of Thin Membranes, Compressible Fluids, and Their Interactions*. PhD thesis, The University of New Mexico, Albuquerque, 1997. [Cited on pages 19, 68, and 70]

Y. Yue, B. Smith, C. Batty, C. Zheng, and E. Grinspun. Continuum foam: A material point method for shear-dependent flows. *ACM Transactions on Graphics*, 2015. [Cited on page 18]

D. Z. Zhang, X. Ma, and P. T. Giguere. Material point method enhanced by modified gradient of shape function. *Journal of Computational Physics*, 230(16):6379 – 6398, 2011. [Cited on pages 11, 41, and 65]

F. Zhang, X. Zhang, K. Y. Sze, Y. Lian, and Y. Liu. Incompressible material point method for free surface flow. *Journal of Computational Physics*, 330:92–110, 2017. [Cited on pages 15 and 80]

H. Zhang, K. Wang, and Z. Chen. Material point method for dynamic analysis of saturated porous media under external contact/impact of solid bodies. *Computer Methods in Applied Mechanics and Engineering*, 198(17-20):1456–1472, 2009. [Cited on page 16]

X. Zhang, K. Sze, and S. Ma. An explicit material point finite element method for hyper-velocity impact. *International Journal for Numerical Methods in Engineering*, 66(4):689–706, 2006. [Cited on page 16]

X. Zhang, Z. Chen, and Y. Liu. *The material point method: a continuum-based particle method for extreme loading cases*. Academic Press, 2016. [Cited on pages 6, 16, 19, 27, and 80]

X. Zhao, D. Liang, and M. Martinelli. Mpm simulations of dam-break floods. *Journal of Hydrodynamics*, 29(3):397–404, 2017. [Cited on pages 15 and 86]

Y. Zheng, F. Gao, H. Zhang, and M. Lu. Improved convected particle domain interpolation method for coupled dynamic analysis of fully saturated porous media involving large deformation. *Computer Methods in Applied Mechanics and Engineering*, 257(0):150 – 163, 2013. [Cited on page 16]

S. Zhou, J. Stormont, and Z. Chen. Simulation of geomembrane response to settlement in landfills by using the material point method. *International Journal for Numerical and Analytical Methods in Geomechanics*, 23(15):1977–1994, 1999. [Cited on page 16]

S. Zhou, X. Zhang, and H. Ma. Numerical simulation of human head impact using the material point method. *International Journal of Computational Methods*, 10(04):1350014, 2013. [Cited on page 19]

Y. Zhu and R. Bridson. Animating sand as a fluid. *ACM Transactions on Graphics*, 24(3):965–972, 2005. [Cited on page 6]

O. C. Zienkiewicz and R. L. Taylor. *The Finite Element Method for Solid and Structural Mechanics*. Butterworth-Heinemann, Oxford, UK, sixth edition, 2006. [Cited on page 30]