DISS. ETH NO. 25679

MESHLESS SOFTWARE TOOL TO SIMULATE METAL CUTTING OPERATIONS BY EMPLOYING CONTEMPORARY NUMERICAL METHODS

A dissertation submitted to
ETH ZURICH

for the degree of
DOCTOR OF SCIENCES

presented by
MATTHIAS RÖTHLIN
Master of Science in Computer Science (ETH Zurich)
born 25 January 1986
citizen of Kerns, Switzerland

accepted on the recommendation of
Prof. Dr. Konrad Wegener, examiner
Prof. Dr. Pavel Hora, co-examiner

2019

ABSTRACT

The economic importance of metal cutting operations is hard to overstate. Optimizing and understanding metal cutting processes is thus of great interest, both academically and in industry. Simulation can be a viable tool to both of these ends.

However, the metal cutting process remains notoriously hard to simulate, exhibiting large deformations, and high deformation- and temperature rates. Especially conventional simulation techniques like the FEM are affected by these characteristics, necessitating computationally very expensive techniques like remeshing or ALE formulations.

Meshless methods offer an interesting alternative to FEM, since they are conceptually not limited in the amount of deformation they can reproduce. While some meshless solvers capable of running metal cutting simulations exist in some commercial solver packages, they are quite limited in both the numerical techniques and physics modeling they offer.

This thesis is thus dedicated to developing a new meshless simulation tool for metal cutting. A wide range of meshless algorithms is explored, and the most stable and efficient techniques are selected by means of rigorous benchmark simulations. Care is taken to develop a complete physical model, equivalent to the state of the art in metal cutting simulations. The simulations remain computationally affordable by employing general purpose computing on the graphics processor, a technique that has been hardly explored so far for meshless methods for solid mechanics.

This work is completed by a wide array of metal cutting simulations, both orthogonal and in full 3D space, using the especially challenging situation of single grain cutting.

ZUSAMMENFASSUNG

Der Zerspanungsprozess bleibt einer der wichtigsten Fertzigungsprozesse in den industrialisierten Ländern. Diesen Prozess zu verstehen und zu optimieren ist also von grösstem Interesse. Simulation kann ein wichtigstes Werkzeug zu diesem Zweck sein.

Allerdings bleibt der Zerspanungsprozess nach wie vor besonders schwer zu simulieren, zumal er grosse Deformationen und hohe Deformations- und Temperaturraten aufweist. Klassische Simulationstechniken wie die FEM sind von diesen Eigenschaften besonders schwer betroffen. Damit diese den Zerspanungsprozess simulieren können sind numerisch sehr teure Massnahmen wie die Neuvernetzung oder ALE Formulierungen nötig.

Netzfreie Methoden bieten hier eine interessante Alternative zu FEM, zumal diese nicht limitiert sind in den Deformationen die sie auflösen können. Es gibt einige kommerzielle Solverpakete die die Simulation von Zerspanung mit netzfreien Methoden ermöglichen, jedoch sind diese in den angebotenen numerischen Methoden sowie in der physikalischen Modellierung stark eingeschränkt.

In dieser Arbeit wird deshalb ein neues Software-Tool entwickelt zur netzfreien Zerspanungssimulation. Eine breite Auswahl an netzfreien Algorithmen wird untersucht. Die stabilsten und effizientesten Techniken daraus werden anhand von herausfordernden Testrechnungen identifiziert. Dabei wird auf ein vollständiges physikalisches Modell geachtet, das dem Stand der Technik entspricht. Die Simulationen erhalten moderate Laufzeiten durch Paralellisierung auf der Graphikkarte, eine Technik die bisher kaum für netzfreihe Methoden in der Solidmechanik Anwendung fand.

Diese Arbeit wird komplettiert durch umfangreiche Zerspanungssimulationen, sowohl im Orthogonalschnitt wie auch in vollem 3D anhand von Einzelkornschnittversuchen, welche als besonders herausfordernd gelten.

PUBLICATIONS

The following publications are included in parts or in an extended version in this thesis:

- Röthlin Matthias, Hagen Klippel, and Konrad Wegener. "Meshless Methods for Large Deformation Elastodynamics", In: *arXiv preprint arXiv:1807.01117* (2018).
- Afrasiabi Mohamadreza, Röthlin Matthias, and Konrad Wegener. "Contemporary Meshfree Methods for Three Dimensional Heat Conduction Problems", Under peer review in: *Mathematics and Computers in Simulation* (2018).
- Afrasiabi Mohamadreza, Röthlin Matthias, Hagen Klippel, Eleni Chatzi, and Konrad Wegener. "Beyond FEM: Meshfree Simulations of Manufacturing Processes on GPU", In: *ECCOMAS Newsletter* (2018).
- Röthlin Matthias, Hagen Klippel, Afrasiabi, Mohamadreza, and Konrad Wegener. "Meshless metal cutting simulations on the GPU", Under peer review in: *Journal of Advanced Manufacturing Technology* (2018).
- Röthlin Matthias, Hagen Klippel, Afrasiabi Mohamadreza, and Konrad Wegener. "Design and Implementation of a Meshless, Updated Lagrangian, Orthogonal Metal Cutting Solver", Under peer review in: *CIRP Journal of Manufacturing Science and Technology* (2018).
- Röthlin, Matthias, Hagen Klippel, Afrasiabi Mohamadreza, and Konrad Wegener. "Meshless metal cutting in 3D on the GPU", In preparation, (2019).

Furthermore, the following publication was part of my PhD research, is however not covered in this thesis. The topic of this publication is outside of the scope of the material covered here:

- Afrasiabi Mohamadreza, Röthlin Matthias, and Konrad Wegener. "A Robust Particle-Based Solver for Modeling Heat Transfer in Multiphase Flows", In: *ECCM-ECFD, UK* (2018).

ACKNOWLEDGMENTS

This thesis is the result of my work as research assistant at the Institute of Machine Tools and Manufacturing (IWF) at ETHZ.

First of all, I would like to express my gratitude to my supervisor Prof. Dr. Konrad Wegener, director of the IWF. I thank him for the trust he put in me, the freedom he gave me in conducting my research, the many discussions and his always rigorous and profound feedback.

I'd also like to thank Prof. Dr. Pavel Hora for co-examining and reviewing this thesis.

Special thanks to Dr. Niklaus Rüttimann for his guidance during my first months at IWF as well as his help in the preparation for the SNF fund request that made this research possible. In this regard I'd also like to extend my gratitude to Dr. Fredy Kuster.

Furthermore I'd also like to thank my collaborators Hagen Klippel and Mohamadreza "Mamzi" Afrasiabi, whose contributions are an integral part of this work. I'd also like to thank my colleagues Andreas Zschippang and Marcel Gerstgrasser who joined countless high performance meetings, sometimes even late into the night.

Last but not least I'd also like to extend my special gratitude to my family, especially for their support in the early years of my studies, and to my group of friends that kept me sane. Finally, my deepest gratitude goes to Jenny Betschart.

CONTENTS

1

INTRODUCTION

It is hard to overstate the importance of the metal cutting process to the economy in any developed country. It is by far the most widely used process in manufacturing of mechanical products. In fact, cost of machining amounts to more than 15% of the total value of all products manufactured, mechanical or not, see [161]. In the United States, according to [235] the cost of removing material in manufacturing exceeded 10% of the gross national product and, for a wide range of products, the cost of metal cutting dictates a 40%-60% fraction of the price [242].

Due to its tremendeous economical impact, the process came under scrutiny of the scientific community in the early 20th century in the landmark publication by Taylor [258]. Since then, countless experimental studies were undertaken and numerous empirical models with regard to cutting forces, tool wear and other quantities have been proposed. With the exponential growth of computational resources due to Moores law [173], simulating the metal cutting process on a computer came in the realm of possibility in the late 20th century. The aim of studying the metal cutting process, no matter whether the study is experimental, empirical or by means of computer simulation is twofold: To gain insight into the physical process on one hand and to improve process parameters on the other. The nature of the metal cutting process makes it a prime candidate for computer simulations:

- Experiments are hard to design since the most important areas of the metal cutting process, i.e. the area of material separation is not visible from the outside. Measurements are hard to take and often done indirectly, afflicting them with large measurement uncertainties. Some processes like those inside the chip are out of reach to experimental inspection entirely

- The parameter space is extremely large. Just the most basic tool parameters include rake- and clearance angle, chamfer radius, cutting speed and depth. This ignores any chip breaker geometry or the use of cutting fluids. Exhausting the parameter space experimentally in order to design optimal process parameters becomes thus prohibitively expensive.

At the same time, the metal cutting process is challenging to simulate:

- Strains are extremely large, exceeding 700% with strain rates up to $10^6/s$ [9].

- Large temperature gradients impact the material behavior and friction, rendering isothermal models inappropriate for anything but qualitative simulations

- Material separation not only dictates large strains but also topological changes to the workpiece

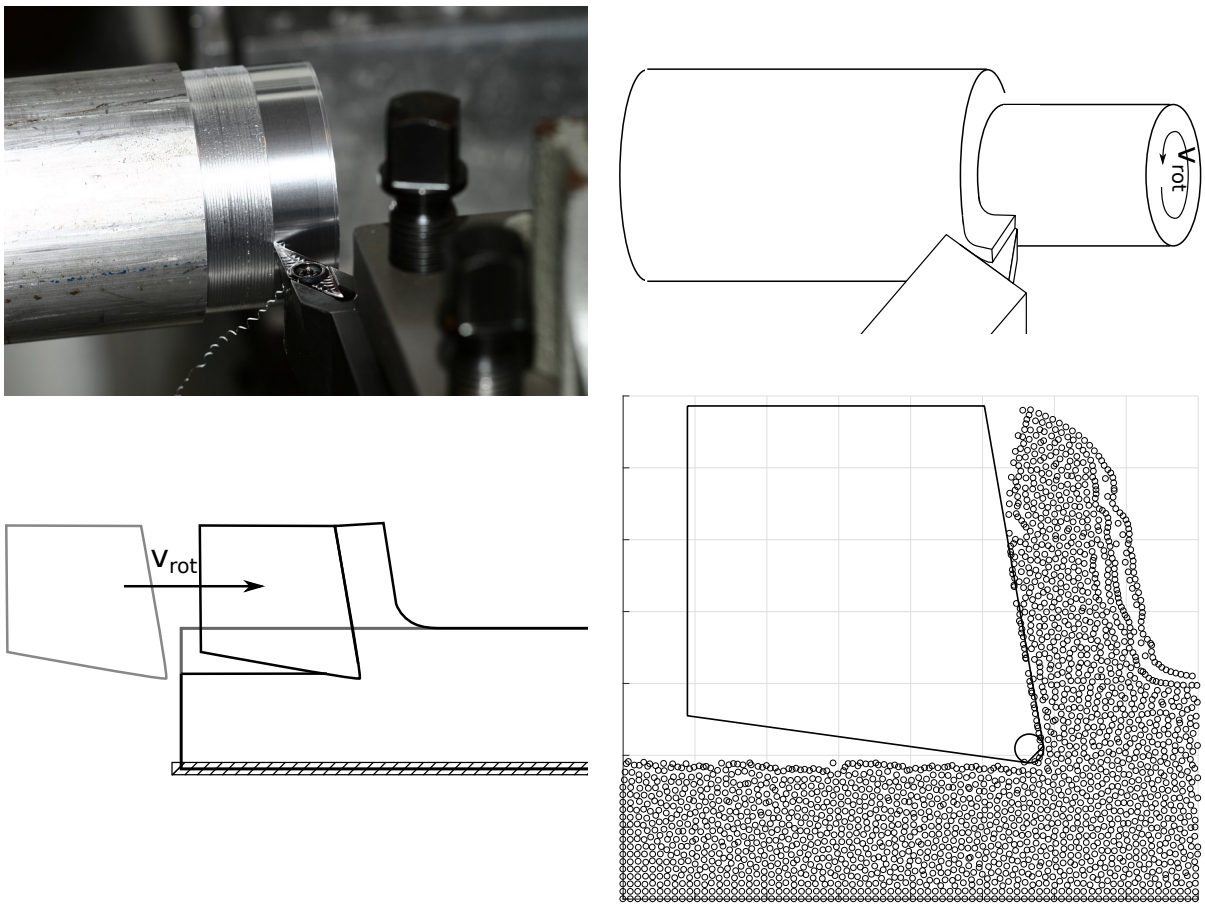

Figure 1.1: Picture of a finishing operation in orthogonal turning, schematic view, 2 dimensional simplification and a low resolution meshfree metal simulation result using the 2 dimensional simplification, left to right, top to bottom. Top left picture by Florian Schott, `https://upload.wikimedia.org/wikipedia/commons/9/90/SchlichtenDrehen.jpg`, `CC BY-SA 4.0`

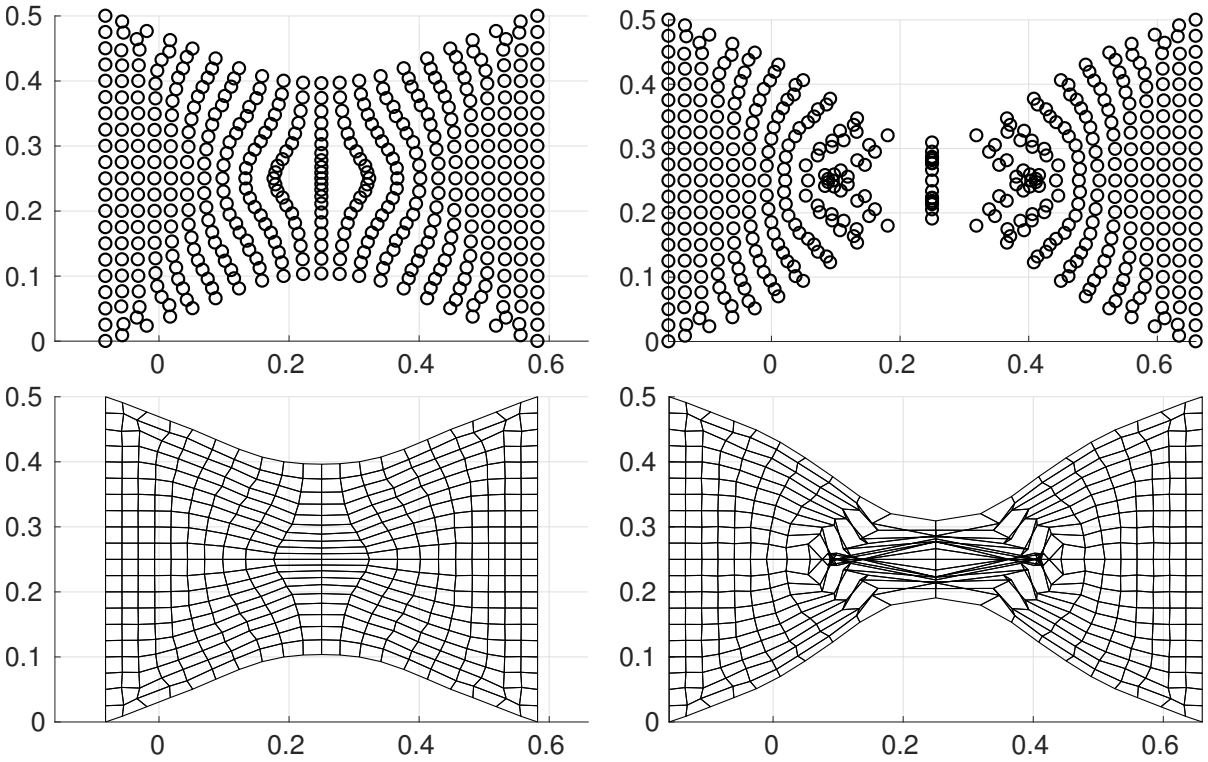

Figure 1.2: A tensile test is performed using a meshfree method [217]. The resulting particle distribution at two timesteps is shown at the top. The bottom shows a mesh as if it would have deformed by the particle motion due to the meshless method. The earlier, left frame shows some elements with bad aspect ratios, hampering the accuracy of the simulation at hand. The right, later, frame shows inverted elements, which would lead to failure of the simulation in FEM.

- The process is inherently three dimensional. Only the very specific case of orthogonal cutting can be modeled in two dimensions, and even then the necessary assumptions are questionable at best.

While the tool of choice to simulate the metal cutting process certainly is the Finite Element Method (FEM), some of these challenges can be addressed by the use of meshfree methods. While the FEM is limited in the amount of deformation it can represent before costly remeshing operations are necessary, meshless methods do not maintain a global connectivity structure but establish a neighbor set in each time step. Figure 1.2 illustrates this topic. Figure 1.1 shows a real life picture of a metal cutting operation as well as some schematic views and an exemplary result frame from a meshless metal cutting simulation.

The idea to apply meshless methods to metal cutting is not new. One of the most well known methods, Smoothed Particle Hydrodynamics (SPH) was applied in [104], and has seen occasional use in metal cutting simulations since, especially afters it was implemented into some commercial solver packages.

While meshless methods are conceptually not limited in the amount of deformation they can represent they are not without their own share of problems. Various instability modes are known, especially with regard to SPH [256]. A host of approaches to counteract these

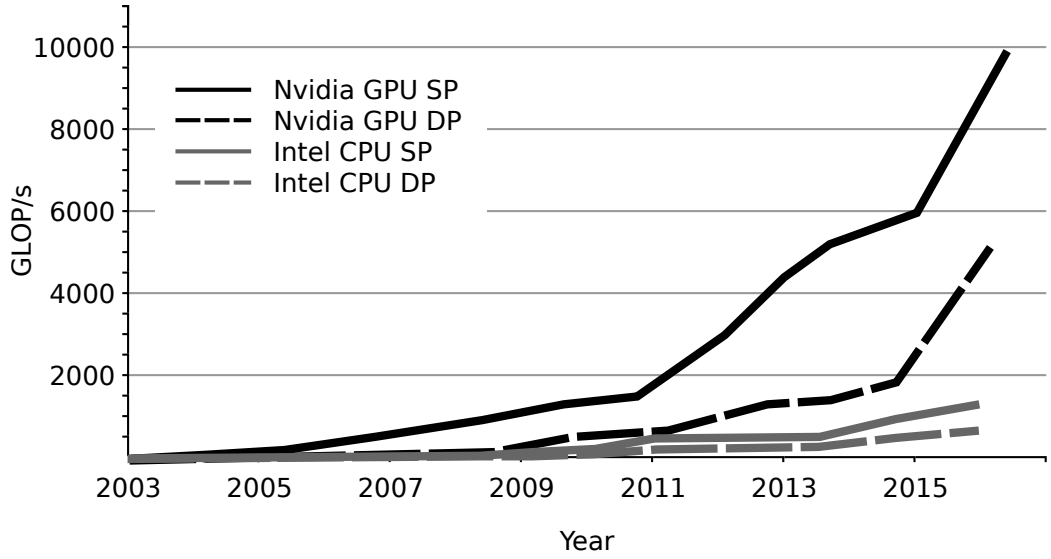

Figure 1.3: Theoretical peak billion Floating Point Operations (GFLOPS) per second for both intel as well as nVidia GPU chips, according to [186], both for Double Precision (DP) and Single Precision (SP).

instability modes exist, but they are rarely used or ignored completely in conjunction with metal cutting, possibly due to their absence from said commercial solver packages.

However, only focusing on the algorithmic aspect, for instance replacing the FEM by the SPH ignores half of the aspect of computational science. The other half, so to speak, is to ensure that the algorithm was implemented in such a fashion that the underlying hardware is exploited as much as possible. This may range from vectorizing code for single core applications, over parallelizing the code for shared memory machines using e.g. `openMP` up to the very challenging task of designing work loads on the fly for heterogenous clusters / super computers. In the latter case this is not only an issue of enabling shorter production cycles, but may become an ecological issue too: top of the line super computers use 4 to 6 megawatts, which is equal to about 5000 households [131].

One class of processors, the Graphics Processing Unit has seen especially impressive increase of computing power in the last decade, see figure 1.3. However, not every algorithm is suitable for the Single Instruction Multiple Data (SIMD) parallelism appropriate for GPU processing. It has been shown that the SPH parallelizes efficiently for Computational Fluid Dynamics (CFD) applications [53], but with the exception of [80] no GPU accelerated SPH codes exist for solid mechanics. Hence, this thesis will focus on combining efficient, contemporary meshless algorithms with techniques to exploit the computational power of GPUs. Some of the key contributions of this thesis can be summarized as:

- A wide number of meshless algorithms is implemented. Some of them are subjected to metal cutting simulations for the first time. This includes a proof of concept for the first ever total Lagrangian metal cutting simulations.
- It is demonstrated how GPUs can be leveraged to reduce the time to solution of low resolution metal cutting simulations by two orders of magnitude over conventional means

- It is shown how short production cycles can be employed to optimize the cost of an orthogonal metal cutting process
- Again employing GPUs the, to the best of the authors knowledge, most high resolution meshless orthogonal cutting simulations to date are presented
- Finally, the implementation of a metal cutting solver in 3D with some unique features is described, including first order completeness and correction by Transport Velocity Formulation [282] or Artifical Stresses [91]. Some of these simulations run at unprecedented resolutions.

2

STATE OF THE ART

This section provides an overview over some notable works in all topics covered by this thesis. Some cornerstones of meshless methods are covered first. Additional references with regard to the specific meshless methods employed in this thesis are also given in sections 4.1, 4.3 where they are properly introduced. A history of GPGPU computing follows. Afterwards, an overview over metal cutting simulations in general and meshless metal cutting simulations in particular is provided. A research gap is identified and objectives based on this research gap are derived.

2.1 MESHLESS METHODS

There is no universally accepted definition of what a meshless method exactly is. If the very general definition that a meshless method is simply a numerical method that simulates a physical system without maintaining element connectivity during simulation time is assumed, the very first meshless methods are probably Monte Carlo type simulations, which have been used as early as 1946 to simulate phenomena on the atomic scale, see [129] and the references therein. Another contestant for very early simulations without meshes are molecular dynamics applications. They have been around from the early sixties [86] [212], or arguably even late fifties [4].

However, nowadays a slightly different definition is more common, i.e. that the discretization points of the numerical method are not only devoid of a global mesh structure, but also discretize the continuum in the same fashion Finite Element Methods (FEM) do. This is not the case both for Monte Carlo methods, where the elements are simply query points which are rejection sampled, as well as molecular dynamics where each element is a discrete molecule. The first method conforming to this definition is Smoothed Particle Hydrodynamics (SPH) by Lucy [154] and Gingold and Monaghan [87]. Originally the method was incepted for celestial dynamics but quickly found adoption in a variety of fields. [171] provides an overview of a lot of applications while [205] presents an excellent review of the SPH theory.

The SPH method has been subject to constant development. The most important developments with regard to this thesis are establishing higher order kernels using some correction functions. These include the Randles Libserky correction [143] and the Reproducing Kernel Particle Method (RKPM) [151]. A host of other correctors of the same fashion exist. A comprehensive list can be found in [140].

The SPH traditionally operates on the strong form of the Partial Differential Equations (PDE). A different class of meshfree methods can be established by considering the weak form instead, chose the same variational approach as in the development of the FEM but give up the locality of the shape function, i.e. allow the shape function to span multiple elements.

This leads to the Diffuse Element Method [180]. A variation thereof which mostly differs in the approximation scheme with regard to spatial derivatives is the well known Element Free Galerkin (EFG) Method [22]. It is debatable to what extent these methods conform to the definition mentioned, since they rely on a weak coupling to a background mesh to perform the numerical quadrature of the weak form. However, if a total Lagrangian formalism is chosen, this coupling may be a mute point since the mesh can be established once in the beginning of the simulation and does not need to be maintained, or even remeshed, during simulation.

It is worth noting that variations of the method were also explored by the computer graphics community, enabling real time simulations or at least very fast simulation cycles for animation purposes. A few results include a technique to reduce the order of the PDEs associated with solid dynamics by one [178] or the use of a co-rotational variant of the SPH based on a shape matching procedure [19].

A third path leading to meshless methods stems from the advances in Particle In Cell (PIC) [102] and FLuid Implicit Particle Method (FLIP) [28] schemes, which are considered meshless by some researchers themselves. FLIP was developed into the Material Point Method (MPM) in 1994 in [253]. Again, a background mesh is employed, but the mesh is constant in time, even in updated Lagrangian approaches. This means that the simulation domain has to be known beforehand, but other than that the method can reasonably coined meshless. The method has seen surge in popularity since the publication of [251].

This list of meshless methods is far from exhaustive. More complete reviews of meshless methods, their derivations and comparison of approximation theory can be found in [24] and more recently [39]. An impressive list of application is found in [84]. Computer Implementation aspects are also discussed in [181].

2.2 GPGPU COMPUTING

The Graphics Processing Unit (GPU) refers to special purpose circuitry intended to accelerate graphics processing operations. Originally, the programs to be executed on the GPU, called shaders, were quite rigid and could hardly be altered. This changed with the advent of programmable shaders in the early 2000s. While intended for graphics effects, e.g. *shading* a triangle with a reflection of its surroundings to simulate a mirror, these small programs could potentially perform any computation; hence the term General Purpose Graphics Computing (GPGPU). Originally, these programs were very limited, i.e. the first programmable graphics chip allowed for programs of a length of 128 opcodes only, and programming was very awkward since programs had to be mapped to the graphics pipeline, necessitating tricks like, assuming a 2 dimensional fluid simulation, mapping the pressure field to the blue color channel of a texture, then applying a fragment shader that would put the resulting velocity field given said pressure in the red and green channels. None the less, a matrix-matrix multiplication was presented in 2001 [138] using the extremely limited graphics hardware available at the time. This was extended to LU-factorization in [83], outperforming the CPUs available at the time. Even a complete but rudimentary

implementation of SPH for fluid dynamics exists using these principles, achieving moderate speedups [5]. Since a satisfactory overview over early graphics processing could not be identified in the scientific literature, the interested reader is referred to the overview over the history of programmable shaders by the Khronos Group, https://www.khronos.org/opengl/wiki/History_of_Programmability, retrieved 30.06.2018.

However, GPGPU computing entered the focus of the scientific community with the introduction of CUDA, a proprietary technology by Nvidia enabling GPUs of that company to be programmed by some high level languages such as C. This significantly lowered the learning curve and allowed to express programs in a fashion more natural to the problem domain. Nowadays, the applications are too numerous to exhaustively list. Some works relevant to this thesis include the efforts to parallelize either the FEM sparse matrix operations only, as presented in [88], or the full FEM as in [259], [101] or [48]. The latter works are all by the virtual surgery community and focused on the simulation of soft tissues.

In 2008, it was shown how to efficiently perform short range particle interactions, as encountered by SPH and most meshfree methods on the GPU in [93]. Note that the manuscript was indeed originally published in 2008, however, the updated version from 2010 is referenced here, hence the mismatch in dates. This gave way to a host of different SPH implementations for fluid dynamics on the GPU. Most notably dualsphysics [54], which claims to be industry ready to simulate coastal safety structures. Another implementation mostly tailored to the computer graphics community can be found in [90].

2.3 METAL CUTTING SIMULATIONS

Even though the metal cutting process is notoriously hard to simulate, efforts to this end seem to have started as early as the late seventies [240]. Unfortunately, attempts to retrieve this article proved futile. However, other well cited works [164] and [165] claim that this publication exists and is indeed the first effort to simulate metal cutting. Apparently, Eulerian meshes were employed to circumvent expensive remeshing operations which were not very well developed at the time, and prohibitively numerically expensive. An even earlier work but limited to thermal effects only with a predetermined chip path is [257]. While the Eulerian approach to metal cutting has seemingly fallen out of favor, there are still occasionally papers published using that principle, e.g. [252]. It is speculated that the approach was mostly abandoned because it can only resolve the chip shape up to the mesh resolution.

Another approach are remeshed Lagrangian FEM simulations. Here, the mesh deforms with the displacement field. Since very bad aspects ratio or even volume inversion may, and in the case of metal cutting simulations most definitely will, occur, these simulations are hampered in their effectiveness by requiring expensive remeshing operations. Nonetheless, this seems to be the most used paradigm in metal cutting simulations. One reason for this might be that the special purpose metal cuttting simulation tools DEFORM and AdvantEdge employ this method. Probably the earliest Lagrangian FEM metal cutting simulations are presented in [45]. These simulations were achieved with a serious limitation: the mesh was designed with a predetermined seperation path. The material separation point can not be determined trivially and should be a simulation output, not input.

One of the first metal cutting simulations without this limitation is given in [232]. Shortly after this fact, high speed cutting was presented in [159]. Other authors opted to simulate metal cutting in the steady state [126] or focused on visco-plastic effects [65]. An overview over these "early years" of metal cutting can be found in [155], mentioning hundreds of works.

Later on, additional computing power allowed to expand the modeling, i.e. by considering damage, even in three dimensions [198]. Study of other process variables than the most basic ones, i.e. the cutting forces and chip shape became possible, as demonstrated by the residual stress analysis presented in [238]. Other researchers chose to employ the now available computational power to perform parameter studies, e.g. with regard to friction, see [239], [195], or the rake angle [237]. An overview of developments up until 2013 can be found in [9]. Three dimensional cutting simulations are quite rare and only start to emerge recently. Two works include [278] for milling and [3] for single grain cutting operations.

Even though the Lagrangian FEM approach is the predominant one it is not without its criticisms. In [27] it is shown that the general purpose solver `Msc.Marc` and the special purpose solvers `Deform2D` and `AdvantEdge` produce widely different results under the same inputs and may not only underpredict the thrust force, but even predict thrust forces pointing into the wrong direction. This prompts the question if other approaches might be more suitable for the task at hand. Indeed, a study to this end is for example found in [284] where Lagrangian FEM, Arbitrary Lagrangian Eulerian (ALE) and Coupled Eulerian Lagrangian (CEL) FEM methods are compared. It is found that both the ALE and CEL approaches are superior to the Lagrangian method, at least for the prediction of the chip shape.

The ALE approach has been used in parallel to the purely Lagrangian approach at least since 1999, see [189]; with extension to three dimensions in [8] and [153]. This method avoids a lot of the drawbacks of the Lagrangian FEM by frequently interpolating between a Eularian and Lagrangian mesh.

2.3.1 *Meshless Metal Cutting Simulations*

One of the most severe drawbacks of simulating metal cutting with mesh-bound methods like the Lagrangian FEM and variations thereof is mesh distortion. That is, due to the very large deformations encountered during metal cutting simulations the mesh may exhibit elements with very bad aspect ratios or even volume inverted elements, necessitating remeshing operations. These methods are computationally very expensive and may introduce additional numerical errors. Oscillatory modes relevant to the solution may be suppressed since values at newly generated nodes need to be interpolated from existing ones, which is, in essence, a form of low pass filtering the solution.

Since meshless methods are either completely devoid of a mesh or only weakly bound to a background mesh, they are, conceptually, not limited in the amount of deformation they can represent. However, in practice, instability modes like the tensile instability may still limit the amount of deformation possible, see [255] and section 4.6.1. Still, meshless methods are

a promising candidate to simulate metal cutting. Since this topic is the focus of the thesis at hand, works with regard to meshless metal cutting are singled out in this section, it is attempted to present a quite comprehensive overview over the literature available.

The first work considering SPH for metal cutting was presented in 1997 [104]. Orthogonal metal cutting was modeled with various rake angles, including quite large negative rake angles which are especially hard to simulate using FEM due to the high burden they impose on the remeshing algorithms [3]. The results look plausible, but an experimental verification is lacking. With the introduction of the SPH to the commercial solver LSDYNA other works surfaced. First there was [145], closely followed by [146], where orthogonal high speed cutting was explored. This time experimental data was presented along the simulation results, with good agreement. Oblique cutting is shown, but only one exmplary frame of the simulation result is presented, with no experimental validation of any kind. Very similar simulations are presented in [269].

Probably the first publication presenting experimentally verified 3 dimensional metal cutting simulation is [226]. Just as in [3], single grain cutting is presented, with results of very comparable accuracy, but at a fraction of the computational cost. These results are also summarized in the thesis [225] by the same author, along with some orthogonal cutting and milling simulations. Some additional notes in the same single grain situation with regard to stagnant zones are provided in [227]. Meanwhile, in [156] damage modeling in conjunction with SPH was studied, especially with respect to the chip shape in general and generation of a serrated chip in particular. Another work from the same year again models orthogonal turning, but using a fully three dimensional model [12], thus foregoing the questionable plane stress or strain assumptions. In [33] tool wear effects are investigated by means of the most high resolution SPH metal cutting simulations up until this thesis, at least to the best of the authors knowledge.

All works except the very first one [104] employed the SPH solver in LSDYNA. However, ABAQUS features a SPH solver aswell. This solver was used recently in [13] to model laser assisted machining, demonstrating that the SPH approach is robust even for very complex processes and in [281] to combine the SPH approach with a quite involved crystal plasticity model.

Few researchers present their own meshless codes for metal cutting simulations. In a series of papers the pasimodo SPH code is developed to handle orthogonal metal cutting in a first step [246] [69] [245] and is then made adaptive to dynamically increase the spatial resolution of the cutting zone [247] [244]. In the same vein, a quite involved code using enrichment functions and switching between Eulerian and Lagrangian kernels is developed in [211] to simulate various shear localization phenomena, including metal cutting. Experimental verification of the code is lacking however.

There are also some meshless codes developed beyond SPH: Recently, an EFG solver was presented in [128] and employed to simulate cutting of composites, although it is arguable if the solver at hand can still be considered meshless. The Material Point Method (MPM) was also implemented in [6] and then again more recently in [96], in the latter case at extreme run time costs in the order of months. Development of a Particle Finite Element (PFEM)

solver was started in [222] and further pursued in some recent works [221] [207] specialized in chip geometry and [219] [220] material models, respectively. Again, the PFEM approach is not meshless in the strict sense since, while no persistent mesh structure is maintained, a new mesh is developed on the fly using Delaunay triangulation [57] in each time step.

Finally, the two works which are probably most close to the objectives of this thesis are mentioned: In [183] a corrected SPH approach is employed for metal cutting and in [80] a SPH solver is developed for friction stir welding, accelerated by GPGPU computing and enhanced with artificial stress terms according to publication [91], see also section 4.7.2. However, the first work is limited to two dimensions and only respects the thermo mechanical coupling in one way, while the latter does not employ linear complete kernels and considers a situation without material separation.

2.4 RESEARCH GAP AND OBJECTIVES

Being presented with the literature listed in the preceding section, especially 2.3.1, the following research gaps are identified:

- While there are a few GPU accelerated SPH codes for fluid mechanics, a GPU accelerated SPH code for solid mechanics is missing from the literature
- The modified SPH methods as used by the computer graphics community, e.g. [178], [19], have not been combined with physically valid plasticity models yet, let alone metal cutting.
- There are no total Lagrangian metal cutting simulations, neither meshfree nor otherwise. albeit [211] comes close, but switches to Eulerian kernels in parts of the domain that exhibit large deformations.
- Corrected, i.e. linear complete SPH methods are largely absent from metal cutting simulations, especially in conjunction with a complete thermo-mechanically coupled material model in three dimensions
- Correction terms and procedures such as artificial stress terms [91], transport velocity formulations [282] or XSPH [166] did not find employment in meshless metal cutting simulations so far.

Thus, the ultimate objective of this thesis can be summarized as the development of a three dimensional, thermo mechanically coupled, linear complete, GPU accelerated SPH solver for metal cutting enhanced by various correction terms. The thesis follows the principle of rising complexity. After discussing the continuum mechanics necessary to simulate metal cutting and the numerics of meshless methods, a wide array of algorithms is designed. They are then tested and compared against commercial FEM packages, first in an elastic, then plastic setting. After these trial simulations, the algorithms are subjected to orthogonal metal cutting and finally single grain cutting. Algorithms found unsuitable are eliminated along the way through these preliminary benchmarks.

3

CONTINUUM MECHANICS AND CONSTITUTIVE MODELING

In essence, this chapter describes the physics needed to simulate a metal cutting operation. Of course, it is somewhat debatable what physical effects *need* to be taken into account to simulate metal cutting, i.e. what physics to respect and neglect, respectively. There is currently no consensus to this end, and works introducing more involved physics to metal cutting simulations or more complex constitutive models are still being published.

Since pushing the envelope to what is possible to model was not goal of this thesis it was decided that the state of the art, i.e. the currently most popular models should be employed for this work. Care was taken to clearly explain and, where applicable, derive these models.

This chapter has the following structure: First, the notation used for physical quantities in this work are defined and the very basics of solid mechanics are quickly reiterated. On this basis, the conservation laws in both total and updated Lagrangian frames are discussed. The actual constitutive model is split in elastic, plastic and thermal effects.

3.1 KINEMATICS AND STRESS MEASURES

Note that the beginning of this chapter roughly follows what is given in [25] but is repeated here for completeness. The fundamental goal of continuum mechanics, in this case solid mechanics, is to describe the response of a body undergoing deformation by means of smooth functions of spatial variables. An initial, or reference, configuration of the body is introduced and called Ω_0. Material points belonging to Ω_0 are denoted by uppercase $\underline{X}$. The body undergoes some deformation to arrive at the deformed, or current configuration Ω. That is, the points $\underline{X}$ are mapped by function $\phi(.)$ to their location $\underline{x}$ in space at time t:

$$\underline{x} = \phi(\underline{X}, t) \tag{3.1}$$

lowercase $\underline{x}$ denotes the points in the deformed configuration. These concepts are illustrated in sketch 3.1. The distance vector $\underline{u} = \underline{X} - \underline{x}$ is called the displacement. Velocity and acceleration are subsequently defined as:

$$\begin{aligned} \dot{\underline{x}} = \underline{v} &= \frac{d}{dt}\underline{x} = \frac{d}{dt}(\underline{X} + u) = \dot{\underline{u}} \\ \ddot{\underline{x}} &= \dot{\underline{v}} \end{aligned} \tag{3.2}$$

An important variable in solid mechanics is the deformation gradient $\underline{\underline{F}}$. It is defined as:

$$\underline{\underline{F}} = \frac{\partial \phi(\underline{X}, t)}{\partial \underline{X}} = \frac{\partial \underline{x}}{\partial \underline{X}} \tag{3.3}$$

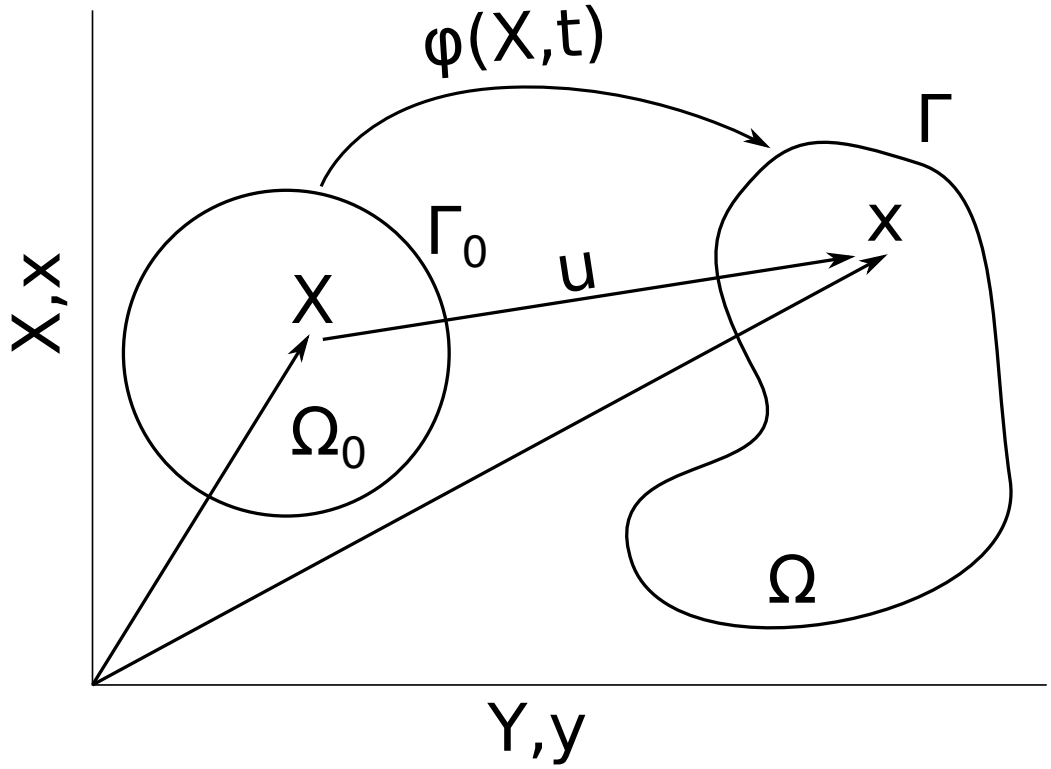

Figure 3.1: Sketch of the most basic solid mechanics quantities.

This is the Jacobian matrix of $\phi(\underline{X}, t)$. Its physical interpretation is that it describes the deformation of an infinitesimal line segment $d\underline{X}$ subjected to $\phi(.)$, hence the name *deformation* gradient:

$$d\underline{x} = \underline{\underline{F}} \cdot d\underline{X} \tag{3.4}$$

Another quantity often encountered in balance equations and constitutive modeling is the velocity gradient:

$$\underline{\underline{L}} = \nabla \underline{v} = \frac{\partial}{\partial \underline{x}} \otimes \underline{v} \tag{3.5}$$

It is also useful to consider the derived quantities:

$$\underline{\underline{D}} = 1/2 \left(\underline{\underline{L}} + \underline{\underline{L}}^T \right) \qquad \underline{\underline{W}} = 1/2 \left(\underline{\underline{L}} - \underline{\underline{L}}^T \right) \tag{3.6}$$

where $\underline{\underline{D}}$ is related to the strain imposed by the current velocity field. The strain is to be discussed later, consider (3.11). $\underline{\underline{W}}$ is related to the rotation imposed by the current velocity field and consequently called the spin tensor.

In principle, $\phi(.)$ could be any function. However, in order for the description to make physical sense $\phi(.)$ has to conform to certain properties.

- $\phi(.)$ is continuous and continuously differentiable. Technically, sets of measure zero, i.e. lines in 2D and surfaces in 3D on which the function may not be differentiable are allowed. However, since this work does not concern itself with fractures this is omitted from any further discussion.

- $\phi(.)$ is one-to-one and onto, i.e. bijective
- the Jacobian determinant, that is $det(\underline{\underline{F}}) = J$ is positive $J > 0$.

The physical meaning of the first two conditions is: Gaps may not open in the deformed configuration, the function may not map two or more points $\underline{X}$ onto the same location $\underline{x}$ and the function is invertible. Since the function is invertible it follows that $J \neq 0$. $J > 0$ is a stronger statement that will be shown later (3.30).

Closely related to the notion of the deformation there is the notion of strain, or measure thereof. A strain measure quantifies how much the displacement differs from a rigid body mode. There are a host of different such strain measures with varying properties. A popular choice is the Green St-Venant or Cauchy Green strain tensor:

$$\begin{aligned}\underline{\underline{E}} &= \frac{1}{2}\left(\underline{\underline{F}}^T \cdot \underline{\underline{F}} - \underline{\underline{I}}\right) \\ &= \frac{1}{2}\left[\nabla_0 \underline{u}^T + \nabla_0 \underline{u} + \nabla_0 \underline{u}^T \cdot \nabla_0 \underline{u}\right]\end{aligned} \tag{3.7}$$

where ∇_0 is the gradient with regard to the reference coordinates $\underline{X}$. It fulfills the following properties:

- It is zero when there is no displacement:

$$\begin{aligned}\underline{\underline{E}} &= \frac{1}{2}\left(\underline{\underline{F}} \cdot \underline{\underline{F}}^T - \underline{\underline{I}}\right) \\ &= \frac{1}{2}\left[\frac{\partial \underline{x}}{\partial \underline{X}} \cdot \left(\frac{\partial \underline{x}}{\partial \underline{X}}\right)^T - \underline{\underline{I}}\right] \\ &= \frac{1}{2}\left[\frac{\partial \underline{X} + \overbrace{\underline{u}}^{0}}{\partial \underline{X}} \cdot \left(\frac{\partial \underline{X} + \overbrace{\underline{u}}^{0}}{\partial \underline{X}}\right)^T - \underline{\underline{I}}\right] \\ &= \frac{1}{2}\left(\underline{\underline{I}} \cdot \underline{\underline{I}} - \underline{\underline{I}}\right) = 0\end{aligned} \tag{3.8}$$

- It increases as the deformation increases [110]
- It is independent from rigid body motions as can be seen using the polar decomposition theorem:

$$\underline{\underline{F}} = \underline{\underline{R}} \cdot \underline{\underline{U}} \tag{3.9}$$

  where $\underline{\underline{R}}$ is a rotation matrix and $\underline{\underline{U}}$ is a positive-semidefinite Hermitian matrix, representing the stretch component of $\underline{\underline{F}}$. By exploring $\underline{\underline{C}} = \underline{\underline{F}}^T \cdot \underline{\underline{F}}$:

$$\underline{\underline{F}}^T \cdot \underline{\underline{F}} = \left(\underline{\underline{R}} \cdot \underline{\underline{U}}\right)^T \cdot \left(\underline{\underline{R}} \cdot \underline{\underline{U}}\right) = \underline{\underline{U}}^T \cdot \underline{\underline{R}}^T \cdot \underline{\underline{R}} \cdot \underline{\underline{U}} = \underline{\underline{U}}^T \cdot \underline{\underline{U}} \tag{3.10}$$

Sometimes $\underline{\underline{E}}$ is linearized by neglecting the quadratic terms in (3.7):

$$\begin{aligned}\underline{\underline{E}}^{\text{lin}} &= \frac{1}{2}\left[(\nabla_0 \underline{u})^T + (\nabla_0 \underline{u}) + \underbrace{(\nabla_0 \underline{u})^T \cdot (\nabla_0 \underline{u})}_{\stackrel{!}{=}0}\right] \\ &= \frac{1}{2}\left(\underline{\underline{F}} + \underline{\underline{F}}^T\right) - \underline{\underline{I}} \end{aligned} \tag{3.11}$$

This approximation, while convenient violates the last property above, i.e. that it is invariant under rigid body rotation. It may still be a useful approximation if only small rotations are present. Instead of by linearizing the Green St-Venant strain tensor $\underline{\underline{E}}^{\text{lin}}$ can also be derived directly by geometric considerations, see [14]. Note that:

$$\dot{\underline{\underline{E}}}^{\text{lin}} = \underline{\underline{D}} \tag{3.12}$$

3.1.1 *Stress Measures*

What has not been discussed so far is the forces acting on a certain point in either Ω_0 or Ω. Again, there are different measures to this end which can be defined. These *stress* measures are contrary to strain measures discussed before, or the objective stress rates discussed later, somewhat more limited in number. They also relate more directly to physics in the sense that they can be measured directly in certain situations, e.g. in a uniaxial tensile test. The various stress measures differ in the representation they map from and on to, and what kind of element they are defined on.

The most common stress measure is the Cauchy stress tensor $\underline{\underline{\sigma}}$. It is a measure of the force $d\underline{f}$ acting on an element of area in the deformed configuration:

$$\underline{n} \cdot \underline{\underline{\sigma}}\, d\Gamma = \underline{t}\, d\Gamma = d\underline{f} \tag{3.13}$$

where $\underline{t}$ is the surface traction and $\underline{n}$ is the surface normal in the current configuration. Due to the conservation of angular momentum, the Cauchy stress tensor is symmetric in the Boltzmann continuum, polar media which are described by the Cosserat continuum do not feature this property and $\underline{\underline{\sigma}} \neq \underline{\underline{\sigma}}^T$. For the rest of this thesis; $\underline{\underline{\sigma}} = \underline{\underline{\sigma}}^T$ and the hydrostatic pressure p is exactly the volumetric part $-tr(\underline{\underline{\sigma}}) = 3 \cdot p$.

The nominal stress is analogous to the Cauchy stress tensor in the sense that it is a measure of force $d\underline{f}$ acting on an element of area again, but this time in the reference configuration:

$$\underline{n}_0 \cdot \underline{\underline{P}}\, d\Gamma_0 = \underline{t}_0\, d\Gamma_0 = d\underline{f} \tag{3.14}$$

contrary to the Cauchy stress tensor, $\underline{\underline{P}}$ is not symmetric. Its transpose is called the First Piola-Kirchhoff stress $\underline{\underline{\sigma}}^{\text{PK1}} = \underline{\underline{P}}^T$.

The last stress measure to be considered in this thesis is the second Piola-Kirchhoff stress tensor. It differs from $\underline{\underline{P}}$ simply by a pull back by $\underline{\underline{F}}^{-1}$ to the traction on the right hand side.

$$\underline{n}_0 \cdot \underline{\underline{\sigma}}^{\text{PK2}}\, d\Gamma_0 = \underline{\underline{F}}^{-1} \cdot \underline{t}_0\, d\Gamma_0 \tag{3.15}$$

this operation makes it symmetric and energy conjugate to the Cauchy Green strain tensor. Since $\underline{\underline{\sigma}}^{\mathrm{PK1}}$, $\underline{\underline{P}}$ and $\underline{\underline{\sigma}}^{\mathrm{PK2}}$ refer to the undeformed state their volumetric part is *not* equal to p.

Sometimes, it is necessary to convert between these different stress measures. A handy table with all possible transforms is given in reference [25]. In this work only two such transformations are necessary. They are readily derived using Nanson's relation [157]:

$$\underline{n} d\Gamma = J \underline{n}_0 \cdot \underline{\underline{F}}^{-1} d\Gamma_0 \tag{3.16}$$

A relation between the nominal and Cauchy stress measures is found by equation (3.13) and (3.14).

$$\underline{n}_0 \cdot \underline{\underline{P}}\, d\Gamma_0 = \underline{n} \cdot \underline{\underline{\sigma}}\, d\Gamma \tag{3.17}$$

applying Nanson's relation on the right hand side gives the desired relation:

$$\begin{aligned} \underline{n}_0 \cdot \underline{\underline{P}}\, d\Gamma_0 &= J \underline{n}_0 \cdot \underline{\underline{\sigma}} \cdot \underline{\underline{F}}^{-1} d\Gamma_0 \\ \underline{\underline{P}} &= J \cdot \underline{\underline{\sigma}} \cdot \underline{\underline{F}}^{-1} \end{aligned} \tag{3.18}$$

where it was used that the above holds for any $\underline{n}_0$ and everywhere on the boundary Γ_0. Another transformation needed in this work is from $\underline{\underline{\sigma}}$ to $\underline{\underline{\sigma}}^{\mathrm{PK2}}$. To this end (3.15) is manipulated by multiplying with $\underline{\underline{F}}$ from the left:

$$\begin{aligned} \underline{\underline{F}} \cdot \underline{\underline{F}}^{-1} \cdot \underline{t}_0\, d\Gamma_0 &= \underline{\underline{F}} \cdot \underline{n}_0 \cdot \underline{\underline{\sigma}}^{\mathrm{PK2}}\, d\Gamma_0 \\ \underline{t}_0\, d\Gamma_0 &= \underline{\underline{F}} \cdot \underline{\underline{\sigma}}^{\mathrm{PK2}} \cdot \underline{n}_0\, d\Gamma_0 \end{aligned} \tag{3.19}$$

where the last equality holds since $\underline{\underline{\sigma}}^{\mathrm{PK2}}$ is symmetric. (3.14) is now made "compatible" with (3.19) by moving the normal to the right:

$$\underline{n}_0 \cdot \underline{\underline{P}}\, d\Gamma_0 = \underline{\underline{P}}^T \cdot \underline{n}_0\, d\Gamma_0 \tag{3.20}$$

Thus:

$$\underline{\underline{F}} \cdot \underline{\underline{\sigma}}^{\mathrm{PK2}} \cdot \underline{n}_0\, d\Gamma_0 = \underline{\underline{P}}^T \cdot \underline{n}_0\, d\Gamma_0 \tag{3.21}$$

again, by noticing that the above is true for any $\underline{n}_0$ on all of $\underline{\Gamma}_0$:

$$\begin{aligned} \underline{\underline{F}} \cdot \underline{\underline{\sigma}}^{\mathrm{PK2}} &= \underline{\underline{P}}^T \\ \underline{\underline{\sigma}}^{\mathrm{PK2}} &= \underline{\underline{F}}^{-1} \cdot \underline{\underline{P}}^T \end{aligned} \tag{3.22}$$

finally, (3.18) is used to relate $\underline{\underline{\sigma}}^{\mathrm{PK2}}$ to $\underline{\underline{\sigma}}$:

$$\begin{aligned} \underline{\underline{\sigma}}^{\mathrm{PK2}} &= \underline{\underline{F}}^{-1} \cdot \left(J \cdot \underline{\underline{\sigma}} \cdot \underline{\underline{F}}^{-1} \right)^T \\ &= J \underline{\underline{F}}^{-1} \cdot \underline{\underline{\sigma}} \cdot \underline{\underline{F}}^{-T} \end{aligned} \tag{3.23}$$

3.2 CONSERVATION LAWS

The definitions in the previous section are enough to formulate the fundamental conservation laws, or balance equations to be solved by meshless methods in this work. The total mass M of a body can be defined as:

$$M = \int_{\Omega} \varrho(\underline{x}, t)\, d\underline{x} \tag{3.24}$$

where $\varrho(.)$ is the density, and $d\underline{x}$ is to be understood as the infinitesimal volume element pointed at by $\underline{x}$. Mass conservation states that the mass does not change, i.e. the material derivative of mass is zero:

$$\frac{DM}{Dt} = \frac{D}{Dt} \int_{\Omega} \varrho(\underline{x}, t)\, d\underline{x} = 0 \tag{3.25}$$

using Reynolds Theorem:

$$\int_{\Omega} \frac{D\varrho(\underline{x}, t)}{Dt} + \varrho(\underline{x}, t) \nabla \underline{v}\, d\underline{x} = 0 \tag{3.26}$$

Thus:

$$\frac{D\varrho(\underline{x}, t)}{Dt} = -\varrho(\underline{x}, t) \nabla \underline{v} \tag{3.27}$$

which is called the continuity equation. Another, equivalent form of the continuity equation is found by stating that the total mass has to be equal in both reference configuration and current configuration:

$$M = \int_{\Omega} \varrho(\underline{x}, t)\, d\underline{x} = \int_{\Omega_0} \varrho_0(\underline{X}, t)\, d\underline{X} \tag{3.28}$$

the left hand side integral is then transformed to the reference configuration:

$$\begin{gathered} \int_{\Omega_0} \varrho(\underline{x}, t) J\, d\underline{X} = \int_{\Omega_0} \varrho_0(\underline{X}, t)\, d\underline{X} \\ \int_{\Omega_0} \varrho(\underline{x}, t) J - \varrho_0(\underline{X}, t)\, d\underline{X} = 0 \end{gathered} \tag{3.29}$$

Hence, in shortened notation:

$$\varrho_0 = J \cdot \varrho \tag{3.30}$$

Note that this statement immediately entails that the determinant of $det(\underline{\underline{F}}) = J$ needs to be strictly greater than zero, since the density can not take negative or null values. Next, the conservation of linear momentum, i.e. Newton's second law of motion shall be investigated. Newton's second law reads:

$$\frac{D\underline{\mathcal{P}}}{Dt} = \underline{\mathcal{F}} \tag{3.31}$$

where $\underline{\mathcal{P}}$ is the total linear momentum and $\underline{\mathcal{F}}$ is the total force acting on a body. This can be written in more verbose fashion:

$$\frac{D}{Dt}\int_\Omega \varrho(\underline{x},t)\cdot\underline{v}(\underline{x},t)\;d\underline{x} = \int_\Omega \varrho(\underline{x},t)\cdot\underline{b}(\underline{x},t)\;d\underline{x} + \int_\Gamma \underline{t}(\underline{x},t)\;d\underline{x} \tag{3.32}$$

in which $\underline{b}(.)$ denotes the body forces. In the following, the notation is shortened for convenience:

$$\frac{D}{Dt}\int_\Omega \varrho\cdot\underline{v}\;d\underline{x} = \int_\Omega \varrho\cdot\underline{b}\;d\underline{x} + \int_\Gamma \underline{t}\;d\underline{x} \tag{3.33}$$

the goal is now to find an expression for the above which resides under the same integral, i.e. $\int_\Omega \ldots\; d\underline{x} = 0$. To this end, the material derivative on the left hand sign is pulled into the integrand by using Reynold's transport theorem:

$$\frac{D}{Dt}\int_\Omega \varrho\cdot\underline{v}\;d\underline{x} = \int_\Omega \frac{D}{Dt}(\varrho\cdot\underline{v}) + (\nabla\underline{v})\varrho\underline{v}\;d\underline{x} \tag{3.34}$$

the term $D/Dt\varrho\cdot\underline{v}$ is then expanded using the product rule:

$$\begin{aligned}\int_\Omega \frac{D}{Dt}(\varrho\cdot\underline{v}) + (\nabla\underline{v})\varrho\underline{v}\;d\underline{x} &= \int_\Omega \varrho\frac{D\underline{v}}{Dt} + \frac{D\varrho}{Dt}\underline{v} + (\nabla\underline{v})\varrho\underline{v}\;d\underline{x}\\ &= \int_\Omega \varrho\frac{D\underline{v}}{Dt} + \underbrace{\left(\frac{D\varrho}{Dt} + (\nabla\underline{v})\varrho\right)}_{0}\underline{v}\;d\underline{x}\end{aligned} \tag{3.35}$$

the term evaluating zero is nothing but the continuity equation (3.27). Hence:

$$\frac{D}{Dt}\int_\Omega \varrho\cdot\underline{v}\;d\underline{x} = \int_\Omega \varrho\frac{D\underline{v}}{Dt}\;d\underline{x} \tag{3.36}$$

Next, the integral over the boundary Γ in (3.33) is manipulated by using the definition of the Cauchy stress (3.13), then by use of Gauss's theorem:

$$\int_\Gamma \underline{t}\;d\underline{x} = \int_\Gamma \underline{\underline{\sigma}}\cdot\underline{n}\;d\underline{x} = \int_\Omega \nabla\underline{\underline{\sigma}}\;d\underline{x} \tag{3.37}$$

Finally:

$$\int_\Omega \varrho\frac{D\underline{v}}{Dt} - \nabla\underline{\underline{\sigma}} - \varrho\underline{b}\;d\underline{x} = 0 \tag{3.38}$$

Or, more commonly written as:

$$\frac{D\underline{v}}{Dt} = \frac{1}{\varrho}\nabla\underline{\underline{\sigma}} + \underline{b} \tag{3.39}$$

which is called the momentum equation in the updated Lagrangian frame. To find the momentum equation in the total Lagrangian frame the analysis could be restarted at (3.33), i.e. Newton's second law, but form the integrals over the reference configuration. A quicker option is to transform (3.38):

$$\int_{\Omega_0} J\left(\varrho\frac{D\underline{v}}{Dt} - \nabla\underline{\underline{\sigma}} - \varrho\underline{b}\right)\;d\underline{x} = 0 \tag{3.40}$$

Table 3.1: Conservation laws in both total Lagrangian and Updated Lagrangian frames

Updated Lagrangian frame	Total Lagrangian frame
$\frac{D\underline{v}}{Dt} = \frac{1}{\varrho}\nabla\underline{\underline{\sigma}} + \underline{b}$	$\frac{D\underline{v}}{Dt} = \frac{1}{\varrho_0}\nabla_0\underline{\underline{P}} + \underline{b}$
$\frac{D\varrho}{Dt} = -\varrho\nabla\underline{v}$	$\varrho = \varrho/J$

using $J\varrho = \varrho_0$ and $\underline{\underline{\sigma}} = J^{-1}\underline{\underline{F}} \cdot \underline{\underline{P}}$ from (3.18).

$$\begin{aligned} 0 &= \int_{\Omega_0} \varrho_0 \frac{D\underline{v}}{Dt} - \nabla\underline{\underline{F}} \cdot \underline{\underline{P}} - \varrho_0\underline{b}\, d\underline{x} \\ &= \int_{\Omega_0} \varrho_0 \frac{D\underline{v}}{Dt} - \frac{\partial}{\partial\underline{x}} \cdot \frac{\partial\underline{x}}{\partial\underline{X}} \cdot \underline{\underline{P}} - \varrho_0\underline{b}\, d\underline{x} \\ &= \int_{\Omega_0} \varrho_0 \frac{D\underline{v}}{Dt} - \nabla_0\underline{\underline{P}} - \varrho_0\underline{b}\, d\underline{x} \end{aligned} \tag{3.41}$$

which leads to the momentum equation in the total Lagrangian frame:

$$\frac{D\underline{v}}{Dt} = \frac{1}{\varrho_0}\nabla_0\underline{\underline{P}} + \underline{b} \tag{3.42}$$

In principal, both (3.27) and (3.30) can be used with (3.42) and vice versa. However, it is common to use (3.30) in conjunction with (3.42) and (3.27) in conjunction with (3.39), since the quantities in the continuum equations are more readily available in the respective frame. The two systems are summarized again in table 3.1 for convenience.

This concludes the discussion about the mechanical conservation laws. The conservation laws for heat will be discussed in section 3.5. In other texts, conservation laws regarding energy for purely mechanical systems are often discussed along the two mechanical conservation laws discussed here, but they are of no consequence for the matter at hand and will thus be omitted for from the discussion. Readers interested in the conservation of energy equation are referred to standard texts such as [25] instead.

3.3 ELASTICITY

So far, various definitions regarding the system variables in a continuum/solid mechanical system were given as well as conservation laws regarding them. When considering the resulting systems in table 3.1 one will notice that the systems are not closed, i.e. there is no evolution equation for the stress.

To relate the stresses to some function of the strains is what lies in the center of continuum- or material modelling. This section deals with the elastic part of these matters, i.e. the deformations which are energetically reversible. It is hard to tell what exactly constitutes the state of the art in metal cutting simulations regarding the elastic regime, since current works usually omit this discussion completely. The reason for this may be two-fold:

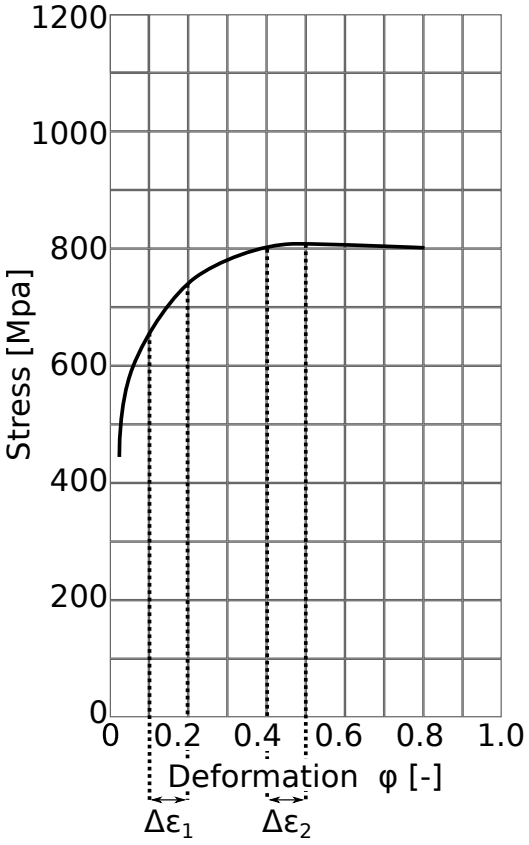

Figure 3.2: A typical stress strain curve for metal as exhibited by 20MnCr5 at room temperature, taken from [62]. Changes to the strain in the elastic regime, e.g. $\Delta\varepsilon_1$ cause larger changes to the stress than in the plastic regime, e.g. $\Delta\varepsilon_2$; $\phi = \log(\varepsilon + 1)$.

- Commercial solver packages do not offer much liberty in modelling to this end, i.e. the user is for example not free to chose which objective stress rate is to be used, c.f. the discussion later on in this section. Most research on metal cutting simulations is done using such commercial packages, see for example the round robin in [119], where 90% of the participating researches opted to use commercial FEM packages.

- The elastic range of metals is very small, usually less then $\sim 0.2\%$ but up to $\sim 0.8\%$ strain in some Titanium alloys.

The last point should not imply that the elastic model can be designed with neglect. Care has to be taken that the elastic model does not induce stress under rigid body rotation or translation. Also, the impact on the stress due to a small change in strain in the elastic regime is orders of magnitude larger than in the plastic regime, see figure 3.2. A reasonable model for the metal elasticity is a linear dependence of the strains on the stresses , i.e. Hooke's law, although it hast to be noted that some metals like Inconel violate this assumption. In its most general form it can be stated as:

$$\underline{\underline{\sigma}} = \underline{\underline{\underline{\underline{C}}}} : \underline{\underline{\varepsilon}} \tag{3.43}$$

where $\underline{\underline{\underline{\underline{C}}}}$ is the fourth order stiffness tensor and $\underline{\underline{\varepsilon}}$ some strain measure. In principle any strain measure can be used, however, should the material undergo finite, as opposed to infinitesimal rotations one is best advised to use a finite strain measure for $\underline{\underline{\varepsilon}}$ such as $\underline{\underline{E}}$ (3.7).

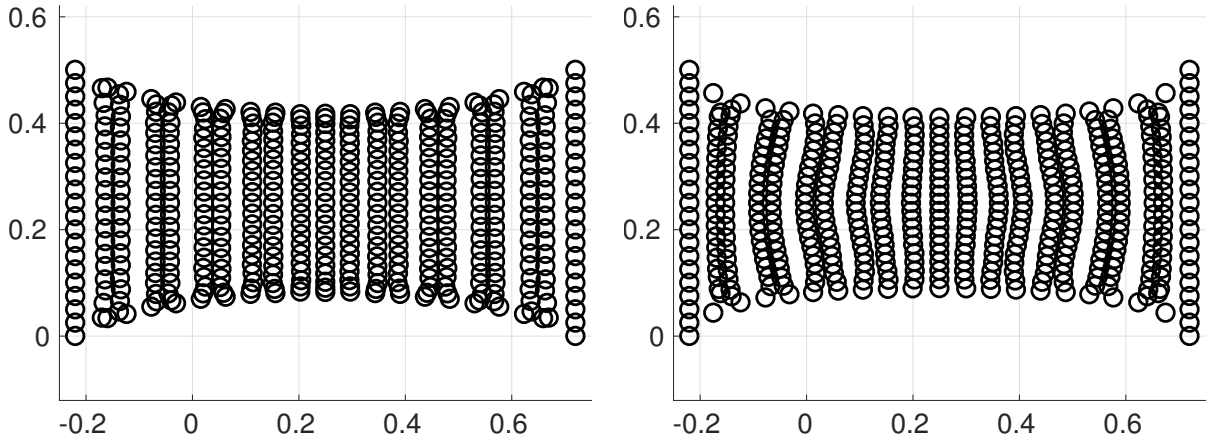

Figure 3.3: A tensile test is carried out using the total Lagrangian framework described in [217], using a direct update of the stress tensor via the Green St-Venant strain tensor on the left and the Jaumann rate to the right. While both simulations show quite severe pairing instabilities along the x-axis, the approach using the direct update additionally shows the same instability modes along the y-axis.

By assuming isotropic media this equation greatly simplifies, from 81 material parameters to just two. Metals typically have a texture from cold rolling, drawing or extrusion, which can reasonably be neglected for metal cutting. Hence, both homogeneity and isotropy is assumed, so the discussion will be limited to this case. In principle, Hooke's law can still be expressed in form (3.43), with most terms in $\underset{\approx}{C}$ being equal or zero. However, for the purpose at hand there is a more convenient expression that distinguishes shape and volumetric changes using the Lamé parameters [94]:

$$\underline{\underline{\sigma}} = \underbrace{\frac{E}{1+\nu}}_{2G} \underline{\underline{\varepsilon}} + \underbrace{\frac{E \cdot \nu}{(1-2\nu)(1+\nu)}}_{\lambda} \underline{\underline{I}} \cdot tr(\underline{\underline{\varepsilon}}) \tag{3.44}$$

with Young's Modulus E and Poisson's ratio ν, simultaneously introducing the first Lamé parameter G, also known as the shear modulus, and first Lamé parameter λ.

while a model like (3.44) is valid for linear elasticity, the question remains whether it is a convenient model, suitable for implementation into a computer code. There are two arguments against this:

- If finite deformations are to be considered, the model contains $\underline{\underline{\varepsilon}} = \underline{\underline{E}}$, which in turn contains derivatives with regard to reference coordinates $\underline{X}$. This might be awkward in use with the updated Lagrangian frame.
- In-house studies have shown that, at least for meshless methods, models that use a rate equation of the form $\underline{\underline{\dot{\sigma}}} = \mathcal{S}(\underline{\underline{\sigma}}, \ldots)$ are more stable, see figure 3.3.

To fix the last idea the straight forward approach would be to derive (3.43) with regard to time:

$$\underline{\underline{\dot{\sigma}}} = \underset{\approx}{C} : \underline{\underline{\dot{\varepsilon}}} \tag{3.45}$$

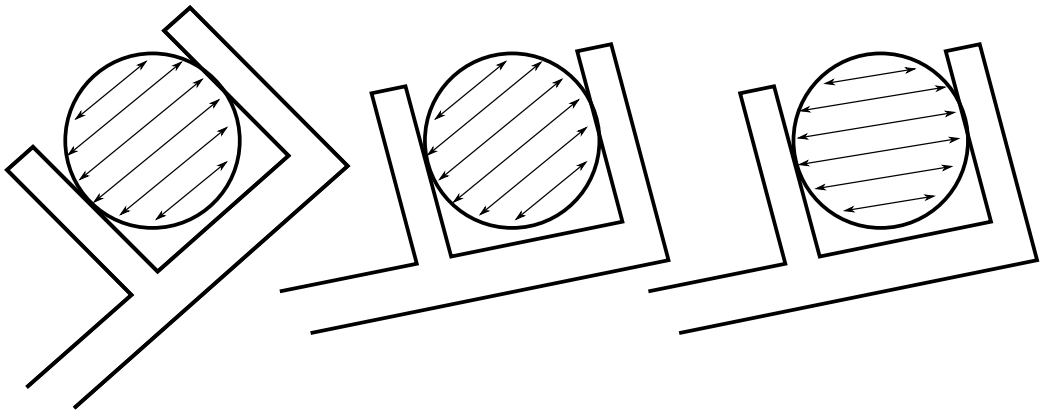

Figure 3.4: A disk is pre-stressed with a wrench. This wrench is then turned to impose a rigid body rotation. If an incorrect stress rate, e.g. one only dependent on the deformation $\underline{\underline{D}}$ is employed, the stress stays spuriously constant like in the middle figure. Correct stress rates exhibit the behavior as presented on the right hand side.

However, there is a problem with this approach. While the Cauchy stress tensor does transform correctly under rigid body rotations:

$$\underline{\underline{\sigma}}_r = \underline{\underline{R}} \cdot \underline{\underline{\sigma}} \cdot \underline{\underline{R}}^T \tag{3.46}$$

it's time derivative does not:

$$\dot{\underline{\underline{\sigma}}}_r = \dot{\underline{\underline{R}}} \cdot \underline{\underline{\sigma}} \cdot \underline{\underline{R}}^T + \underline{\underline{R}} \cdot \dot{\underline{\underline{\sigma}}} \cdot \underline{\underline{R}}^T + \underline{\underline{R}} \cdot \underline{\underline{\sigma}} \cdot \dot{\underline{\underline{R}}}^T \tag{3.47}$$

$$\neq \underline{\underline{R}} \cdot \dot{\underline{\underline{\sigma}}} \cdot \underline{\underline{R}}^T \tag{3.48}$$

To illuminate this issue further, a worked example taken from [272] in shape of a pre-stressed disk is presented, see figure 3.4. This disk shall undergo rotation. As stress update the naive approach (3.45) is employed with $\dot{\underline{\underline{\varepsilon}}} = \dot{\underline{\underline{E}}}^{lin} = \underline{\underline{D}}$. The velocity field of the rotation is

$$\underline{v} = \begin{bmatrix} -c \cdot y \\ c \cdot x \end{bmatrix} \tag{3.49}$$

Hence

$$\underline{\underline{D}} = 1/2 \left(\underline{\underline{L}} + \underline{\underline{L}}^T \right) = \left(\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}^T \right) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{3.50}$$

Which implies that the stress state erroneously does not rotate.

To correctly resolve rigid body rotations a different Ansatz has to be chosen. The following proof has been taken from the webpage, `http://www.continuummechanics.org/corotationalderivative.html`, retrieved 30.01.2018, since it could not be found in any of the established textbooks. The derivation is started with:

$$\underline{\underline{\sigma}}^{\mathrm{PK2}} = \underline{\underline{\underline{\underline{C}}}} : \underline{\underline{E}} \tag{3.51}$$

$\underline{\underline{\sigma}}^{\mathrm{PK2}}$ is then expressed in the Cauchy stress using transformation (3.23):

$$J\underline{\underline{F}}^{-1} \cdot \underline{\underline{\sigma}} \cdot \underline{\underline{F}}^{-T} = \underset{\approx}{C} : \underline{\underline{E}} \tag{3.52}$$

This is solved for $\underline{\underline{\sigma}}$:

$$\underline{\underline{\sigma}} = \frac{1}{J}\underline{\underline{F}} \cdot \left(\underset{\approx}{C} : \underline{\underline{E}}\right) \cdot \underline{\underline{F}}^{T} \tag{3.53}$$

This equation is now derived with regard to time, making use of the product rule on the right hand side:

$$\begin{aligned}\dot{\underline{\underline{\sigma}}} = &-\frac{\dot{J}}{J^2}\,\underline{\underline{F}} \cdot \left(\underset{\approx}{C} : \underline{\underline{E}}\right) \cdot \underline{\underline{F}}^{T} + \frac{1}{J}\,\dot{\underline{\underline{F}}} \cdot \left(\underset{\approx}{C} : \underline{\underline{E}}\right) \cdot \underline{\underline{F}}^{T} \\ &+ \frac{1}{J}\,\underline{\underline{F}} \cdot \left(\underset{\approx}{C} : \dot{\underline{\underline{E}}}\right) \cdot \underline{\underline{F}}^{T} + \frac{1}{J}\,\underline{\underline{F}} \cdot \left(\underset{\approx}{C} : \underline{\underline{E}}\right) \cdot \dot{\underline{\underline{F}}}^{T}\end{aligned} \tag{3.54}$$

To simplify the above further, following identities are required:

$$tr(\underline{\underline{D}}) = \frac{\dot{J}}{J} \tag{3.55}$$

$$\dot{\underline{\underline{F}}} = \underline{\underline{L}} \cdot \underline{\underline{F}} \tag{3.56}$$

$$\dot{\underline{\underline{F}}}^{T} = \underline{\underline{F}}^{T} \cdot \underline{\underline{L}} \tag{3.57}$$

$$\dot{\underline{\underline{E}}} = \underline{\underline{F}}^{T} \cdot \underline{\underline{D}} \cdot \underline{\underline{F}} \tag{3.58}$$

The relation (3.56) can be proven quickly by writing down the derivatives contained in the tensors:

$$\begin{aligned}\underline{\underline{L}} \cdot \underline{\underline{F}} &= \dot{\underline{\underline{F}}} \\ \frac{\partial \underline{v}}{\partial \underline{x}} \cdot \frac{\partial \underline{x}}{\partial \underline{X}} &= \frac{\partial \underline{v}}{\partial \underline{X}} \\ \frac{\partial \underline{v}}{\partial \underline{X}} &= \frac{\partial \underline{v}}{\partial \underline{X}}\end{aligned} \tag{3.59}$$

The relation (3.58) may not be immediately apparent, warranting its derivation. By using the definition of $\underline{\underline{D}}$ from (3.6) and the relation just derived (3.56):

$$\begin{aligned}\underline{\underline{D}} &= 1/2\left(\underline{\underline{L}} + \underline{\underline{L}}^{T}\right) \\ &= 1/2\left(\dot{\underline{\underline{F}}} \cdot \underline{\underline{F}}^{-1} + \underline{\underline{F}}^{-T} \cdot \dot{\underline{\underline{F}}}^{T}\right)\end{aligned} \tag{3.60}$$

This is now pre-multiplied with $\underline{\underline{F}}^{T}$ and pos-multiplied with $\underline{\underline{F}}$:

$$\begin{aligned}\underline{\underline{F}}^{T} \cdot \underline{\underline{D}} \cdot \underline{\underline{F}} &= 1/2\left[\underline{\underline{F}}^{T}\left(\dot{\underline{\underline{F}}} \cdot \underline{\underline{F}}^{-1} + \underline{\underline{F}}^{-T} \cdot \dot{\underline{\underline{F}}}^{T}\right)\underline{\underline{F}}\right] \\ &= 1/2\left(\underline{\underline{F}}^{T} \cdot \dot{\underline{\underline{F}}} + \dot{\underline{\underline{F}}}^{T} \cdot \underline{\underline{F}}\right)\end{aligned} \tag{3.61}$$

Considering the definition of the Green-St. Venant strain tensor (3.7), or rather its time derivative:

$$\begin{aligned}\dot{\underline{\underline{E}}} &= 1/2\left[\frac{D}{Dt}\left(\underline{\underline{F}}\cdot\underline{\underline{F}}^T-\underline{\underline{I}}\right)\right]\\ &= 1/2\left(\underline{\underline{F}}^T\cdot\dot{\underline{\underline{F}}}+\dot{\underline{\underline{F}}}^T\cdot\underline{\underline{F}}\right)\end{aligned} \tag{3.62}$$

which reveals that (3.58) is correct. The long term (3.54) can now be shortened by identifying $\underline{\underline{\sigma}}$ in most constituents:

$$\begin{aligned}\dot{\underline{\underline{\sigma}}} = &-tr(\underline{\underline{D}})\underbrace{\frac{1}{J}\underline{\underline{F}}\cdot\left(\underset{\approx}{C}:\underline{\underline{E}}\right)\cdot\underline{\underline{F}}^T}_{\underline{\underline{\sigma}}}+\underline{\underline{L}}\underbrace{\frac{1}{J}\underline{\underline{F}}\cdot\left(\underset{\approx}{C}:\underline{\underline{E}}\right)\cdot\underline{\underline{F}}^T}_{\underline{\underline{\sigma}}}\\ &+\frac{1}{J}\underline{\underline{F}}\cdot\left(\underset{\approx}{C}:\left(\underline{\underline{F}}^T\cdot\underline{\underline{D}}\cdot\underline{\underline{F}}\right)\right)\cdot\underline{\underline{F}}^T+\underbrace{\frac{1}{J}\underline{\underline{F}}\cdot\left(\underset{\approx}{C}:\underline{\underline{E}}\right)\cdot\underline{\underline{F}}^T}_{\underline{\underline{\sigma}}}\cdot\underline{\underline{L}}^T\end{aligned} \tag{3.63}$$

Thus:

$$\dot{\underline{\underline{\sigma}}} = \underline{\underline{L}}\cdot\underline{\underline{\sigma}}+\underline{\underline{\sigma}}\cdot\underline{\underline{L}}^T-tr(\underline{\underline{D}})\underline{\underline{\sigma}}+\frac{1}{J}\underline{\underline{F}}\cdot\left(\underset{\approx}{C}:\left(\underline{\underline{F}}^T\cdot\underline{\underline{D}}\cdot\underline{\underline{F}}\right)\right)\cdot\underline{\underline{F}}^T \tag{3.64}$$

The last term on the right hand side can further be simplified:

$$\frac{1}{J}\underline{\underline{F}}\cdot\left(\underset{\approx}{C}:\left(\underline{\underline{F}}^T\cdot\underline{\underline{D}}\cdot\underline{\underline{F}}\right)\right)\cdot\underline{\underline{F}}^T=\frac{1}{J}\left(\underline{\underline{F}}\cdot\underline{\underline{F}}\cdot\underset{\approx}{C}\cdot\underline{\underline{F}}^T\cdot\underline{\underline{F}}^T\right):\underline{\underline{D}}=\underset{\approx}{C}':\underline{\underline{D}} \tag{3.65}$$

Since the stiffness tensor is formulated with regard to an isotropic material it is not affected by deformation and $\underset{\approx}{C}=\underset{\approx}{C}'$. This leaves:

$$\dot{\underline{\underline{\sigma}}} = \underline{\underline{L}}\cdot\underline{\underline{\sigma}}+\underline{\underline{\sigma}}\cdot\underline{\underline{L}}^T-tr(\underline{\underline{D}})\underline{\underline{\sigma}}+\underset{\approx}{C}':\underline{\underline{D}} \tag{3.66}$$

this is the Truesdell rate [264] of the Cauchy stress and only one of an infinite number of rates which are co-rotational, i.e. correctly resolve rigid body rotations. Assuming that the material is nearly incompressible, i.e. $tr(\underline{\underline{D}})=0$ and the deformation is mostly rotational, i.e. $\underline{\underline{L}}\approx\underline{\underline{W}}$:

$$\dot{\underline{\underline{\sigma}}} = \underline{\underline{W}}\cdot\underline{\underline{\sigma}}-\underline{\underline{\sigma}}\cdot\underline{\underline{W}}+\underset{\approx}{C}':\underline{\underline{D}} \tag{3.67}$$

by skew-symmetry of $\underline{\underline{W}}=-\underline{\underline{W}}^T$. This above equation is the Jaumann rate of the Cauchy stress [118].

Yet another stress rate can be found by considering (3.66) again, but this time assuming that $tr(\underline{\underline{D}})=0$ and $\underline{\underline{F}}\approx\underline{\underline{R}}$ i.e. neglecting the stretch $\underline{\underline{U}}$ in $\underline{\underline{F}}=\underline{\underline{U}}\cdot\underline{\underline{R}}$. Since $\underline{\underline{L}}=\dot{\underline{\underline{F}}}\cdot\underline{\underline{F}}^{-1}$ this gives $\underline{\underline{L}}=\dot{\underline{\underline{R}}}\cdot\underline{\underline{R}}^{-1}=\dot{\underline{\underline{R}}}\cdot\underline{\underline{R}}^T$ under the assumption made above. Hence:

$$\dot{\underline{\underline{\sigma}}} = \left(\dot{\underline{\underline{R}}}\cdot\underline{\underline{R}}^T\right)\cdot\underline{\underline{\sigma}}+\underline{\underline{\sigma}}\cdot\left(\dot{\underline{\underline{R}}}^T\cdot\underline{\underline{R}}\right)+\underset{\approx}{C}':\underline{\underline{D}} \tag{3.68}$$

This is the Green Naghdi [92] rate of the Cauchy stress. This rate is quite challenging to implement into a computer code. A procedure due to [60] to this end is given in appendix

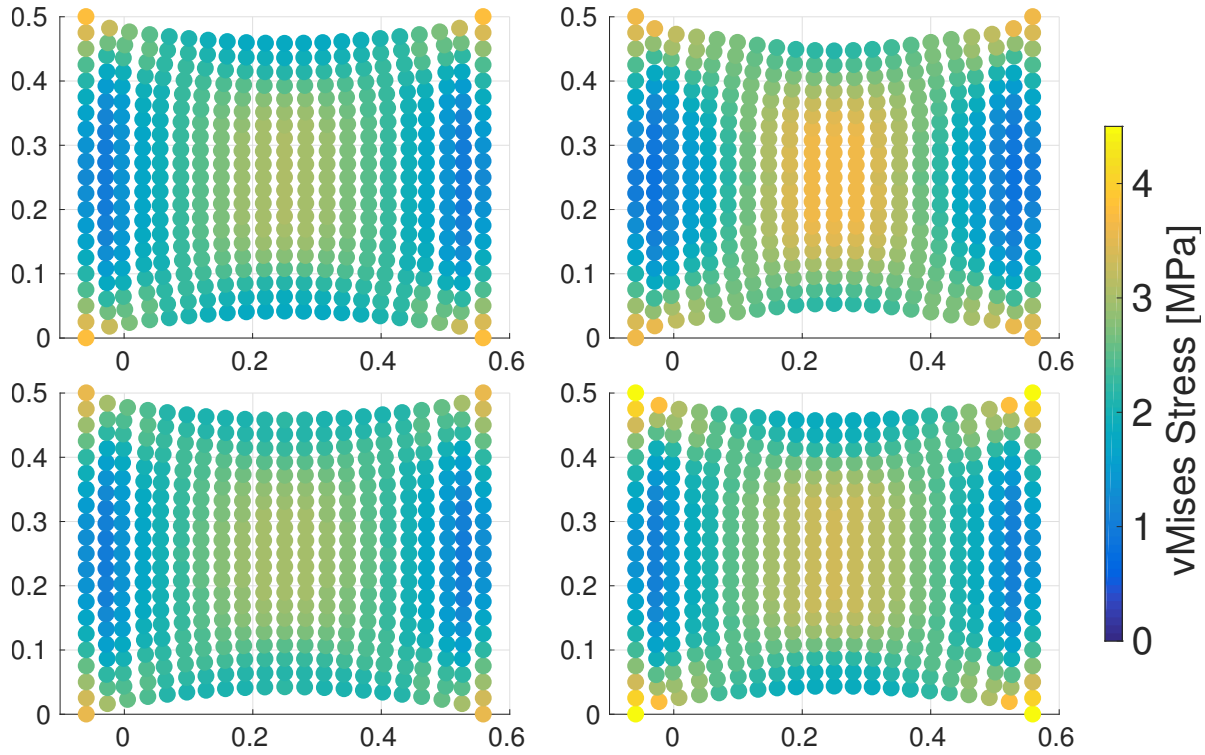

Figure 3.5: A tensile test carried out using the total Lagrangian framework described in [217] employing four different objective stress rates: Jaumann rate of deviatoric stress, Jaumann rate but using the full stress tensor without equation of state (3.70), Truessdell rate of deviatoric stress and Green Naghdi rate of deviatoric stress, starting from top left in clockwise order. Color indicated von Mises stress, scale is in Mpa.

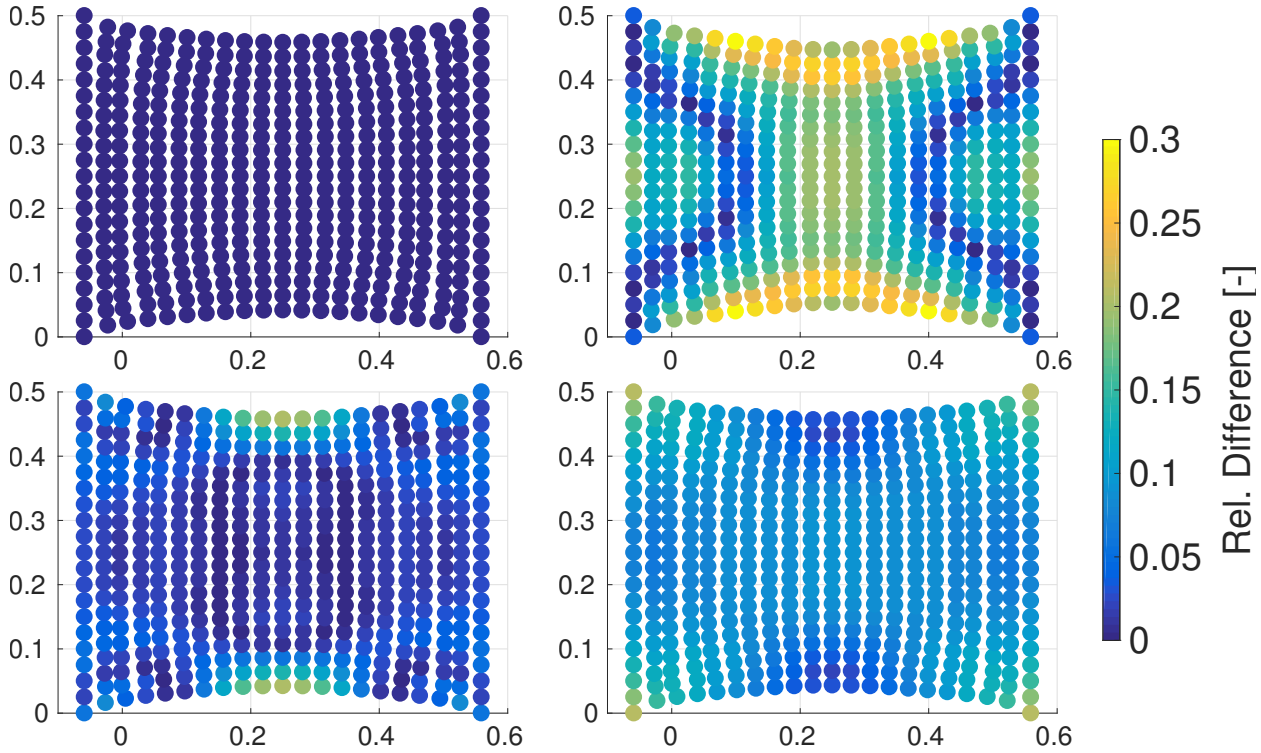

Figure 3.6: The same situation as in figure 3.5 but this time plotting the relative difference to the jaumann rate, which was arbitrarily chosen as reference. The differences exhibited by the different rates are quite severe, exceeding 30%.

9.3. A complete discussion about objective stress rates is beyond the scope of this thesis, but can be found in [272]. Still, a comparison simulation is given in figures 3.5 and 3.6. For repetition of the worked example involving the pre-stressed disk using these objective stress rates, see [25].

All of these rates correctly resolve rigid body rotations. In fact, it can be shown that they are equal under deformations that are purely rotational, see [25]. Furthermore, they only involve derivatives with regard to the current configuration. Thus, they fulfill the requirements demanded in the beginning of this chapter. However, it is customary to model the deviatoric part $\underline{\underline{S}}$ of the Cauchy stress tensor separately from the hydrostatic part p. The reason for this is speculated to be three-fold: on one hand it enables employment of a different constitutive model for the hydrostatic and deviatoric part, i.e.

$$\underline{\underline{S}} = \underline{\underline{\sigma}} - p \cdot \underline{\underline{I}} \tag{3.69}$$

This is for example required if a more involved approach than simple bulk behavior is to be modelled. This can for example be done by using the Mie-Grüneisen [163] [95] equation of state, which also takes the current temperature into account. This then dictates the hydrostatic behaviour while the usual Hooke's law can be applied to the deviatoric part. On the other hand, splitting volumetric effects from deviatoric ones is reported to be beneficial for FEM elements to prevent volume locking, see [179]. While volume locking is not present in meshless codes, the same models used in FEM were inherited due to code reuse, making them wide spread. Finally, some materials, e.g. metals, exhibit plastic strains under deviatoric loading only, i.e. are insensitive to hydrostatic stresses with regard to plasticity. Cleanly

isolating the model with regard to hydrostatic deformations prevents errors from spreading from one model to the next.

The most common equation of state would be the following one:

$$p = c_0^2(\varrho - \varrho_0) \tag{3.70}$$

It is not quite clear why this particular equation of state seems so popular. Investigating Hooke's law again, starting from (3.44):

$$\begin{aligned} \underline{\underline{\sigma}} &= 2G \cdot \underline{\underline{\varepsilon}} + \lambda \cdot \underline{\underline{I}} \cdot tr(\underline{\underline{\varepsilon}}) && (3.71)\\ &= 2G \cdot (dev(\underline{\underline{\varepsilon}}) + vol(\underline{\underline{\varepsilon}})) + 3\lambda \cdot vol(\underline{\underline{\varepsilon}}) && (3.72)\\ &= 2G \cdot dev(\underline{\underline{\varepsilon}}) + (2G \cdot +3\lambda \cdot) vol(\underline{\underline{\varepsilon}}) && (3.73) \end{aligned}$$

using $K = \lambda + (2/3) \cdot G$ yields

$$\underline{\underline{\sigma}} = 3K \cdot vol(\underline{\underline{\varepsilon}}) + 2G \cdot dev(\underline{\underline{\varepsilon}}) \tag{3.74}$$

Considering the volumetric part of this equation:

$$\underbrace{vol(\underline{\underline{\sigma}})}_{p} = 3K \cdot 1/3 tr(\underline{\underline{\varepsilon}}) = K \cdot tr(\underline{\underline{\varepsilon}}) \tag{3.75}$$

Assuming small strains $tr(\underline{\underline{\varepsilon}}) = V - V_0/V_0$ and using mass conservation $m = V_0 \cdot \varrho_0 = V \cdot \varrho$:

$$\begin{aligned} p = K \cdot (tr(\underline{\underline{\varepsilon}})) &= K \cdot \left(\frac{V - V_0}{V_0} \right) = \frac{K}{\varrho}(\varrho_0 - \varrho) \\ &= c(\varrho_0 - \varrho) \end{aligned} \tag{3.76}$$

Note the subtle difference between (3.70) and(3.76), i.e. c_0 vs. c which constitutes the difference in speed of sound in either the reference or deformed configuration and is negligible in case of small volumetric deformations.

The most used objective stress rate in general and in meshless literature in particular is the Jaumann rate. The final elastic constitutive update is thus summarized in accordance with the state of the art as follows:

$$\begin{aligned} p &= c(\varrho_0 - \varrho) \\ \dot{\underline{\underline{S}}} &= 2G \cdot \left(\underline{\underline{D}} - 1/3 tr\left(\underline{\underline{D}} \right) \right) + \underline{\underline{W}} \cdot \underline{\underline{S}} - \underline{\underline{S}} \cdot \underline{\underline{W}} \end{aligned} \tag{3.77}$$

This model can be implemented into a total Lagrangian code with relative ease, even though derivatives with respect to $\underline{x}$ are not available in this situation. This is achieved by using

$$\underline{\underline{L}} = \dot{\underline{\underline{F}}} \cdot \underline{\underline{F}}^{-1} \tag{3.78}$$

which follows from (3.56). This in turn can be used to compute $\underline{\underline{D}}$ and $\underline{\underline{W}}$ by means of (3.6). To solve (3.42) $\underline{\underline{P}}$ is needed, which can be found by $\underline{\underline{\sigma}} = \underline{\underline{S}} - p \cdot \underline{\underline{I}}$ and transformation (3.18).

3.4 PLASTICITY

The discussion so far was limited to elastic deformations, that is, deformations that recover if the specimen under consideration is deloaded. In metals this happens due to the fact that the stretched atomic lattice seeks to return to the lowest possible energy state after deloading. However, metals clearly experience irreversible deformations after they are sufficiently deformed, i.e. plastic deformations. This happens due to sliding of crystals in glide systems by motion of dislocations.

There are numerous plastic effects and the number of models describing these effects is vast, even just considering metal plasticity. Thus, only the most important concepts regarding plasticity will be introduced. For a more complete review regarding metal plasticity in general and metal plasticity in particular, the reader is referred to review papers such as [37], to classic texts such as [241] or the lecture notes [133], from which the following discussion is derived to present a closed and consistent theory. Only one specific flow stress model will be discussed in detail, the Johnson-Cook model [123], which is the de-facto standard in metal cutting simulation according to [9].

3.4.1 *Basic Definitions by Means of Uniaxial Tension*

Figure 3.7 shows various stress - strain relations. The simplest case displayed is the linear elastic one, which can be described using Hooke's law. Metals behave like this only up to very small strains. If this small threshold is exceeded the material begins to yield. If no change in σ is observed with increasing strain the material is called perfectly plastic. For metals this is a very crude assumption. Metals harden with increasing strain ε, i.e. the stress further increases. This is called strain hardening. The material may also behave differently depending on the strain rate $\dot{\varepsilon}$. Increased strain rates result in increased hardening. This is called visco-plasticity.

Besides strain hardening there are also other hardening effects which can't be observed in monotonic loading. To this end, figure 3.8 is presented. In this figure, the various possible behaviors after unloading are exhibited: Isotropic hardening is observed if the yielding in reverse direction only happens after the new yield limit $-\sigma'_{yield}$ is reached. The initial diameter of the elastic stress range increased from $2 \cdot \sigma_{yield}$ to $2 \cdot \sigma'_{yield}$. Alternatively, yielding may start even before $-\sigma_{yield}$ is reached. This is the Bauschinger effect [18] or kinematic hardening. The term kinematic hardening stems from the fact that the elastic regime keeps its original size $2 \cdot \sigma_y$, but shifts in space. This may be more illustrative in the multidimensional case, see 3.4.2. Additionally, there are materials that exhibit both kinematic and isotropic hardening, sometimes referred to as combined hardening. Another effect, so called formative or distortional hardening, can't be visualized in the uniaxial case and is subject of the next section.

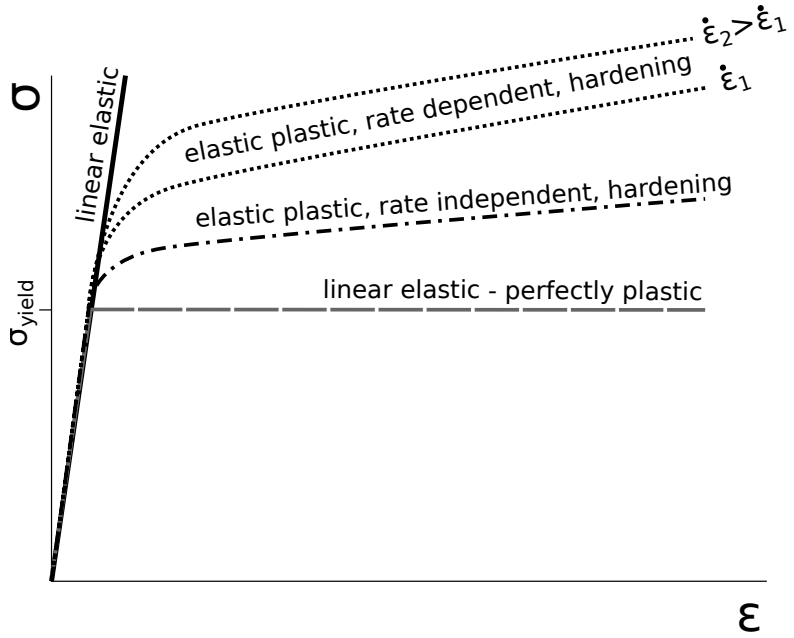

Figure 3.7: Various stress - strain relations: linear elastic (solid line), linear elastic - perfectly plastic (dashed line), elastic plastic with strain hardening (dash - dotted line) and elastic plastic with rate dependent strain hardening for two rates (dotted line)

3.4.2 *Multiaxial Case*

In the multiaxial case, the yield limit generalizes to a yield surface, which can, in principle, be any function of the six independent components of $\underline{\underline{\sigma}}$ and any number of material parameters $\underline{\beta}$:

$$F(\underline{\underline{\sigma}}, \underline{\beta}) = 0 \tag{3.79}$$

β will immediately be dropped from notation again and it is implicitly understood that $F(\cdot)$ depends on some material parameters. Since the yield surface should certainly not be influenced by the coordinate system chosen it is more convenient to write

$$F(\underline{\tilde{\sigma}}) = 0 \tag{3.80}$$

where $\underline{\tilde{\sigma}}$ is the principal stress state $\underline{\tilde{\sigma}} = [\sigma_1, \sigma_2, \sigma_3]$. This implies that the yield surface reduces from a six dimensional hyper surface to a surface in three dimensions and from a surface to a closed loop in two dimensions, simplifying geometrical considerations considerably.

Again, the stress state is not allowed to exceed yield surface $F(\cdot)$, but the yield surface may change depending on the hardening rules present, analogous to the one dimensional case, see 3.8. Figure 3.9 shows the same hardening rules as present in the one dimensional case: The yield surface changes in size in isotropic hardening, shifts in space in kinematic hardening and changes shape in formative, also called distortional hardening, which was not discussed in the one dimensional case since it can not be shown in the unaxial state. The material may also exhibit any combination of the hardening behaviors mentioned, e.g. kinematic, formative as well as isotropic hardening, which would entail shifting as well as scaling the yield surface. These combined cases are not displayed in the figure.

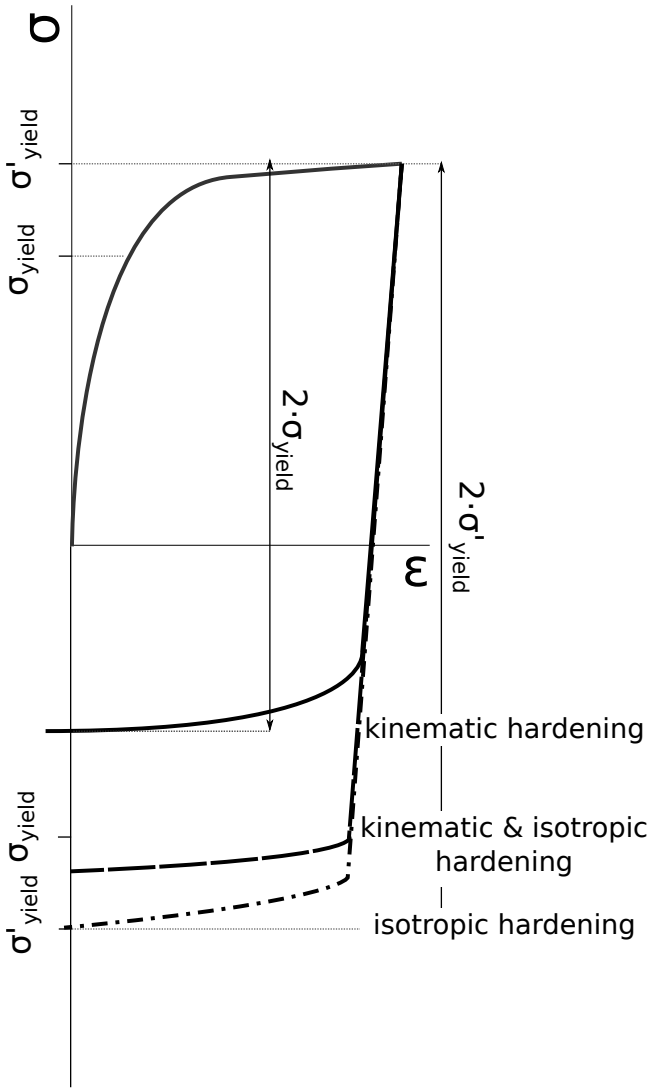

Figure 3.8: Various hardening effects in load reversal: isotropic hardening (dash dotted line), kinematic hardening (solid line) and kinematic isotropic hardening (dashed line).

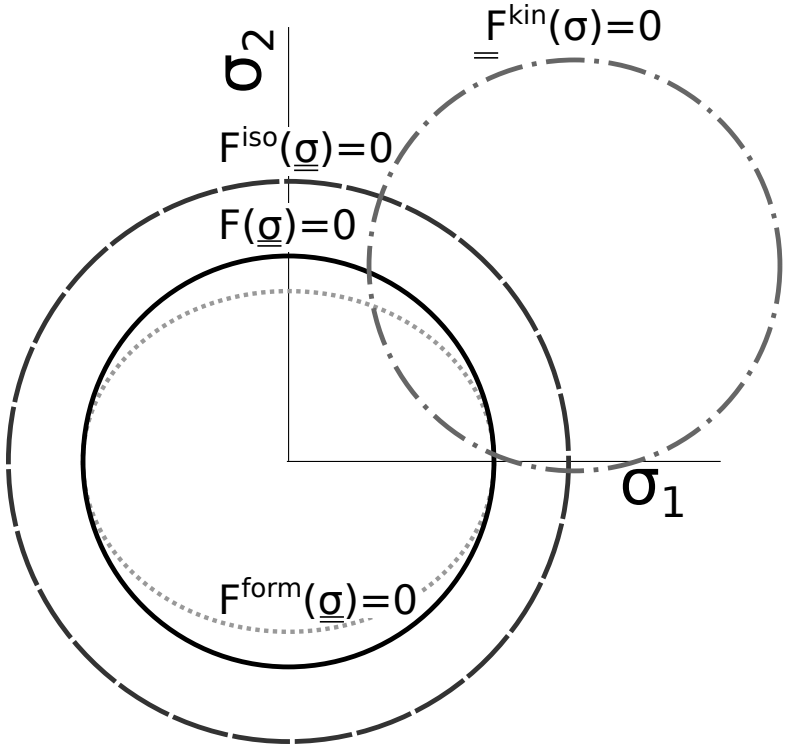

Figure 3.9: Various hardening effects in the biaxial case: isotropic hardening (dashed line), kinematic hardening (dash dotted line) and formative hardening (dotted line).

Even though the discussion about the various hardening effects was quite extensive, the focus is returned to the perfectly plastic case without any hardening for now. This is to simplify notation and the geometrical considerations, especially with regard to the figures presented.

Since the yield surface stays constant, a stress point may remain in one fixed position, or slide along the yield surface by redistribution of the stress components. Expressed mathematically:

$$F(\underline{\underline{\sigma}} + d\underline{\underline{\sigma}}) = \underbrace{F(\underline{\underline{\sigma}})}_{0} + \frac{\partial F(\underline{\underline{\sigma}})}{\partial \underline{\underline{\sigma}}} d\underline{\underline{\sigma}} = \nabla F(\underline{\underline{\sigma}}) d\underline{\underline{\sigma}} = 0 \tag{3.81}$$

or, put in different words, the stress increment is not allowed to move in direction of the outward normal to the yield surface, i.e. $\nabla F(\underline{\underline{\sigma}})$. (3.81) is called the consistency condition. This prompts the definition of the plastic strain increment $d\underline{\underline{\varepsilon}}^{\mathrm{pl}}$ caused by some amount of plastic loading. The plastic strain increment can be derived by some plastic potential $G(\cdot)$:

$$d\underline{\underline{\varepsilon}}_{\mathrm{pl}} = \frac{\partial G(\underline{\underline{\sigma}})}{\partial \underline{\underline{\sigma}}} G(\underline{\underline{\sigma}}) d\lambda_{\mathrm{pl}} = d\lambda_{\mathrm{pl}} \nabla G \tag{3.82}$$

where λ_{pl} is the plastic multiplier. For better understanding, the quantities are re-explained by geometrical means in figure 3.10.

Most often it is assumed that $G(\cdot) = F(\cdot)$, since it would be quite a demanding task to determine $G(.)$ experimentally if $G(\cdot) \neq F(\cdot)$. The case where $G(\cdot) = F(\cdot)$ is called associated flow rule. Notable examples of models with non-associated flow rule include works by

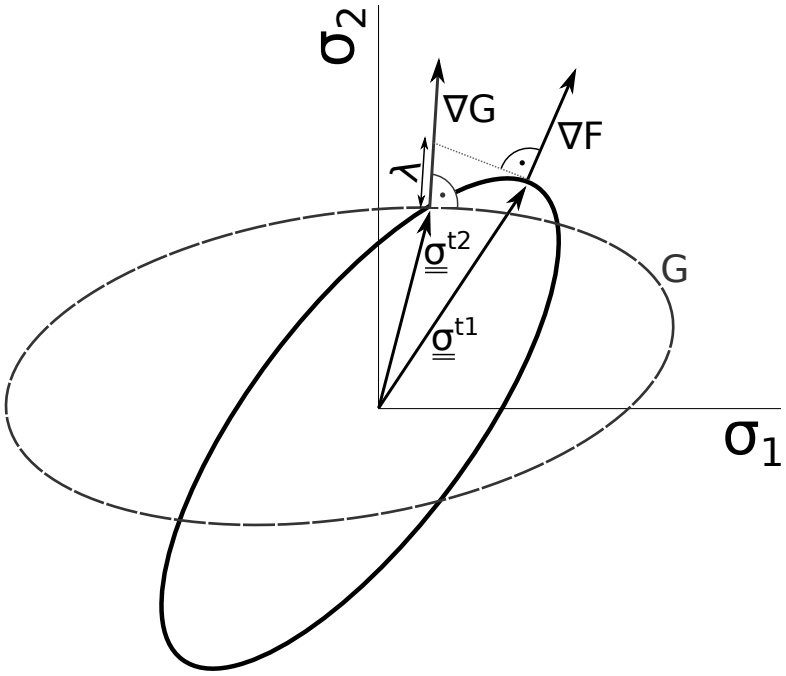

Figure 3.10: Plastic loading occurs from t_1 to t_2 with stress states $\underline{\underline{\sigma}}^{t1}$ and $\underline{\underline{\sigma}}^{t2}$. The plastic loading happens tangentially, i.e. normal to the normal to the yield surface ∇F. The stress state immediately returns to surface F, such that potential G is minimized, i.e. along its normal ∇G. The distance traveled along ∇G is the plastic multiplier λ_{pl}.

Mroz [175] and Krieg [134]. The associated flow rule entails some useful properties like that the yield surface needs to be convex. However, according to [133] it often fails to describe experimentally observed results, especially for materials other than metals. The discussion in this thesis is limited to the associated flow rule. (3.82) thus simplifies:

$$d\underline{\underline{\varepsilon}}_{\text{pl}} = \frac{\partial F(\underline{\underline{\sigma}})}{\partial \underline{\underline{\sigma}}} d\lambda_{\text{pl}} = d\lambda_{\text{pl}} \nabla F \tag{3.83}$$

which means that the plastic strain increment points normal to the yield surface.

So far the yield surface was kept as general as possible and allowed to depend on the full stress state $\underline{\tilde{\sigma}}$. It turns out that it is often more convenient to have $F(\cdot)$ depend on the invariants I_{1-3}

$$I_1 = tr(\underline{\underline{\sigma}}) \tag{3.84}$$
$$I_2 = 1/2(tr(\underline{\underline{\sigma}})^2 - tr(\underline{\underline{\sigma}}^2)) \tag{3.85}$$
$$I_3 = det(\underline{\underline{\sigma}}) \tag{3.86}$$

of the stress tensor $\underline{\underline{\sigma}}$ or the invariants J_{1-3}

$$J_1 = 0 \tag{3.87}$$
$$J_2 = 1/2(\underline{\underline{S}} : \underline{\underline{S}}) \tag{3.88}$$
$$J_3 = det(\underline{\underline{S}}) \tag{3.89}$$

of the deviatoric stress tensor $\underline{\underline{S}}$. A popular assumption for metals is that metal plasticity does not depend on the hydrostatic pressure, eliminating the first invariant from model considerations. It is often further assumed that the plastic state depends on J_2 only, hence the terms metal plasticity and J_2-plasticity are often, but spuriously, used interchangeably. It can be also be shown that these assumptions may be quite inappropriate in some situations, see for example [34], where Barlat and colleagues argue that J_3 needs to be considered as well. This work is limited to J_2 theory, however.

There are a lot of different ways to write J_2. Some care has to be taken regarding the exact form of these alternative expressions for J_2 since they depend on the form of the Cayley-Hamilton theorem [100] and tensor normalization used. Some convenient forms to be used in this text include:

$$J_2 = 1/2 \left(tr(\underline{\underline{S}})^2 - tr(\underline{\underline{S}}^2) \right) = \sqrt{1/2 \underline{\underline{S}} : \underline{\underline{S}}} = \sqrt{1/2} ||\underline{\underline{S}}|| \tag{3.90}$$

where the tensor norm $|| \cdot ||$ was defined. This also introduces the concept of the "direction" of a tensor, let the direction of some tensor $\underline{\underline{A}}$ be $\underline{\underline{A}}/||\underline{\underline{A}}||$.

The only yield criteria used in this work is of the following form:

$$F(J_2) = J_2 - k^2 = 0 \tag{3.91}$$

i.e. the von Mises criterion, where k is the yield strength in pure shear. This criterion will be generalized such that k does not need to be constant in the next section 3.4.3, i.e. isotropic hardening will be re-introduced.

Usually (3.91) is modified such that the yield strength in pure shear can be replaced by the yield strength in uniaxial tension $\sigma_{yield} = \sigma_y$. To this end, (3.91) is expanded and investigated:

$$J_2 = k^2 \tag{3.92}$$

$$\frac{(\sigma_{xx} - \sigma_{yy})^2 + (\sigma_{yy} - \sigma_{zz})^2 + (\sigma_{zz} - \sigma_{xx})^2 + 6(\sigma_{xy}^2 + \sigma_{yz}^2 + \sigma_{zx}^2)}{6} = k^2 \tag{3.93}$$

Setting everything but σ_{xy}, i.e. pure shear, zero yields indeed:

$$|\sigma_{xy}| = k \tag{3.94}$$

Now, applying uniaxial tension in x direction gives:

$$\frac{\sigma_{xx}}{\sqrt{3}} = k \tag{3.95}$$

thus, it is concluded if $k = \sigma_y$ in (3.91) then the yield surface would be a factor of $\sqrt{3}$ off. Correcting for this:

$$F(J_2) = \sqrt{3 \cdot J_2} - \sigma_y = 0 \tag{3.96}$$

$\sigma_{vM} = \sqrt{3 \cdot J_2}$ is called the equivalent stress or, more often, the von-Mises stress. It is convenient to define a scalar, equivalent quantity to the plastic strain $\underline{\underline{\varepsilon}}$ as well:

$$\bar{\varepsilon}_{\text{pl}} = \sqrt{2/3 \, \underline{\underline{\varepsilon}}_{\text{pl}} : \underline{\underline{\varepsilon}}_{\text{pl}}} \tag{3.97}$$

the factor of 2/3 is motivated by considering uniaxial tension again. The material is assumed isotropic and plastically incompressible, i.e. $tr(\underline{\underline{\varepsilon}}) = 0$. Hence:

$$\underline{\underline{\varepsilon}}_{\text{pl}} = \begin{bmatrix} \epsilon_{\text{pl}}^{xx} & 0 & 0 \\ 0 & -\frac{1}{2}\epsilon_{\text{pl}}^{xx} & 0 \\ 0 & 0 & -\frac{1}{2}\epsilon_{\text{pl}}^{xx} \end{bmatrix} \tag{3.98}$$

inserting into (3.97):

$$\bar{\underline{\underline{\varepsilon}}}_{\text{pl}} = \sqrt{2/3\left(\epsilon_{\text{pl}}^{xx2} + \left(1/2 \cdot \epsilon_{\text{pl}}^{xx}\right)^2 + \left(1/2 \cdot \epsilon_{\text{pl}}^{xx}\right)^2\right)} = \epsilon_{\text{pl}}^{xx} \tag{3.99}$$

thus, in the case of uniaxial tension, the equivalent plastic strain reduces to the plastic strain.

3.4.3 *Johnson-Cook Flow Stress Model and Radial Return*

In this section, the simple yield surface (3.96) will be generalized to models of the form:

$$F(J_2) = \sqrt{3 \cdot J_2} - \sigma_y(\bar{\varepsilon}_{\text{pl}}, \dot{\bar{\varepsilon}}_{\text{pl}}, T) = 0 \tag{3.100}$$

i.e. σ_y is now not a constant but depends on equivalent plastic strain $\bar{\varepsilon}_{\text{pl}}$, equivalent plastic strain rate $\dot{\bar{\varepsilon}}_{\text{pl}}$ and temperature T. As mentioned, only one such model will be considered, i.e. the Johnson-Cook model [123]:

$$\sigma_y(\bar{\varepsilon}_{\text{pl}}, \dot{\bar{\varepsilon}}_{\text{pl}}, T) = \left(A + B(\bar{\varepsilon}_{\text{pl}})^n\right) \cdot \left(1 + C \ln\left(\frac{\dot{\bar{\varepsilon}}_{\text{pl}}}{\dot{\bar{\varepsilon}}_{\text{pl}}^0}\right)\right) \cdot \left(1 - \left(\frac{T - T_0}{T_m - T_0}\right)^m\right) \tag{3.101}$$

Where A, B, C, n, m are the Johnson-Cook material parameters. $\dot{\bar{\varepsilon}}_{\text{pl}}^0$ is the equivalent plastic reference strain rate used in determining A, B and n. T, T_0 and T_m are current, reference and melting temperature. This model for the yield, or flow stress remains the most popular in metal cutting, see [9], even though it can be shown to behave quite wrong in some situations as demonstrated for classic benchmarks in [15], and even for cutting as shown in [98]. In terms of hardening effects it contains strain hardening as isotropic hardening but no kinematic hardening. It is also viscoplastic, i.e. dependent not only on the strain but also strain rate and respects thermal softening, see section 3.5.

In the following it is explained how such models can be implemented into computer codes. This is done by means of the radial return algorithm. The radial return algorithm was proposed by Wilkins [275] for simple cases such as (3.96). If the equation is nonlinear in λ_{pl}, however, references include Winnicki [277] and Wang [271], the latter in turn crediting the radial return to [191]. The most comprehensive description however is probably found in the user manual [97] to the `UINTAH` solver [85].

In order to apply the radial return algorithm to the situation at hand there is still one key result missing. Combining (3.83) with (3.100):

$$\begin{aligned} d\underline{\underline{\varepsilon}}_{\mathrm{pl}} &= \frac{\partial F(\underline{\underline{\sigma}})}{\partial \underline{\underline{\sigma}}} d\lambda_{\mathrm{pl}} = \frac{\partial \left[\sqrt{3 \cdot J_2} - \sigma_y(\bar{\varepsilon}_{\mathrm{pl}}, \dot{\bar{\varepsilon}}_{\mathrm{pl}}, T)\right]}{\partial \underline{\underline{\sigma}}} d\lambda_{\mathrm{pl}} = \frac{\partial \sqrt{3 \cdot J_2}}{\partial \underline{\underline{\sigma}}} d\lambda \\ &= \frac{\sqrt{3}}{2\sqrt{J_2}} \cdot \frac{\partial J_2}{\partial \underline{\underline{\sigma}}} d\lambda_{\mathrm{pl}} \end{aligned} \tag{3.102}$$

after expressing J_2 in terms of $\underline{\underline{\sigma}}$:

$$\begin{aligned} \frac{\partial J_2}{\partial \underline{\underline{\sigma}}} = \frac{\partial}{\partial \underline{\underline{\sigma}}} \Big\{ & 1/6 \cdot \left[(\sigma_{xx} - \sigma_{yy})^2 + (\sigma_{yy} - \sigma_{zz})^2 + (\sigma_{zz} - \sigma_{xx})^2 \right] \\ & + \sigma_{xy}^2 + \sigma_{yz}^2 + \sigma_{zx}^2 \Big\} \end{aligned} \tag{3.103}$$

respecting symmetry of $\underline{\underline{\sigma}}$:

$$\begin{aligned} \frac{\partial J_2}{\partial \underline{\underline{\sigma}}} = \frac{\partial}{\partial \underline{\underline{\sigma}}} \Big\{ & 1/6 \cdot \left[(\sigma_{xx} - \sigma_{yy})^2 + (\sigma_{yy} - \sigma_{zz})^2 + (\sigma_{zz} - \sigma_{xx})^2 \right] \\ & + 1/2 \cdot \left[\sigma_{xy}^2 + \sigma_{yx}^2 + \sigma_{yz}^2 + \sigma_{zy}^2 + \sigma_{zx}^2 + \sigma_{xz}^2 \right] \Big\} \end{aligned} \tag{3.104}$$

expanding the squares in the first bracket (note the factor of 2):

$$\begin{aligned} \frac{\partial J_2}{\partial \underline{\underline{\sigma}}} = \frac{\partial}{\partial \underline{\underline{\sigma}}} \Big\{ & 2/6 \cdot \left[\sigma_{xx}^2 + \sigma_{yy}^2 + \sigma_{zz}^2 - \sigma_{xx}\sigma_{yy} - \sigma_{yy}\sigma_{zz} - \sigma_{xx}\sigma_{zz} \right] \\ & + 1/2 \cdot \left[\sigma_{xy}^2 + \sigma_{yx}^2 + \sigma_{yz}^2 + \sigma_{zy}^2 + \sigma_{zx}^2 + \sigma_{xz}^2 \right] \Big\} \end{aligned} \tag{3.105}$$

the gradient can now be taken quite conveniently:

$$\frac{\partial J_2}{\partial \underline{\underline{\sigma}}} = \begin{bmatrix} (2\sigma_{xx} - \sigma_{yy} - \sigma_{zz})/3 & \sigma_{xy} & \sigma_{xz} \\ \sigma_{xy} & (2\sigma_{yy} - \sigma_{xx} - \sigma_{zz})/3 & \sigma_{yz} \\ \sigma_{xy} & \sigma_{yz} & (2\sigma_{zz} - \sigma_{xx} - \sigma_{yy})/3 \end{bmatrix} \tag{3.106}$$

by expanding the diagonal terms in the fashion $2\sigma_{xx} - \sigma_{yy} - \sigma_{zz} = 3\sigma_{xx} - \sigma_{xx} - \sigma_{yy} - \sigma_{zz}$ it can be seen that:

$$\frac{\partial J_2}{\partial \underline{\underline{\sigma}}} = \underline{\underline{\sigma}} - 1/3 \cdot tr(\underline{\underline{\sigma}}) = \underline{\underline{S}} \tag{3.107}$$

And finally by reinserting into (3.102) :

$$d\underline{\underline{\varepsilon}}_{\mathrm{pl}} = \sqrt{\frac{3}{2}} \frac{\underline{\underline{S}}}{||\underline{\underline{S}}||} d\lambda_{\mathrm{pl}} \tag{3.108}$$

since $\sqrt{J_2} = \sqrt{1/2} \cdot ||\underline{\underline{S}}||$, see for example [216]. Which means that the plastic strain increment points into the same direction as the deviatoric stress tensor.

The discussion is simplified in the following by assuming the simplest possible time stepper, i.e. Euler integration and omitting co-rotated derivatives as discussed in section 3.3, for brevity of notation. In this case, the elastic update reduces to:

$$\begin{aligned}\dot{\underline{\underline{S}}} &= 2G\left(dev(\dot{\underline{\underline{\varepsilon}}}) - dev(\dot{\underline{\underline{\varepsilon}}}_{\mathrm{pl}})\right)\\ &= 2G\left(dev(\dot{\underline{\underline{\varepsilon}}}) - \dot{\underline{\underline{\varepsilon}}}_{\mathrm{pl}}\right)\end{aligned} \tag{3.109}$$

where it was used again that metal plasticity is independent of the hydrostatic pressure. The principle of the radial return algorithm is now to compute the trial stress by assuming that there is no plastic strain $\dot{\underline{\underline{\varepsilon}}}_{\mathrm{pl}} = 0$:

$$\underline{\underline{S}}^{\mathrm{trial}} = \underline{\underline{S}}^{\mathrm{n}} + 2G \cdot dev(\dot{\underline{\underline{\varepsilon}}})\Delta t \tag{3.110}$$

Note that the analysis is continued in discretized time. As mentioned, Euler integration was employed. It is now checked whether this trial stress is admissible, i.e. if $F(\underline{\underline{S}}^{\mathrm{trial}}, \ldots) \leq 0$. Should this be the case the constitutive update is straight forward:

$$\begin{aligned}\underline{\underline{S}}^{\mathrm{n}+1} &= \underline{\underline{S}}^{\mathrm{trial}}\\ \bar{\varepsilon}_{\mathrm{pl}}^{n+1} &= \bar{\varepsilon}_{\mathrm{pl}}^{n}\\ \dot{\bar{\varepsilon}}_{\mathrm{pl}}^{n+1} &= 0\end{aligned} \tag{3.111}$$

Otherwise, the full update needs to be used:

$$\underline{\underline{S}}^{\mathrm{n}+1} = \underline{\underline{S}}^{\mathrm{n}} + \dot{\underline{\underline{S}}}\Delta t \tag{3.112}$$

by solving (3.110) for $\underline{\underline{S}}^{\mathrm{n}}$ and inserting it as well as (3.109) into the above (3.112):

$$\begin{aligned}\underline{\underline{S}}^{\mathrm{n}+1} &= \underline{\underline{S}}^{\mathrm{trial}} - 2G \cdot dev(\dot{\underline{\underline{\varepsilon}}})\Delta t + 2G\left(dev(\dot{\underline{\underline{\varepsilon}}}) - \dot{\underline{\underline{\varepsilon}}}_{\mathrm{pl}}\right)\Delta t\\ &= \underline{\underline{S}}^{\mathrm{trial}} - 2G \cdot \dot{\underline{\underline{\varepsilon}}}_{\mathrm{pl}}\Delta t\\ &= \underline{\underline{S}}^{\mathrm{trial}} - 2G \cdot \left(\sqrt{\frac{3}{2}}\frac{\underline{\underline{S}}^{n+1}}{||\underline{\underline{S}}^{n+1}||}\dot{\lambda}_{\mathrm{pl}}\right)\Delta t\end{aligned} \tag{3.113}$$

where in the last step result (3.108) was used. This can be rearranged:

$$\underline{\underline{S}}^{\mathrm{trial}} = \underline{\underline{S}}^{\mathrm{n}+1}\left[1 + \sqrt{\frac{3}{2}}\frac{\left(2G \cdot \underline{\underline{S}}^{n+1} \cdot \dot{\lambda}_{\mathrm{pl}} \cdot \Delta t\right)}{||\underline{\underline{S}}^{n+1}||}\right] \tag{3.114}$$

which implies that $\underline{\underline{S}}^{\mathrm{trial}}$ and $\underline{\underline{S}}^{\mathrm{n}+1}$ share the same direction:

$$\frac{\underline{\underline{S}}^{\mathrm{trial}}}{||\underline{\underline{S}}^{\mathrm{trial}}||} = \frac{\underline{\underline{S}}^{\mathrm{n}+1}}{||\underline{\underline{S}}^{\mathrm{n}+1}||} \tag{3.115}$$

this finding can now be used to turn (3.113) into a scalar equation:

$$\left[\frac{\underline{\underline{S}}^{\mathrm{trial}}}{||\underline{\underline{S}}^{\mathrm{trial}}||}\right]^{-1}\left[\underline{\underline{S}}^{\mathrm{trial}}\right] = \left[\underline{\underline{S}}^{\mathrm{n}+1} + 2G \cdot \left(\sqrt{\frac{3}{2}}\frac{\underline{\underline{S}}^{n+1}}{||\underline{\underline{S}}^{n+1}||}\dot{\lambda}_{\mathrm{pl}}\Delta t\right)\right]\left[\frac{\underline{\underline{S}}^{\mathrm{n}+1}}{||\underline{\underline{S}}^{\mathrm{n}+1}||}\right]^{-1}$$

$$||\underline{\underline{S}}^{\text{trial}}|| = ||\underline{\underline{S}}^{\text{n+1}}|| + \sqrt{3/2} \cdot 2G\dot{\lambda}_{\text{pl}}\Delta t \tag{3.116}$$

using that:

$$||\underline{\underline{S}}|| = \sqrt{2/3} \cdot \sigma_y(\bar{\varepsilon}_{\text{pl}}, \dot{\bar{\varepsilon}}_{\text{pl}}, T) \tag{3.117}$$

gives:

$$||\underline{\underline{S}}^{\text{trial}}|| = \sqrt{2/3} \cdot \sigma_y(\bar{\varepsilon}_{\text{pl}}, \dot{\bar{\varepsilon}}_{\text{pl}}, T) + \sqrt{3/2} \cdot 2G \cdot \dot{\lambda}_{\text{pl}}\Delta t \tag{3.118}$$

since $\bar{\varepsilon}_{\text{pl}}, \dot{\bar{\varepsilon}}_{\text{pl}}$ are functions of λ_{pl} and $\dot{\lambda}_{\text{pl}}$ respectively, this is a non-linear function in λ_{pl}. In a final step $\dot{\lambda}_{\text{pl}}\Delta t$ is set to $\Delta\lambda$ to arrive at a final objective function:

$$0 = ||\underline{\underline{S}}^{\text{trial}}|| - \sqrt{2/3} \cdot \sigma_y(\bar{\varepsilon}_{\text{pl}}, \dot{\bar{\varepsilon}}_{\text{pl}}, T) - \sqrt{3/2} \cdot 2G \cdot \Delta\lambda_{\text{pl}} = \mathcal{G}(\Delta\lambda_{\text{pl}}) \tag{3.119}$$

This can now be solved with any standard root finding method, e.g. using Newton iterations:

$$\Delta\lambda_{\text{pl}}^{j+1} = \Delta\lambda_{\text{pl}}^{j} - \frac{\mathcal{G}(\Delta\lambda_{\text{pl}}^{j})}{\frac{d}{d\Delta\lambda_{\text{pl}}^{j}}\mathcal{G}(\Delta\lambda_{\text{pl}}^{j})} \tag{3.120}$$

3.5 FRICTION AND THERMAL EFFECTS

The flow stress model discussed in section 3.4.3, i.e. the Johnson-Cook model depends on temperature T. In other words, it respects thermal softening. This was so far neglected from the discussion but shall be elaborated in this section. In isotropic solids heat conducts according to the Fourier heat equation:

$$\dot{T} = \alpha^{\text{trml}}\nabla\left(\nabla T\right) + \frac{q}{\varrho \cdot c_p} \tag{3.121}$$

where α^{trml} is the thermal diffusivity, which was assumed to be independent of temperature T, and c_p the specific heat capacity. The usual notation $\Delta = \nabla \cdot \nabla$ shall not be used in this work since Δ is already used to denote an increment. q can consist of one or more source terms. Having explained how heat is distributed throughout the body simulated it remains to clarify how heat is introduced, or removed from the body. There are two effects considered to introduce heat. The first one is by plastic work. If there is plastic loading anywhere in the body, that work is converted to heat:

$$q^{\text{plast}} = t_q \cdot \dot{\varepsilon}_{pl} \cdot \sigma_Y(\varepsilon_{pl}, \dot{\varepsilon}_{pl}, T) \tag{3.122}$$

where t_q is the Taylor-Quinney coefficient. The specific heat capacity c_p relates to thermal diffusivity α^{trml} with relation:

$$\alpha^{\text{trml}} = \frac{k}{\varrho \cdot c_p} \tag{3.123}$$

With thermal conductivity k. The conversion of plastic work to heat is often called "adiabatic heating" in literature. The other aspect is heat introduced by friction. The friction model used in this work is the simplest one available, i.e. Coulomb friction:

$$|\underline{f}_{\text{fric}}| = \mu |\underline{f}_{\text{cont}}| \tag{3.124}$$

with Coulomb friction parameter μ, friction force $\underline{f}_{\text{fric}}$ and $\underline{f}_{\text{cont}}$ acting normal on the contacting surface. The frictional work is just $\underline{f}_{\text{fric}} \cdot |v_{\text{rel}}|$. The heat flow rate due to this work is:

$$\dot{Q}^{\text{fric}} = \eta \, |\underline{f}_{\text{fric}}| \cdot |v_{\text{rel}}| \tag{3.125}$$

where η is a dimensionless parameter determining how much of the frictional work is turned into heat, similar to the Taylor-Quinney coefficient, see [117]. The presence of this parameter is phsically questionable, at best. That is, $\eta = 1$ from physical considerations. However, its precense in popular simulation packages like ABAQUS, see [105] necessitates its precense in the software implemented as well, in order to be able to conduct comparative simulations. Expressed as a source term:

$$q^{\text{fric}} = \frac{\varrho \cdot \dot{Q}^{\text{fric}}}{m} \tag{3.126}$$

Where m is the mass of the body receiving the temperature jump. Only two kinds of thermal boundary conditions are considered, either a thermal sink of constant temperature or perfect isolation.

The complete model hence reads:

$$\dot{T} = \alpha^{\text{trml}} \nabla \left(\nabla T \right) + \frac{q^{\text{plast}} + q^{\text{fric}}}{\varrho \cdot c_p} \tag{3.127}$$

4

MESHLESS METHODS

This chapter discusses the meshless methods used throughout this work. The presentation is made from a function interpolation perspective, i.e. how meshless methods can be used to interpolate function values and derivatives from given, scattered samples. The connection to how the PDEs in the previous chapter can be solved using the methods at hand is made in the next chapter. This is to cleanly separate the mathematical foundation of the meshless methods form the physics at hand.

First, Smoothed Particle Hydrodynamics is derived. Next higher order schemes are discussed, followed by a short discussion about higher order derivatives. Afterwards the choice of smoothing length, that is, how much smoothing should be employed by the *Smoothed* Particle Hydrodynamics, is considered. The last two sections concern themselves with numerical instabilities exhibited by some meshless methods as well as correction schemes to counteract said instabilities.

4.1 DERIVATION OF THE SPH

Even though there are methods older than the SPH which can be reasonably called "meshless", like molecular dynamics, the SPH is the first meshless method which truly discretizes the continuum in the same sense the FEM does. The method was introduced by Lucy, Gingold and Monaghan [154] [87]. Originally devised for celestial dynamics it was quickly adopted for a wide range of applications. Just to highlight how diverse it's application are some works concerning fluid flows [148], explosions [147] [149], magnetic fields [63] and solid mechanics [142] [144] are mentioned here. This section is not intended to give a complete publication history regarding SPH nor a comprehensive review regarding its theory. The reader is referred to the excellent accounts in [205] and [171], or text books such as [140]. However, the most important cornerstones regarding its derivation and approximation errors are derived in the following.

As stated, the SPH shall be discussed as a means to interpolate some function $f(\cdot)$ and its derivative(s) at a location $\underline{x}$ given some scattered samples of $f(\cdot)$. Simply evaluating the function $f(\cdot)$ at $\underline{x}$ can be stated using the Dirac sampling theorem:

$$f(\underline{x}) = \int_\Omega f(\underline{x}')\delta(\underline{x}' - \underline{x})d\underline{x}' \tag{4.1}$$

Since

$$\delta(\underline{x}) = \begin{cases} \infty & \text{for} \quad \underline{x} = 0 \\ 0 & \text{for} \quad \underline{x} \neq 0 \end{cases} \tag{4.2}$$

this identity can't be used to interpolate the function at locations $\tilde{\underline{x}} \neq \underline{x}$. To fix this, the Dirac delta function is replaced by a smooth function W, called kernel function:

$$f(\underline{x}) \approx \int_\Omega f(\underline{x}')W\left(\frac{\underline{x}' - \underline{x}}{h}\right) d\underline{x}' \tag{4.3}$$

Which is often shortened to $W_h(\underline{x}' - \underline{x})$. The parameter h is called smoothing length and describes the size of the domain of $W(\cdot)$. In SPH, the sampling points $f(\underline{x}')$ are called particles. Often, not a single function f is defined on the particles but a number of functions referring to physical quantities like mass, density or velocity. Furthermore, the neighborhood of a particle at $\underline{x}'$ is defined as the particles which reside in the domain of $W_h(\underline{x}' - \underline{x})$.

The expression (4.3) prompts the question how exact the approximation is in relation to the properties of W_h. Assuming one spatial dimension for now the above can be written as:

$$f(x) \approx \int_{-\infty}^{+\infty} \left[f(x) + \frac{\partial f}{\partial x}|_x(x' - x) + \frac{1}{2}\frac{\partial f}{\partial x^2}|_x(x' - x)^2 + \dots\right] \times W_h(x' - x)dx' \tag{4.4}$$

by expanding $f(x')$ into its Taylor series. This entails that if W_h fulfils the first r conditions of the form

$$\begin{aligned}
\int_{-\infty}^{+\infty} W_h(x' - x)dx' = 1 &= m_0 \\
\int_{-\infty}^{+\infty} W_h(x' - x) \cdot (x' - x)dx' = 0 &= m_1 \\
\int_{-\infty}^{+\infty} W_h(x' - x) \cdot (x' - x)^2dx' = 0 &= m_2 \\
\dots & \\
\int_{-\infty}^{+\infty} W_h(x' - x) \cdot (x' - x)^r dx' = &= m_r
\end{aligned} \tag{4.5}$$

expression (4.3) converges with order $\mathcal{O}\left((x - x')^r\right)$. This error is called the mollification error and the conditions above are called the moment conditions. These moments coincide with the moments of the Dirac delta function. The first three conditions allow for some interesting observations:

- The zeroth moment condition necessitates that the kernel function always integrates to one
- The first moment condition necessitates that the kernel is symmetric about x, since $(x' - x)$ is antisymmetric, and only a symmetric function yields zero if convoluted with a antisymmetric one.
- The second moment would necessitate that $W_h(x' - x)$ needs to have negative parts since $(x' - x)^2$ is strictly positive. Kernels with this property are rarely used for physically based simulations since some field variables like density or mass can only take positive values. An unfavourable particle arrangement might then lead to approximation of negative values of mass or density in some locations.

The zeroth and first moment condition combined ensure that W_h converges to the Dirac delta function as h approaches zero:

$$\lim_{h \to 0} W_h(x) = \delta(x) \tag{4.6}$$

All of these facts also hold in three spatial dimensions, and the derivations above could be repeated in the same manner, albeit with cumbersome notation. A more concise proof regarding three dimensions of the same results is given in [51].

While (4.3) is clearly suitable to approximate functions values given scattered data it remains to be clarified how to interpolate $\nabla f(\underline{x})$. While it holds that

$$[\nabla f(\underline{x})] \approx \int_\Omega \left[\nabla f(\underline{x}')\right] W_h(\underline{x}' - \underline{x}) d\underline{x}' \tag{4.7}$$

this formula would necessitate that $\nabla f(\underline{x})$ be given at the sample locations. The idea is now to shift the gradient from $f(\cdot)$ to the kernel function. This can be achieved by expanding $\nabla \cdot f(\underline{x}')W_h(\underline{x}' - \underline{x})$ by the product rule:

$$[\nabla f(\underline{x})] \cdot W_h(\underline{x}' - \underline{x}) = \nabla \left[f(\underline{x}) \cdot W_h(\underline{x}' - \underline{x})\right] - f(\underline{x}') \cdot \left[\nabla W_h(\underline{x} - \underline{x}')\right] \tag{4.8}$$

Reinserting into (4.7) and using the divergence theorem:

$$\begin{aligned}
[\nabla f(\underline{x})] &\approx \int_\Omega \nabla \left[f(\underline{x}) \cdot W_h(\underline{x}' - \underline{x})\right] d\underline{x}' - \int_\Omega f(\underline{x}') \cdot \left[\nabla W_h(\underline{x} - \underline{x}')\right] d\underline{x}' \\
&= \int_S \left[f(\underline{x}) \cdot W_h(\underline{x}' - \underline{x})\right] \cdot \underline{n}\, d\underline{x}' - \int_\Omega f(\underline{x}') \cdot \left[\nabla W_h(\underline{x} - \underline{x}')\right] d\underline{x}'
\end{aligned} \tag{4.9}$$

where S is the surface of Ω and $\underline{n}$ is the surface normal. This entails that W_h has to be chosen such that it vanishes on S:

$$\nabla f(\underline{x}) \approx \int_\Omega f(\underline{x}') \cdot \nabla W_h(\underline{x} - \underline{x}') d\underline{x}' \tag{4.10}$$

In other words: This means that $W_h(\cdot)$ has to have compact support. Typical kernels feature a support of $c \cdot h$ with small c. Consequently, the mollification error can be restated as:

$$\mathcal{O}\left(h^r\right) \tag{4.11}$$

Now, enough conditions for the Kernel function have been revealed to think about its specific shape. In addition to the conditions of symmetry and normality revealed from the moment conditions (4.5), the approximation to the derivative (4.10) necessitates that the kernel must have a smooth first derivative. A natural choice might be the Gaussian, since it is normal and symmetric, and is not only smooth in its first, but any number of derivatives. In fact, the Gaussian is a viable kernel if its domain is clamped to a few h. Other choices that do not necessitate the evaluation of an exponential, which is numerically quite costly, are more popular, see (4.31).

To implement equations (4.3) or (4.10) into a computer program the integral has to be replaced by some numerical quadrature. Usually a Riemann sum approach is taken, giving the following discretized equations:

$$< f(\underline{x}) > = \sum_{j=1}^{N} f(\underline{x}_j) \cdot W_h(\underline{x} - \underline{x}_j)\ \omega_j \tag{4.12}$$

$$< \nabla f(\underline{x}) > = \sum_{j=1}^{N} f(\underline{x}_j) \cdot \nabla W_h(\underline{x} - \underline{x}_j)\ \omega_j \tag{4.13}$$

where the interpolation operator has been introduced as $< \cdot >$ and ω_j are the integration weights. These expressions also reveal that the compact support requirement of W_h not only enables the estimation of derivatives but also brings the method in the realm of computational feasibility; would W_h be of long range evaluating the interpolation at M locations would mean a runtime complexity of $\mathcal{O}(N^2)$ if $M \propto N$, which is prohibitively expensive even for moderately large N.

Besides the mollification error discussed, replacing the integral with numerical quadrature introduces yet another source of error, the quadrature or discretization error. This error shall be investigated in a simplified setting, assuming one spatial dimension and a uniform particle spacing. The proof which follows is a reworked and more verbose form of the one given in [209], where a proof for a disordered particle array in one dimension is also given. For three spatial dimension, consider [51]. The proof starts with the second Euler-MacLaurin formula:

$$\Delta x \sum_{j=1}^{N} g_j = \int_{x_1 - \Delta x/2}^{x_n + \Delta x/2} g(x)dx + \sum_{k=1}^{\infty} \frac{B_{2k} \Delta x^{2k}}{(2k)!}(1 - 2^{-2k+1}) \left(\frac{\partial^{2k-1} g}{\partial x^{2k-1}}|_{x_{n+1/2}} - \frac{\partial^{2k-1} g}{\partial x^{2k-1}}|_{x_{1-1/2}} \right) \tag{4.14}$$

where B_{2k} are the Bernoulli numbers. The Bernoulli numbers increase non-monotonically with k but not as fast as $2k!$ in the denominator and $x_{j+1/2}$ is defined as $x + j \cdot \Delta x + 1/2 \cdot \Delta x$. The general idea is now to set $g(\cdot) = f(\tilde{x}) \frac{\partial W_h(x - \tilde{x})}{\partial x} = f(\tilde{x}) W_h'(x - \tilde{x}) =< \frac{\partial f(x)}{\partial x} >$. x' has been renamed $\tilde{x}$ for the following proof to avoid confusion with the shorthand for the derivative:

$$\sum_{j=1}^{N} f(x_j) W_h'(x - x_j) \Delta x = \int_{x - c \cdot h}^{x + c \cdot h} f(\tilde{x}) W_h'(x - \tilde{x}) d\tilde{x} + \sum_{k=1}^{\infty} \frac{B_{2k} \Delta x^{2k}}{(2k)!}(1 - 2^{-2k+1}) \times \underbrace{\left(\frac{\partial^{2k-1} f(\tilde{x}) W_h'(x - \tilde{x})}{\partial \tilde{x}^{2k-1}}|_{x + c \cdot h} - \frac{\partial^{2k-1} f(\tilde{x}) W_h'(x - \tilde{x})}{\partial \tilde{x}^{2k-1}}|_{x - c \cdot h} \right)}_{\mathcal{S}} \tag{4.15}$$

The integration boundaries have been adapted using that W_h has compact support. That is, integrating from $-\infty \ldots \infty$ is the same as integrating from $x - c \cdot h \ldots x + c \cdot h$. The term $\mathcal{S}$ on the right hand side is the boundary smoothness of W_h which is defined as the highest integer β such that the β^{th} derivative vanishes at the edges of the compact support. If W_h has boundary smoothness β then W_h' has boundary smoothness $\beta - 1$. Thus, the term $\mathcal{S}$ is

zero for all $2k-1 \leq \beta-1$. Anticipating later results β is assumed even. The proof could be repeated for odd β requiring only little modification. Thus, the first non zero term $\mathcal{S}$ in the series (4.15) has $2k = \beta+2$:

$$\begin{aligned}\sum_{j=1}^{N} f(x_j)W_h'(x-x_j)\Delta x &= \int_{x-c\cdot h}^{x+c\cdot h} f(\tilde{x})W_h'(x-\tilde{x})d\tilde{x} \\ &+\Delta x^{\beta+2}\frac{B_{\beta+2}}{(\beta+2)!}(1-2^{-\beta-1}) \times \left(\frac{\partial^{\beta+1} f(\tilde{x})W_h'(x-\tilde{x})}{\partial \tilde{x}^{\beta+1}}|_{x+c\cdot h} - \frac{\partial^{\beta+1} f(\tilde{x})W_h'(x-\tilde{x})}{\partial \tilde{x}^{\beta+1}}|_{x-c\cdot h}\right) \\ &+\mathcal{O}(\Delta x^{\beta+4}) \end{aligned} \tag{4.16}$$

The term $\mathcal{S}$ of this order is now investigated: $f(\tilde{x})$ in $\partial f(\tilde{x})W_h'(x-\tilde{x})/\partial x^{\beta+1}$ is developed into its Taylor series about point x:

$$\begin{aligned}\frac{\partial^{\beta+1} f(\tilde{x})W_h'(x-\tilde{x})}{\partial \tilde{x}^{\beta+1}} &= [f(x)W_h'(x-\tilde{x}) + \frac{\partial f(\tilde{x})}{\partial \tilde{x}}|_x W_h'(x-\tilde{x})(\tilde{x}-x) \\ &+ \frac{\partial^2 f(\tilde{x})}{2\partial \tilde{x}^2}|_x W_h'(x-\tilde{x})(\tilde{x}-x)^2 + \underbrace{\mathcal{O}((x-\tilde{x})^3)}_{\mathcal{O}(\Delta x^3)}]\frac{\partial^{\beta+1}}{\partial \tilde{x}^{\beta+1}} \\ &= f(x)\frac{\partial^{\beta+2} W_h(x-\tilde{x})}{\partial \tilde{x}^{\beta+2}} \\ &+ \frac{\partial f(\tilde{x})}{\partial \tilde{x}}|_x \left[(\tilde{x}-x)\frac{\partial^{\beta+2} W(x-\tilde{x})}{\partial \tilde{x}^{\beta+2}} + (\beta+1)\frac{\partial^{\beta+1} W(x-\tilde{x})}{\partial \tilde{x}^{\beta+1}}\right] \\ &+ \frac{\partial^2 f(\tilde{x})}{2\partial \tilde{x}^2}|_x \left[(\tilde{x}-x)^2\frac{\partial^{\beta+2} W(x-\tilde{x})}{\partial \tilde{x}^{\beta+2}}\right. \\ &\left.+ 2(\tilde{x}-x)(\beta+1)\frac{\partial^{\beta+1} W(x-\tilde{x})}{\partial \tilde{x}^{\beta+1}} + (\beta+1)\frac{\partial^{\beta+2} W(x-\tilde{x})}{\partial \tilde{x}^{\beta+2}}\right] \end{aligned} \tag{4.17}$$

Where the second equality is by "pulling in" of the partial derivative $\frac{\partial^{\beta+1}}{\partial \tilde{x}^{\beta+1}}$. For example:

$$f(x)W_h'(x-\tilde{x}) \cdot \frac{\partial^{\beta+1}}{\partial \tilde{x}^{\beta+1}} = f(x)\frac{\partial^{\beta+2} W_h(x-\tilde{x})}{\partial \tilde{x}^{\beta+2}} \tag{4.18}$$

And so on. Now (4.17) is analyzed further:

$$A(\tilde{x}) = f(x)\frac{\partial^{\beta+2} W_h(x-\tilde{x})}{\partial \tilde{x}^{\beta+2}} \tag{4.19}$$

$$B(\tilde{x}) = \frac{\partial f(\tilde{x})}{\partial \tilde{x}}|_x \left[(\tilde{x}-x)\frac{\partial^{\beta+2} W(x-\tilde{x})}{\partial \tilde{x}^{\beta+2}} + (\beta+1)\frac{\partial^{\beta+1} W(x-\tilde{x})}{\partial \tilde{x}^{\beta+1}}\right] \tag{4.20}$$

$$\begin{aligned}C(\tilde{x}) = \frac{\partial^2 f(\tilde{x})}{2\partial \tilde{x}^2}|_x &\left[(\tilde{x}-x)^2\frac{\partial^{\beta+2} W(x-\tilde{x})}{\partial \tilde{x}^{\beta+2}}\right. \\ &\left.+ 2(\tilde{x}-x)(\beta+1)\frac{\partial^{\beta+1} W(x-\tilde{x})}{\partial \tilde{x}^{\beta+1}} + (\beta+1)\frac{\partial^{\beta+2} W(x-\tilde{x})}{\partial \tilde{x}^{\beta+2}}\right] \end{aligned} \tag{4.21}$$

The terms A to C are now investigated in turn at the integration boundaries from (4.16):

$$\begin{aligned} A(\tilde{x})|_{x+c\cdot h} - A(\tilde{x})|_{x-c\cdot h} &= f(x)\frac{\partial^{\beta+2}W_h(c\cdot h)}{\partial\tilde{x}^{\beta+2}} - f(x)\frac{\partial^{\beta+2}W_h(-c\cdot h)}{\partial\tilde{x}^{\beta+2}} \\ &= f(x)\frac{\partial^{\beta+2}W_h(c\cdot h)}{\partial\tilde{x}^{\beta+2}} - f(x)\frac{\partial^{\beta+2}W_h(c\cdot h)}{\partial\tilde{x}^{\beta+2}} \\ &= 0 \end{aligned} \tag{4.22}$$

since W_h is symmetric and β was assumed even.

$$\begin{aligned} &B(\tilde{x})|_{x+c\cdot h} - B(\tilde{x})|_{x-c\cdot h} \\ &\quad= \frac{\partial f(\tilde{x})}{\partial\tilde{x}}|_x \left[c\cdot h\frac{\partial^{\beta+2}W(c\cdot h)}{\partial\tilde{x}^{\beta+2}} + (\beta+1)\frac{\partial^{\beta+1}W(c\cdot h)}{\partial\tilde{x}^{\beta+1}}\right] \\ &\quad- \frac{\partial f(\tilde{x})}{\partial\tilde{x}}|_x \left[-c\cdot h\frac{\partial^{\beta+2}W(-c\cdot h)}{\partial\tilde{x}^{\beta+2}} + (\beta+1)\frac{\partial^{\beta+1}W(-c\cdot h)}{\partial\tilde{x}^{\beta+1}}\right] \\ &\quad= 2\frac{\partial f(\tilde{x})}{\partial\tilde{x}}|_x \left[c\cdot h\frac{\partial^{\beta+2}W(c\cdot h)}{\partial\tilde{x}^{\beta+2}} + (\beta+1)\frac{\partial^{\beta+1}W(c\cdot h)}{\partial\tilde{x}^{\beta+1}}\right] \end{aligned} \tag{4.23}$$

since any odd derivative of W_h is antisymmetric

$$\begin{aligned} &C(\tilde{x})|_{x+c\cdot h} - C(\tilde{x})|_{x-c\cdot h} = \\ &\quad\frac{\partial^2 f(\tilde{x})}{2\partial\tilde{x}^2}|_x \left[(c\cdot h)^2\frac{\partial^{\beta+2}W(c\cdot h)}{\partial\tilde{x}^{\beta+2}} + \right. \\ &\qquad\left. 2(c\cdot h)(\beta+1)\frac{\partial^{\beta+1}W(c\cdot h)}{\partial\tilde{x}^{\beta+1}} + (\beta+1)\frac{\partial^{\beta+2}W(c\cdot h)}{\partial\tilde{x}^{\beta+2}}\right] \\ &\quad-\frac{\partial^2 f(\tilde{x})}{2\partial\tilde{x}^2}|_x \left[(-c\cdot h)^2\frac{\partial^{\beta+2}W(-c\cdot h)}{\partial\tilde{x}^{\beta+2}} + \right. \\ &\qquad\left. 2(-c\cdot h)(\beta+1)\frac{\partial^{\beta+1}W(-c\cdot h)}{\partial\tilde{x}^{\beta+1}} + (\beta+1)\frac{\partial^{\beta+2}W(-c\cdot h)}{\partial\tilde{x}^{\beta+2}}\right] \\ &\quad= 0 \end{aligned} \tag{4.24}$$

Reinserting the only remaining term into (4.15):

$$\begin{aligned} \sum_{j=1}^{N} f(x_j)W_h'(x-x_j)\Delta x = \int_{x-c\cdot h}^{x+c\cdot h} f(\tilde{x})W_h'(x-\tilde{x})d\tilde{x} + \Delta x^{\beta+2}\frac{B_{\beta+2}}{(\beta+2)!}(1-2^{-\beta-1}) \\ \times\left(2\frac{\partial f(\tilde{x})}{\partial\tilde{x}}|_x \left[c\cdot h\frac{\partial^{\beta+2}W(c\cdot h)}{\partial\tilde{x}^{\beta+2}} + (\beta+1)\frac{\partial^{\beta+1}W(c\cdot h)}{\partial\tilde{x}^{\beta+1}}\right]\right) \\ + \mathcal{O}(\Delta x^3) + \mathcal{O}(\Delta x^{\beta+4}) \end{aligned} \tag{4.25}$$

since

$$\frac{\partial^{\beta+2}W((x-\tilde{x})/h)}{\partial^{\beta+2}\tilde{x}} = \frac{1}{h^{\beta+3}}\frac{\partial\hat{W}(x-\tilde{x})}{\partial\tilde{x}} \tag{4.26}$$

which can be seen from the chain rule. I.e. for any function $f(\cdot)$:

$$\frac{\partial f(x/h)}{\partial x} = \frac{1}{h}\left[\frac{\partial}{\partial x}f(x/h)\right] \tag{4.27}$$

It remains to determine the asymptote of:

$$\underbrace{\Delta x^{\beta+2}\frac{B_{\beta+2}}{(\beta+2)!}(1-2^{-\beta-1})}_{A}\times\underbrace{\left(2\frac{\partial f(\tilde{x})}{\partial\tilde{x}}|_x\left[c\cdot h\frac{\partial^{\beta+2}W(c\cdot h)}{\partial\tilde{x}^{\beta+2}}+(\beta+1)\frac{\partial^{\beta+1}W(c\cdot h)}{\partial\tilde{x}^{\beta+1}}\right]\right)}_{B} \tag{4.28}$$

Since the Bernoulli numbers do not grow as fast as the factorial in the denominator the term is dominated by $\Delta x^{\beta+2}$. Hence $A\in\mathcal{O}\left(\Delta x^{\beta+2}\right)$. And $B\in\mathcal{O}\left(h^{-(\beta+2)}\right)$ because of 4.26. Thus, the complete discretization error is of order:

$$\sum_{j=1}^{N}f(x_j)W_h'(x-x_j)\Delta x=\int_{-\infty}^{+\infty}f(\tilde{x})W_h'(x-\tilde{x})d\tilde{x}+\mathcal{O}\left(\left(\frac{\Delta x}{h}\right)^{\beta+2}\right) \tag{4.29}$$

Where the original integration limits have been reinserted, and higher order asymptotic terms were dropped from notation.

This means that the error is only bound if $h>\Delta x$. Or, put in words, the particles have to overlap. Combining the discretization and the mollification error allows for an interesting observation:

$$\epsilon^{\text{tot}}=\mathcal{O}(h^r)+\mathcal{O}\left(\left(\frac{\Delta x}{h}\right)^{\beta+2}\right) \tag{4.30}$$

The first term decreases the error if h is decreased while the second term gives a strict lower limit for h, making the choice of h an interesting one. This will be discussed in section 4.4.

Having established the most important convergence / discretization all that is left to determine is the specific shape of the kernel function W_h. The most popular choice is the cubic spline [58]:

$$W_h(\underline{x}-\underline{x}')=n_c\times\begin{cases}1-\frac{3}{2}q^2+\frac{3}{4}q^3, & 0\le q<1\\ \frac{1}{4}(2-q)^3, & 1\le q\le 2\\ 0, & \text{otherwise}\end{cases} \tag{4.31}$$

with $q=|\underline{x}-\underline{x}'|/h$. It is the only kernel function considered in this thesis except for (4.74). An overview over different options for the choice of the kernel function and their performance is for example given in [82] and [114].

4.2 NON-UNIQUENESS OF SPH APPROXIMATORS

A natural question to emerge is whether (4.3) or (4.10) are the only possible SPH approximators with regard to the function value or gradient, respectively. It turns out that this is not the case. For example, by subtracting a null term from (4.10):

$$\begin{aligned} < \nabla f(\underline{x}) > &= \int_{-\infty}^{+\infty} f(\underline{x}')\nabla W_h(\underline{x}-\underline{x}')dx' - f(\underline{x}) \underbrace{\int_{-\infty}^{+\infty} \nabla W_h(\underline{x}-\underline{x}')dx'}_{0} \\ &= \int_{-\infty}^{+\infty} (f(\underline{x}')-f(\underline{x}))\nabla W_h(\underline{x}-\underline{x}')dx' \\ &\approx \sum_{j=1}^{N} (f(\underline{x}_j)-f(\underline{x}))\nabla W_h(\underline{x}-\underline{x}_j)\omega_j \end{aligned} \tag{4.32}$$

The null term on the right hand side is zero because the integral over the whole domain of an anti-symmetric function is always zero. Another approach to construct yet another SPH approximator for the gradient is to consider $\nabla f(\underline{x})/\varrho(\underline{x})$, expand it by the product rule and isolate the term containing $\nabla f(\underline{x})$:

$$\begin{aligned} \nabla\left(\frac{f(\underline{x})}{\varrho(\underline{x})}\right) &= \frac{\varrho(\underline{x})\cdot(\nabla f(\underline{x}))}{\varrho(\underline{x})^2} - \frac{f(\underline{x})\cdot(\nabla\varrho(\underline{x}))}{\varrho(\underline{x})^2} \\ \frac{(\nabla f(\underline{x}))}{\varrho(\underline{x})} &= \nabla\left(\frac{f(\underline{x})}{\varrho(\underline{x})}\right) + \frac{f(\underline{x})\cdot(\nabla\varrho(\underline{x}))}{\varrho(\underline{x})^2} \end{aligned} \tag{4.33}$$

using the "standard" SPH approximator for the gradient (4.10) for the two gradients on the right hand side:

$$\begin{aligned} &\nabla\left\langle\frac{f(\underline{x})}{\varrho(\underline{x})}\right\rangle + \frac{f(\underline{x})\cdot < \nabla\varrho(\underline{x}) >}{\varrho(\underline{x})^2} \\ &= \sum_{j=1}^{N} \frac{f(\underline{x}_j)}{\varrho(\underline{x}_j)} W_h(\underline{x}-\underline{x}_j)\frac{m(\underline{x}_j)}{\varrho(\underline{x}_j)} + \frac{f(\underline{x})}{\varrho(\underline{x})^2}\cdot\sum_{j=1}^{N}\varrho(\underline{x}_j)\nabla W_h(\underline{x}-\underline{x}_j)\frac{m(\underline{x}_j)}{\varrho(\underline{x}_j)} \\ &= \sum_{j=1}^{N} \frac{f(\underline{x}_j)}{\varrho(\underline{x}_j)^2} W_h(\underline{x}-\underline{x}_j)m(\underline{x}_j) + \sum_{j=1}^{N}\frac{f(\underline{x}_i)}{\varrho(\underline{x}_i)^2}\nabla W_h(\underline{x}-\underline{x}_j)m\underline{x}_j \\ &= \sum_{j=1}^{N}\left(\frac{f(\underline{x}_j)}{\varrho(\underline{x}_j)^2} + \frac{f(\underline{x}_i)}{\varrho(\underline{x}_i)^2}\right)\nabla W_h(\underline{x}-\underline{x}_j)m(\underline{x}_j) \end{aligned} \tag{4.34}$$

where the integration weight was related to the particle volume, that is $\omega_j = m(\underline{x}_j)/\varrho(\underline{x}_j) = m_j/\varrho_j$ with m being the particle mass and ϱ being the particle density. This approach is used in SPH to discretize a continuum in a physical context, see chapter 5.

In fact, it turns out that there are an infinite number of SPH approximators, at least for the gradient: It remains debatable how many of these infinite number of SPH approximators are meaningful options. For what its worth, in [140] it is shown that

$$\nabla\left\langle\frac{f(\underline{x})}{\varrho(\underline{x})}\right\rangle = \sum_{j=1}^{N}\left(\frac{f(\underline{x}_j)}{\varrho(\underline{x}_i)^{2-\zeta}\varrho(\underline{x}_j)^{\zeta}} + \frac{f(\underline{x}_i)}{\varrho(\underline{x}_i)^{\zeta}\varrho(\underline{x}_j)^{2-\zeta}}\right)\nabla W_h(\underline{x}-\underline{x}_j)\frac{m(\underline{x}_j)}{\varrho(\underline{x}_j)} \tag{4.35}$$

is a valid SPH approximator for any $\zeta \geq 0$.

Construction of these different approximation formulas serve different purposes. For example (4.32) turns out to retrieve the correct null gradient for constant functions, i.e. it is zero order complete, even in its discretized form. This is because $f(\underline{x}_j) - f(\underline{x}_i) = 0$ for constant functions. It is also anti symmetric about $\underline{x}$ and $\underline{x}_j$ whereas (4.34) is symmetric. These considerations are important when using SPH not only in a function approximation context but in order to solve PDEs and are further discussed in chapter 5.

4.3 CORRECTED KERNELS

Even if a kernel is designed to fulfill any number of moment conditions (4.5) it will fulfill these conditions only in an infinite domain, even before discretizing the integral into a Riemann sum. Since the SPH shall be applied to simulate continua with a well defined boundary it is worthwhile to scrutinize what kind of errors are introduced near and on the boundaries of the domain to be simulated.

Beside the convergence with regard to choice of Δx and h another quality measure of a kernel is the ability of a kernel to reproduce a polynomial of some degree exactly. This is called the "completeness" of a kernel. Investigation of the completeness of the SPH approximation (4.12) reveals that it is not even zeroth order complete, see figure 4.1.

So called corrected kernels are available to address this issue. In this section three such kernels will be discussed.

- The Reproducing Kernel Particle Method (RKPM)
- The Corrective Smoothed Particle Method (CSPM)
- The Randles Libersky correction

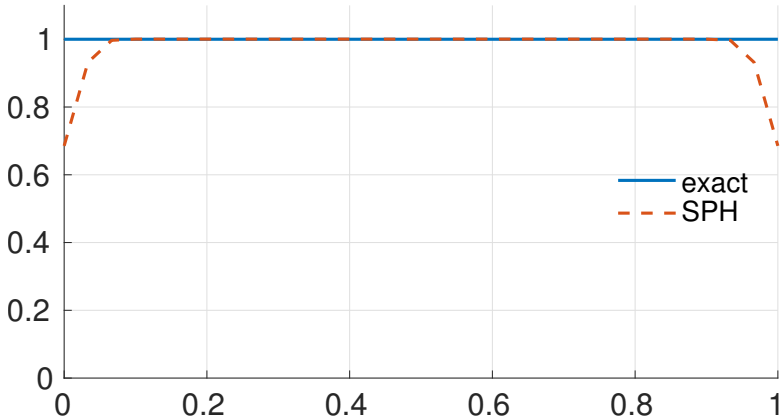

Figure 4.1: Reconstruction of a constant function using SPH. The boundary deficiency of the SPH, and thus lack of zeroth order completeness are clearly visible.

It will be shown that the latter two are equivalent. Some convergence results will be presented to demonstrate the effectiveness of these corrections. The discussion is limited to linear, also called first order, complete kernels. Higher order Kernels are possible which each correction scheme, however, due to the reasons discussed, loss of positivity, see the discussion just below (4.5) that is, the discussion is restricted to first order.

4.3.1 *Reproducing Kernel Particle Method*

The RKPM was first introduced by Liu et al. in 1995 in [151]. The method has seen application to a wide range of physical problems. Some notable examples include structural dynamics [152], soft tissues under very large deformations [127], CFD [150] and metal forming[40]. An improved version of the RKPM, called Gradient Reproducing Kernel Particle Method [103] (GRKPM) exists. However, this method assumes the gradient of the field function to be known, complicating the implementation of the method into common PDEs, especially on the strong form.

The idea of the RKPM is to enhance the kernel W_h in (4.12) with an additional correction function $\psi(\cdot)$:

$$< f(\underline{x}) > = \int_\Omega f(\underline{x}')W_h(\underline{x}-\underline{x}')d\underline{x}' \tag{4.36}$$

$$= \int_\Omega f(\underline{x}') \underbrace{\psi(\underline{x},\underline{x}')\cdot W_h(\underline{x}-\underline{x}')}_{K(\underline{x},\underline{x}')} d\underline{x}' \tag{4.37}$$

The following Ansatz is chosen for $\psi(\cdot)$:

$$\psi(\underline{x},\underline{x}') = \left[C_0(\underline{x}) + \underline{C}_1(\underline{x})\cdot(\underline{x}-\underline{x}')\right] \tag{4.38}$$

with unknown coefficients C_0 and coefficient vector $\underline{C}_1$. The corrected kernel $K(\cdot)$ is now inserted into the first two moment conditions (4.5):

$$1 = \int_\Omega K(\underline{x},\underline{x}')d\underline{x}' \tag{4.39}$$

$$0 = \int_\Omega (\underline{x}-\underline{x}')\cdot K(\underline{x},\underline{x}')d\underline{x}' \tag{4.40}$$

The first condition is rewritten by expanding $K(\cdot)$:

$$\begin{aligned} 1 &= \int_\Omega \left[C_0(\underline{x}) + \underline{C}_1(\underline{x})(\underline{x}-\underline{x}')\right] W_h(\underline{x}-\underline{x}')d\underline{x}' \\ &= C_0(\underline{x})\int_\Omega W_h(\underline{x}-\underline{x}')d\underline{x}' + \underline{C}_1(\underline{x})\int_\Omega (\underline{x}-\underline{x}')W_h(\underline{x}-\underline{x}')d\underline{x}' \\ &= C_0(\underline{x})\, m_0 + \underline{C}_1(\underline{x})\, \underline{m}_1 \end{aligned} \tag{4.41}$$

The same approach is taken for the second condition:

$$\begin{aligned} 0 &= \int_\Omega (\underline{x}-\underline{x}')\left[C_0(\underline{x}) + \underline{C}_1(\underline{x})(\underline{x}-\underline{x}')\right] W_h(\underline{x}-\underline{x}')d\underline{x}' \\ &= C_0(\underline{x})\int_\Omega (\underline{x}-\underline{x}')W_h(\underline{x}-\underline{x}')d\underline{x}' + \underline{C}_1(\underline{x})\int_\Omega (\underline{x}-\underline{x}')\otimes(\underline{x}-\underline{x}')W_h(\underline{x}-\underline{x}')d\underline{x}' \\ &= C_0(\underline{x})\, \underline{m}_1 + \underline{C}_1(\underline{x})\, \underline{\underline{m}}_2 \end{aligned} \tag{4.42}$$

Note that the moment conditions number one and two are vectors and matrices of degree 2 (3) in two (three) dimensions, respectively. This yields a linear system of equations with 3 (4) unknowns:

$$\begin{aligned} \begin{bmatrix} 1 \\ \underline{0} \end{bmatrix} &= \begin{bmatrix} C_0 \\ \underline{C}_1 \end{bmatrix} \begin{bmatrix} m_0 & \underline{m}_1^T \\ \underline{m}_1 & \underline{\underline{m}}_2 \end{bmatrix} \\ \underline{P} &= \underline{C}(\underline{x}) \underline{\underline{M}}(\underline{x}) \end{aligned} \tag{4.43}$$

This covers the approximation of function values. An approximator for the derivative with regard to some spatial coordinate x can be found by consecutively taking the derivative starting from (4.37)

$$\frac{\partial}{\partial x} K(\underline{x}, \underline{x}') = \left(\frac{\partial}{\partial x} \psi(\underline{x}, \underline{x}') \right) W(\underline{x} - \underline{x}') + \psi(\underline{x}, \underline{x}') \left(\frac{\partial}{\partial x} W(\underline{x} - \underline{x}') \right) \tag{4.44}$$

To obtain $\frac{\partial}{\partial x} \psi(\underline{x}, \underline{x}')$ the derived coefficients $\frac{\partial}{\partial x} \underline{C}(\underline{x})$ are required. To find these, the system above (4.43) is derived with regard to x:

$$\begin{aligned} \frac{\partial}{\partial x} \underline{P} &= \frac{\partial}{\partial x} (\underline{C}(\underline{x}) \underline{\underline{M}}(\underline{x})) \\ 0 &= \left(\frac{\partial}{\partial x} \underline{C}(\underline{x}) \right) \cdot \underline{\underline{M}}(\underline{x}) + \underline{C}(\underline{x}) \cdot \left(\frac{\partial}{\partial x} \underline{\underline{M}}(\underline{x}) \right) \end{aligned} \tag{4.45}$$

This procedure is repeated for $\frac{\partial}{\partial y}$ (and $\frac{\partial}{\partial z}$) to arrive at $\nabla K(\cdot)$.

It remains to note that the RKPM can be shown to be mathematically equivalent to the Moving Least Sqaures kernel [120]. This means that so called Element Free Galerkin (EFG), see [22], methods are in fact RKPM methods in "disguise".

The RKPM Kernel will be denoted $\overset{\circ}{W}$ in this thesis. Figure 4.2 presents some RKPM kernels. Note how the additional kernel mass towards the boundary of the domain compensates for the mass that would be cut off in SPH.

4.3.2 *The Randles Libersky Correction*

The Randles-Libersky correction was first mentioned in [214] but only properly derived in [143]. It has been incorporated into a Total Lagrangian SPH code in [217] and was part of the review given in [210]. Also, quite relevant to this thesis, it was also employed for metal cutting recently in [183]. The Randles Libersky correction operates on the gradient only.

To derive the Randles Libersky correction the right hand side of (4.32) is enhanced with a to be determined second order correction tensor $\underline{\underline{B}}$:

$$< \nabla f(\underline{x}) > = \left[\sum_{j=1}^{N} (f(\underline{x}_j) - f(\underline{x}_i) \nabla W_h(\underline{x}_i - \underline{x}_j) \omega_j \right] \cdot \underline{\underline{B}} \tag{4.46}$$

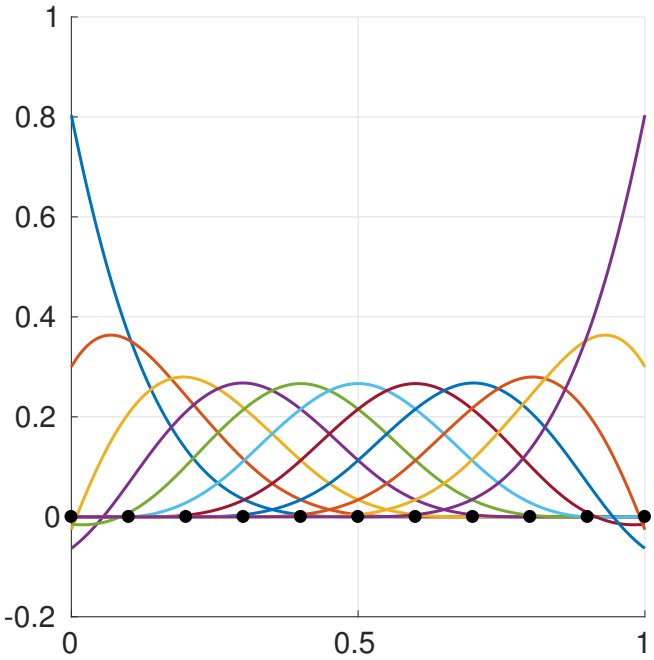

Figure 4.2: Some RKPM kernels in the interval $[0, 1]$. Particle locations are indicated in black.

Since a linear complete corrector is wanted, a linear function $f = \underline{\underline{A}} \cdot \underline{x} + \underline{b}$ with $\underline{\underline{A}}$ being the diagonal coefficient matrix is chosen for $f(\cdot)$. This is inserted into above and demanded to be equal to the analytical gradient $\nabla f = \underline{\underline{A}}$:

$$\begin{aligned}
\underline{\underline{A}} &= \left[\sum_{j=1}^{N}((\underline{\underline{A}} \cdot \underline{x}_i + \underline{b}) - (\underline{\underline{A}} \cdot \underline{x}_j + \underline{b})) \otimes \nabla W_h(\underline{x}_i - \underline{x}_j)\, \omega_j\right] \cdot \underline{\underline{B}} \\
\underline{\underline{A}} &= \underline{\underline{A}} \left[\sum_{j=1}^{N}(\underline{x}_i - \underline{x}_j) \otimes \nabla W_h(\underline{x}_i - \underline{x}_j)\, \omega_j\right] \cdot \underline{\underline{B}} \\
\underline{\underline{1}} &= \left[\sum_{j=1}^{N}(\underline{x}_i - \underline{x}_j) \otimes \nabla W_h(\underline{x}_i - \underline{x}_j)\, \omega_j\right] \cdot \underline{\underline{B}}
\end{aligned} \tag{4.47}$$

$\underline{\underline{B}}$ can now be written in compact form:

$$\underline{\underline{B}} = \left[\sum_{j=1}^{N}(\underline{x}_i - \underline{x}_j) \otimes \nabla W_h(\underline{x}_i - \underline{x}_j)\, \omega_j\right]^{-1} \tag{4.48}$$

Some authors [210, 217] prefer to replace W by its Shephard interpolant [236]:

$$\tilde{W}_h(\underline{x} - \underline{x}') = \frac{W_h(\underline{x} - \underline{x}')}{\int_\Omega W_h(\underline{x} - \underline{x}')d\underline{x}'} \tag{4.49}$$

This interpolant is readily derived by developing $f(\underline{x}')$ into its Taylor series about $\underline{x}$ and immediately cutting at the linear term:

$$f(\underline{x}') = f(\underline{x}) + \mathcal{O}(\underline{x}' - \underline{x}) \tag{4.50}$$

This is multiplied by the kernel $W_h(\cdot)$ and integrated over the whole domain:

$$\int_\Omega W_h(\underline{x}-\underline{x}')f(\underline{x}')d\underline{x}' = f(\underline{x})\int_\Omega W_h(\underline{x}-\underline{x}')d\underline{x}' + \mathcal{O}(\underline{x}'-\underline{x}) \tag{4.51}$$

and solved for $f(\underline{x})$:

$$f(\underline{x}) = \frac{1}{\int_\Omega W_h(\underline{x}-\underline{x}')d\underline{x}'}\int_\Omega W_h(\underline{x}-\underline{x}')f(\underline{x}')d\underline{x}' + \mathcal{O}(\underline{x}'-\underline{x}) \tag{4.52}$$

Its not clear why chosing $\tilde{W}_h(\cdot)$ over $W_h(\cdot)$ is beneficial. It is speculated that the smoothing properties of $\tilde{W}_h(\cdot)$ are beneficial due to its property of smoothing spurious osicllatory modes, see [192, 193].

4.3.3 *The Corrective Smoothed Particle Method and Equivalence to the Randles Libersky Correction*

The CSPM was first presented under that name in [42]. It has since seen application in quite a few applications like impact computations [70], heat conduction [41] or electro magnetics [199].

To derive the CSPM a Taylor series expansion about $\underline{x}$ is used again:

$$f(x',y') = f(x,y) + (x'-x)\frac{\partial f(x',y')}{\partial x'}|_{(x,y)} + (y'-y)\frac{\partial f(x',y')}{\partial y'}|_{(x,y)} + \mathcal{O}(\underline{x}-\underline{x}') \tag{4.53}$$

with $\underline{x} = [x,y]$. Note that the derivation is restricted to two dimensions here. This is merely for brevity of the presentation. The metodology extends to three dimensions in straight foward fashion.

Again, an approximator for the gradient that is linearly complete is sought after, hence the cut off after the linear element in the Taylor series expansion above. Two equations are devised by multiplying with $\frac{\partial W_h}{\partial x}$ and $\frac{\partial W_h}{\partial y}$ respectively, then integrating over the whole domain:

$$\begin{aligned}
&\int_\Omega \frac{\partial W}{\partial x}(f(x',y') - f(x,y))d\underline{x}' \\
&\qquad = \frac{\partial f(x',y')}{\partial x'}|_{(x,y)}\int_\Omega (x'-x)\frac{\partial W}{\partial x}d\underline{x}' + \frac{\partial f(x',y')}{\partial y'}|_{(x,y)}\int_\Omega (y'-y)\frac{\partial W}{\partial x}d\underline{x}' \\
&\int_\Omega \frac{\partial W}{\partial y}(f(x',y') - f(x,y))d\underline{x}' \\
&\qquad = \frac{\partial f(x',y')}{\partial x'}|_{(x,y)}\int_\Omega (x'-x)\frac{\partial W}{\partial y}d\underline{x}' + \frac{\partial f(x',y')}{\partial y'}|_{(x,y)}\int_\Omega (y'-y)\frac{\partial W}{\partial y}d\underline{x}'
\end{aligned} \tag{4.54}$$

Note that the notation has been shorted somewhat for better legibility: $W_h(\underline{x} - \underline{x}_j)$ was contracted to W. This is discretized using the usual Riemann sum approach:

$$\sum_{j=1}^{N} \frac{\partial W}{\partial x}(f(x_j, y_j) - f(x, y))\omega_j = \frac{\partial f(x', y')}{\partial x'}|_{(x,y)} \sum_{j=1}^{N}(x_j - x)\frac{\partial W}{\partial x}\omega_j + \frac{\partial f(x', y')}{\partial y'}|_{(x,y)} \sum_{j=1}^{N}(y_j - y)\frac{\partial W}{\partial x}\omega_j \quad (4.55)$$

$$\sum_{j=1}^{N} \frac{\partial W}{\partial y}(f(x_j, y_j) - f(x, y))\omega_j = \frac{\partial f(x', y')}{\partial x'}|_{(x,y)} \sum_{j=1}^{N}(x_j - x)\frac{\partial W}{\partial y}\omega_j + \frac{\partial f(x', y')}{\partial y'}|_{(x,y)} \sum_{j=1}^{N}(y_j - y)\frac{\partial W}{\partial y}\omega_j \quad (4.56)$$

rewriting the above in compact matrix form retrieves the Randles Libersky correction presented previously:

$$\underbrace{\left[\sum_{j=1}^{N}(\underline{x} - \underline{x}_j) \otimes \nabla W_h(\underline{x} - \underline{x}_j)\omega_j\right]}_{\underline{\underline{B}}^{-1}} < \nabla f >= \left[\sum_{j=1}^{N}(f(\underline{x}_j) - f(\underline{x}))\nabla W_h(\underline{x} - \underline{x}_j)\omega_j\right] \quad (4.57)$$

While equal in the end result the Randles Libersky correction and the CSPM certainly differ in their mathematical approach: The CSPM is the result of multiple simultaneous Taylor series expansions while the Randles Libersky correction introduces a correction tensor, then fitting that correction tensor in such a fashion that it exactly reconstructs a linear test function. It is a matter of opinion which approach one might consider more elegant. However, the choice of approach is not without consequence: Following the CSPM methodology the approximator is uniquely defined; it is not possible to replace W by $\tilde{W}$. Conceptually, it is straight foward to construct kernels of higher order using the CSPM approach by cutting the Talyor expansion only after a higher order term. The Randles Libersky correction offers no such systematic approach.

In this work the Randles Libersky correction is chosen, employing the Shephard interpolant $\tilde{W}$. Kernels constructed in such a fashion will be denoted $\overset{\triangle}{W}$.

4.3.4 *Some Numerical Studies*

This section is intended to highlight some of the results discussed in the previous section and is not intended as a complete parameter study regarding the numerical properties of the methods discussed. Instead, only two situations will be investigated:

- Reconstruction of the gradient of a linear function using a regular particle ensemble
- Convergence of the approximation of the gradient of a trigonometric function in both regular and irregular particle ensembles

The first test is supposed to exhibit the linear completeness of the corrected schemes. The second test is arguably the more interesting one: It is challenging since a transcendental function is approximated by an algebraic basis; recall that W_h is an algebraic function, i.e. a cubic (polynomial) spline. It further permits some insight on how convergence results derived on the continuous, infinite domain may or may not translate to the discretized, finite domain.

For the first example the domain $[0,1] \times [0,1]$ is discretized using 16×16 particles, with uniform integration weights $\omega_j = 1/(16 \cdot 16)$ and $h = 1.7 \cdot \Delta x$, Δx being the particle spacing. The function $f(x,y) = x$ is to be approximated on this domain by SPH, the Randles Libersky correction and RKPM. Figure 4.3 shows $|\partial/(\partial x) f(x,y) - \langle \partial/(\partial x) f(x,y) \rangle|$ for the three schemes. The boundary deficiency of the SPH is clearly visible, resulting in errors of up to 50% at the corners. The Randles Libersky correction and the RKPM essentially exhibit a null error. The error of the Randles Libersky correction was below the machine resolution while the RKPM error fluctuates with a maximum amplitude of 12 machine epsilon.

The machine epsilon denotes the smallest constant ϵ such that $1 + \epsilon > 1$ on a given computer architecture. While the expression is obviously only true for $\epsilon = 0$ in the continuum, in floating point math $1 + \epsilon$ is rounded off at some point. $\epsilon \sim 2 \cdot 10^{-16}$ for double and $\epsilon \sim 1 \cdot 10^{-7}$ for single precision.

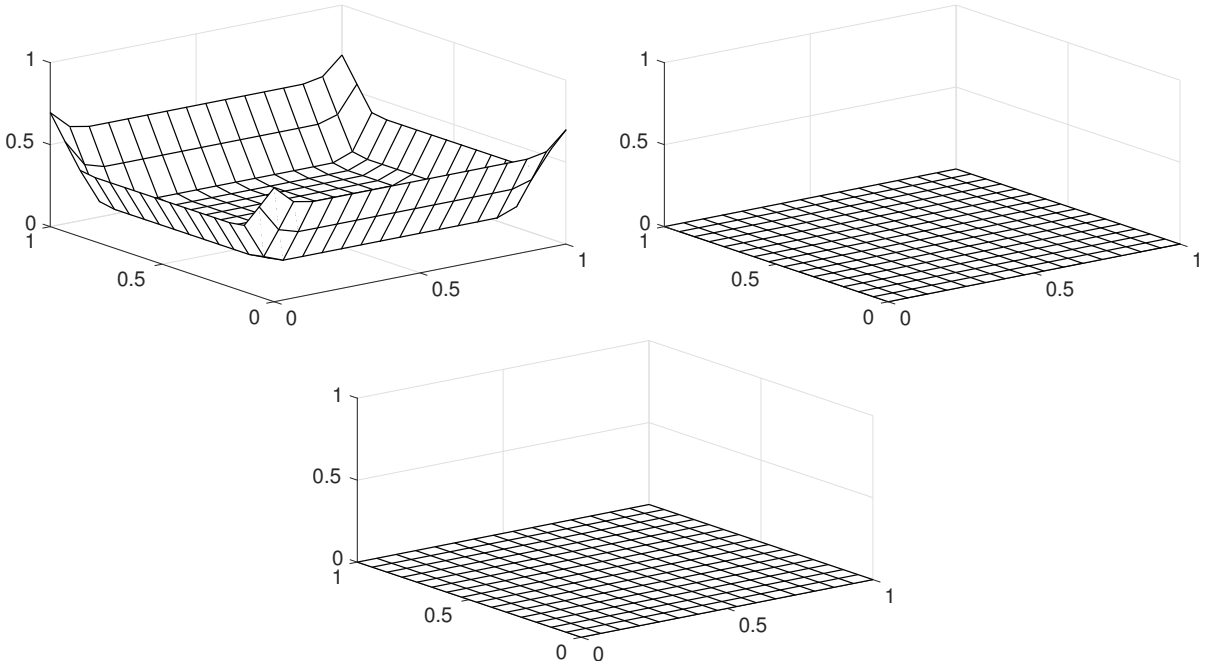

Figure 4.3: Approximation of the gradient of a linear function using SPH, the RKPM and the Randles Libersky correction, clockwise starting from top left.

For the next test a trigonometric function $f(x,y) = sin(x) \cdot sin(y)$ in the domain $[0,\pi] \times [0,\pi]$ is considered. This time a convergence study is performed. That is, some errors are

investigated with regard to increasing particle number / decreasing particle spacing. The errors consided are:

$$L_\infty = \sup\{|\partial/(\partial x)f(x,y) - \langle \partial/(\partial x)f(x,y)\rangle| \;\; x \in [0,\pi], y \in [0,\pi]\} \tag{4.58}$$

$$L_2 = \int_0^\pi \int_0^\pi \sqrt{1/N(\partial/(\partial x)f(x,y) - \langle \partial/(\partial x)f(x,y)\rangle)^2} dxdy \tag{4.59}$$

L_∞ can be understood as the maximum local error while L_2 is a measure of the average (quadratic) global error. The integral in L_2 is replaced by its Riemann sum. Two particle ensembles are considered, once regularly spaced, once sampled randomly from a uniform random distribution: $x, y \sim \mathcal{U}(0,\pi)$. The smoothing length h and integration weight ω_j is always chosen uniformly, even in the randomized case using $h = 1.7 \cdot \Delta x$ and $h = 1.7 \cdot \bar{\Delta} x$, respectively, where $\bar{\Delta}$ denotes the average spacing. ω_j is just $1/(\Delta x)$ in the regular and $1/(\bar{\Delta} x)$ in the randomized case.

Figure 4.4 and 4.6 show the reconstruction of said function using the methods at hand, using $16 \cdot 16 = 256$ randomly distributed particles. The SPH approximation shows heavy boundary deficiencies in the uniform case, which is further highlighted in figure 4.5, and barely resembles the structure of the expected solution in the random case, displaying heavy oscillations. Both the Randles Libersky and RKPM methods resolve the sine wave with very little error in the uniform case and display a smoother solution than SPH in the random case, while still exhibiting quite severe errors of up to 25 %. It has to be noted that the situation at hand is far from optimal and the error could be improved by chosing more appropriate, non-uniform integration weights ω_j and h. As for the smoothing length, see section 4.4. Optimal integrations can be computed from geometrical considerations only, e.g. by computing the Voronoi map [270] [188] and using the area of the cells for ω_j.

Subsequently the convergence with increasing N, or alternatively, decreasing $\Delta x/\bar{\Delta} x$ is analyzed. The convergence plots are best read from right to left. Since the plots are log-log a slope of 2 would mean quadratic convergence with particle spacing. Figure 4.7 shows the convergence for the regular particle spacing. In this case both the Randles Libersky correction as well as the RKPM show an almost perfect quadratic convergence. The SPH diverges because the error is dominated by the boundary deficiency. Figure 4.8 shows sublinear convergence for both RKPM and Randles Libersky, while SPH diverges again, but more rapidly. The slope for the converging methods is roughly 1/2. This makes sense since evaluating an integral of the form (4.3) using randomized, uniformly weighted sampling is essentially a Monte Carlo integration, and Monte Carlo integration is in $\mathcal{O}(\sqrt{N})$.

The results in this section can be summarized as:

- The boundary deficiency in SPH can be quite severe, severe enough to completly dominate the errors derived in the continuous, infinite domain
- The completely randomized, uniformly weighted setting is a challenging one. The error is apparently dominated by the simple integration scheme, which is essentially a Monte Carlo integration.

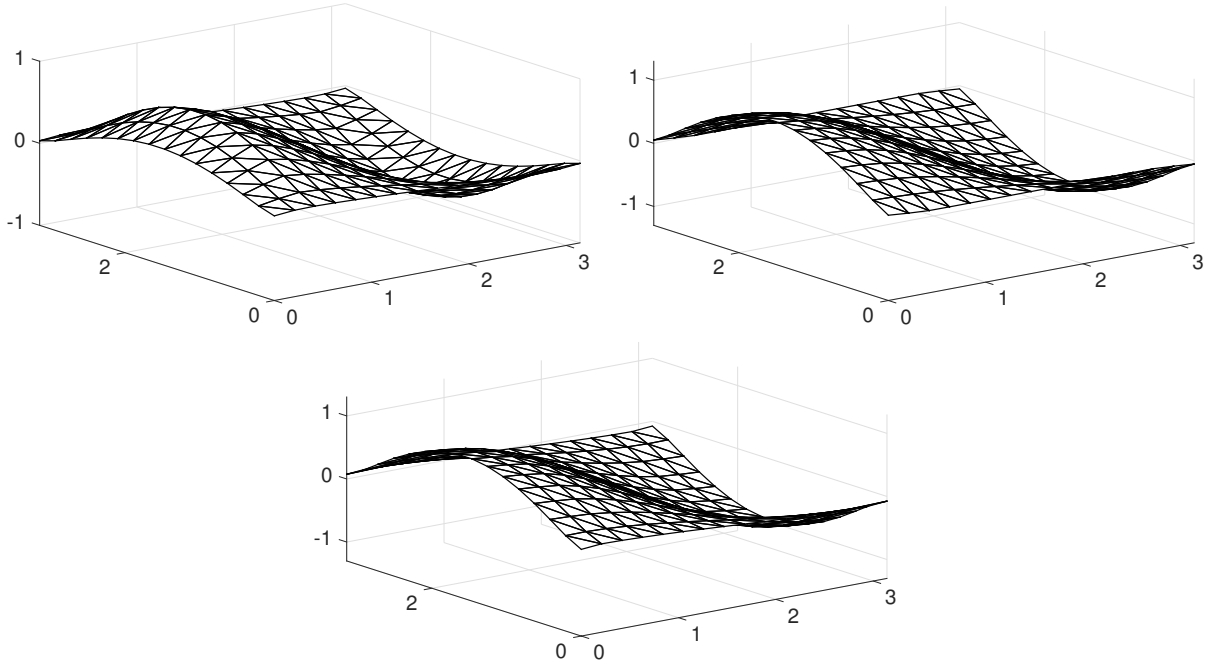

Figure 4.4: Approximation of the gradient of a trigonometric function using SPH, the Randles Libersky correction and RKPM, clockwise from top left. The particle ensemble is uniform. SPH exhibits heavy boundary deficiencies while the other methods are unaffected.

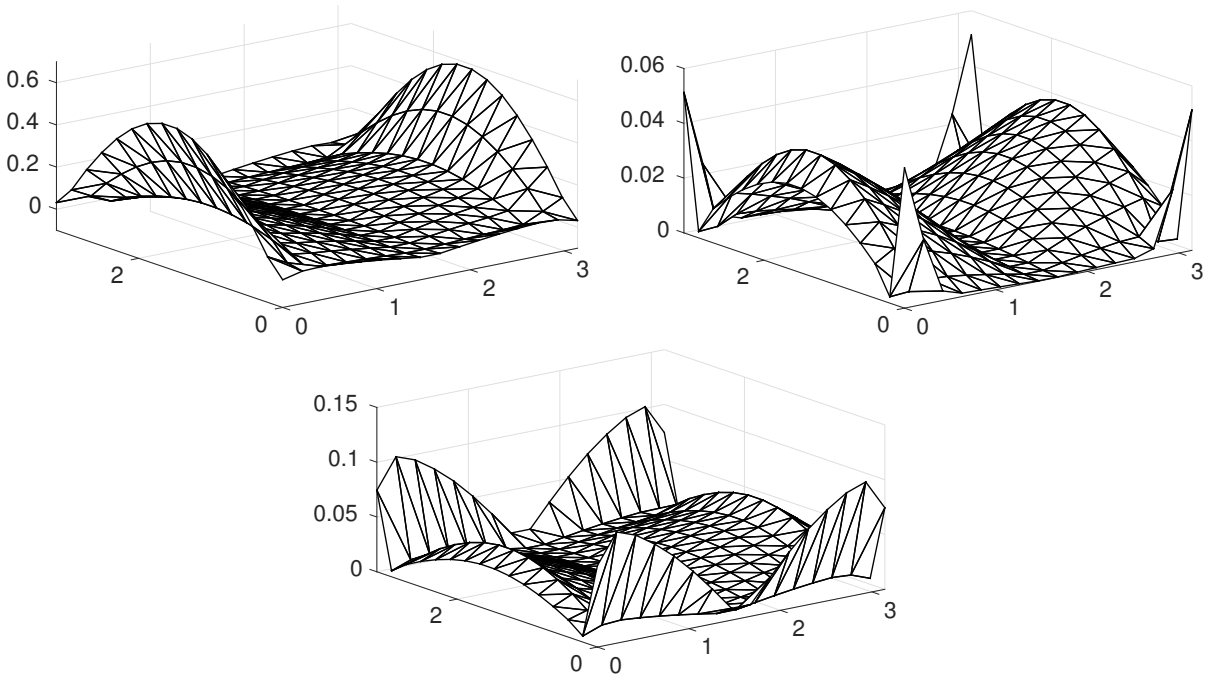

Figure 4.5: Approximation of the gradient of a trigonometric function using SPH, the Randles Libersky correction and RKPM, clockwise from top left. The residuals, i.e. $|\partial/(\partial x)f(x,y) - \langle \partial/(\partial x)f(x,y)\rangle|$ $x \in [0,\pi], y \in [0,\pi]$ are shown. Note the different scaling of the axes.

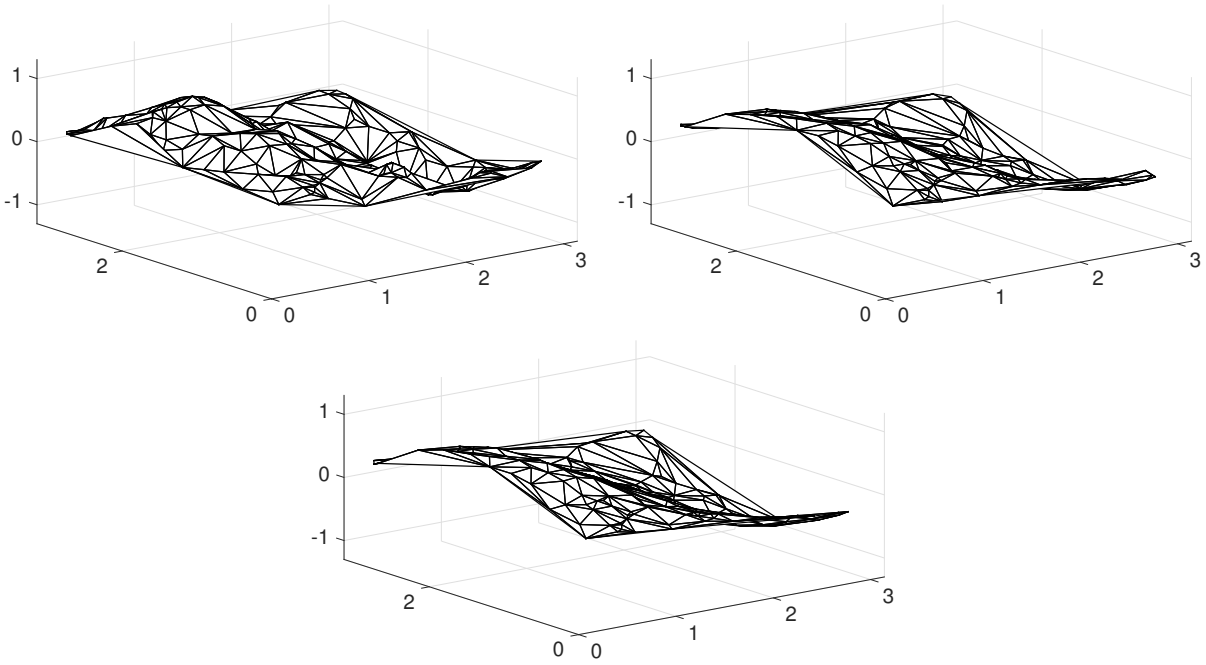

Figure 4.6: Approximation of the gradient of a trigonometric function using SPH, the Randles Libersky correction and RKPM, clockwise from top left. The particle ensemble is randomized. At this low resolution the structure of the sine wave is barely visible using SPH, but at least qualitatively retrieved by the other methods.

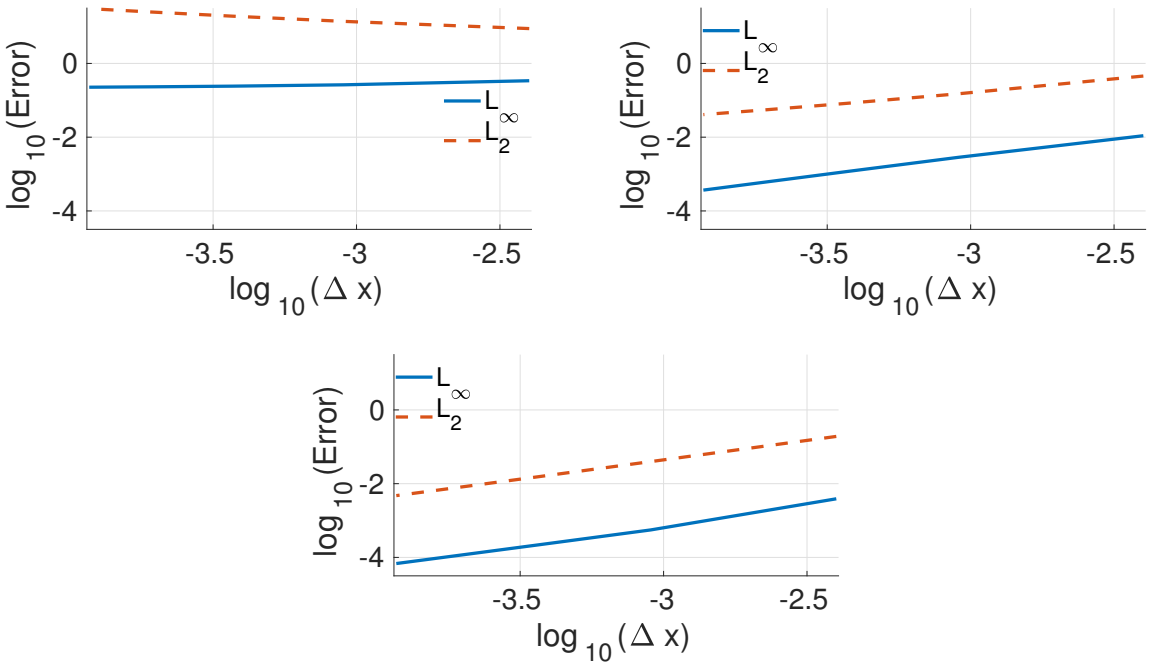

Figure 4.7: Convergence for the regular particle ensemble: SPH, Randles Libersky, RKPM, clockwise starting from top left.

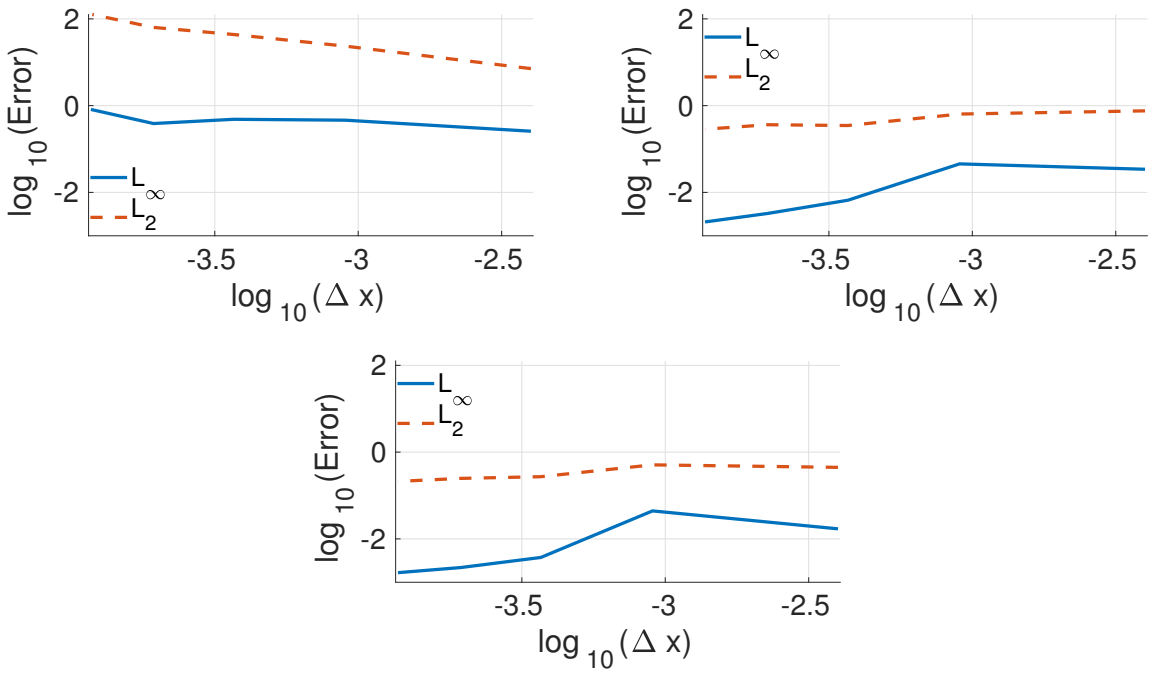

Figure 4.8: Convergence for the randomized particle ensemble: SPH, Randles Libersky, RKPM, clockwise starting from top left.

4.4 CHOICE OF SMOOTHING LENGTH

So far the discussion was restricted to stationary particles. Since the simulation methods applied to particle methods are usually Lagrangian in nature the particle movement can not be ignored. Equation (4.30) shows that particles need to overlap at all times. Furthermore, the equation systems (4.43) and (4.47), i.e. the corrected schems, need to have a unique solution. This necessitates that any given particle has a certain number of particles contained within its neighborhood which are not positioned in collinear fashion, see Dilts [61]. If collinear ensembles can't be avoided, Muller [178] suggests using safe matrix inversion by means of the Singular Value Decomposition, see [204].

Since the particle movement can't be predicted before the simulation is started this leaves two options

1. The particle arrangement is periodically re-regularized. This idea from vortex methods [51] can be applied to SPH, resulting in rSPH methods, see [38] for flows and [108] or [107] for solids. These methods, while interesting, do not retain certain characteristics of truly meshfree methods, like the arbitrary evolution of the simulation domain, and will thus not be further pursued in this thesis.

2. The smoothing length is adapted throughout the course of the simulation, essentially shrinking h in regions where particles amass and expanding h in regions where particles disperse. The first mechanism ensures that the runtime of the algorithm stays bounded while the second mechanism ensures that the error of the algorithm stays bounded.

This section will give a non exhaustive overview of some of these methods and will conclude with a numerical experiment comparing the methods at hand.

Already the very first SPH simulations [154] had a mechanism to adapt h, using:

$$h(\underline{x}) \propto \bar{\varrho}(\underline{x})^{-1/\eta} \tag{4.60}$$

$$\bar{\varrho}(\underline{x}) = \sum_{j=1}^{N} W_h(\underline{x} - \underline{x}_j) \tag{4.61}$$

That is, h is inversely proportional to the local particle density. Monaghan [168] and Springel, Helmquist [248] noticed simultaneously that equations (4.60) as well as (4.61) are not independent, since (4.61) is a function of the smoothing length as well. To simultaneously solve the two equations a standard root solving procedure can be applied. It might sound prohibitively expensive to apply an iterative root finding method per particle per timestep, however, numerical investigations have shown that the root finding method converges in very few iterations, usually well below 10. It is shown in [205] that this is the only method for consistent choice of h with regard to a number of conservation properties.

Slight variations of the idea in [154] is to evolve $\bar{\varrho}$ with an ordinary differential equation:

$$\dot{\bar{\varrho}} = \bar{\varrho} \nabla \underline{v} \tag{4.62}$$

where $\underline{v}$ is the velocity of the particles. This is the continuity equation applied to $\bar{\varrho}$. Yet another idea is for example presented in [192], where h is set to

$$h = \sqrt{\frac{\bar{\varrho}^n}{\bar{\varrho}^{n+1}}} \tag{4.63}$$

where n is the time step number. This concludes the overview of some choices of h, assuming a spherical neighborhood. There are other, more involved schemes, which track the local anistropy of a particles' neighbors and choose neighbors accordingly. An interesting idea to this end is presented in [213, 215]. Their algorithm is quite involved, so only the rough outline is presented here. The core of the algorithm depends on a remapping of the neighbors j of some particle i into a prime space where the close neighbors are mapped far away and vice versa. A simple map would be

$$\underline{x}'_j = \frac{\underline{x}_{ij}}{|\underline{x}_{ij}|^2} \tag{4.64}$$

which maps $\underline{x}_i$ to infinity and the closest shell of neighbors to the exterior hull in the prime space. This exteriour hull can then be recovered using a fast convex hull algorithm, e.g. [16]. Of course, nothing is gained or lost so far since map (4.64) is an isotropic one. To address this the local covariance matrix $\underline{\underline{\mathcal{C}}}$ is formed:

$$\underline{\underline{\mathcal{C}}}_i = \frac{1}{N-1} \sum_{j \in nbh(i)} \underline{x}_{ij} \otimes \underline{x}_{ij} \tag{4.65}$$

i.e. the index j is only over the particles close to i. This is now subjected to an Eigenvalue decomposition:

$$\underline{\underline{\mathcal{C}}}_i = \underline{\underline{Q}} \cdot \underline{\underline{\Lambda}} \cdot \underline{\underline{Q}}^{-1} \tag{4.66}$$

and the mapping (4.64) is modified to:

$$\underline{x}'_j = 1/\sqrt{\underline{\underline{\Lambda}}} \cdot \underline{\underline{Q}} \cdot \underline{x}_{ij} \tag{4.67}$$

Such that the local anisotropy and rotation is considered. Implementation details, including weighting of stretching and rotation can be found in the original works [213, 215].

The final idea presented here is to fit an ellipse (ellipsoid in 3D) to the local neighborhood directly. The following can be seen as a stark simplifaction of the method given in [194, 234]. Again, the local neighborhood j of particles i is considered and the local covariance matrix $\underline{\underline{\mathcal{C}}}$ is computed. A famous result from statistics is that an optimal ellipse can be fit to the data points $\underline{x}$ by way of an Eigenvalue decomposition of $\underline{\underline{\mathcal{C}}}$, under the assumption that $\underline{x}_j$ are bivariate (trivariate in 3D) normally distributed about $\underline{x}_i$. It turns out that the length of the half axis of the fitted ellipse are the eigenvalues λ_k of $\underline{\underline{\mathcal{C}}}$, and eigenvectors $\underline{q} \in \underline{\underline{Q}}$ the directions. Additionally, since $\sqrt{\lambda_k} = \sigma_k$ with σ_k being the standard deviation of the underlying normal distribution along dimension k, the ellipse can be scaled using a suitable χ^2 value, e.g. $\sqrt{5.991}$ for 95% confidence.

To conclude this section all methods mentioned will be subjected to three situations typically encountered in classic material tests in solid mechanics: compression, shear and tension. No physical simulation has been performed, the particles are passively advected by a velocity field emulating the material test at hand. The methods considered are

- Compute (4.61) and apply (4.60). Called “Density - Sum”.
- Solve (4.60) (4.61) simultaneously with a root finding method. Called “Iterative”.
- Compute (4.60) but evolve $\bar{\varrho}$ by (4.62). Called “Density - Continuity”.
- Use equation (4.63) while evolving $\bar{\varrho}$ by (4.62). Called “Fraction”.
- Use the prime space method by [213, 215]. Called “Prime Space”.
- Use the ellipse approach discussed in this thesis. Called “Ellipse”.

The results can be seen in figures 4.9, 4.10 and 4.11. Where the unit square is compressed, stretched and sheared, respectively. Since the particle arrangement is very small, the full particle set is considered as candidate set. The figures permit a number of observations: The Density Sum and Iterative methods yield very similar results. This is not surprising, since using Density Sum is essentially performing one iteration of the iterative method, and the iterative method converges very quickly. Similar results are also obtained by the Density - Continuity and Fraction methods. This is the case since both methods obtain $\bar{\varrho}$ in the same fashion. Both the prime space and ellipse methods follow the local anisotropy of the

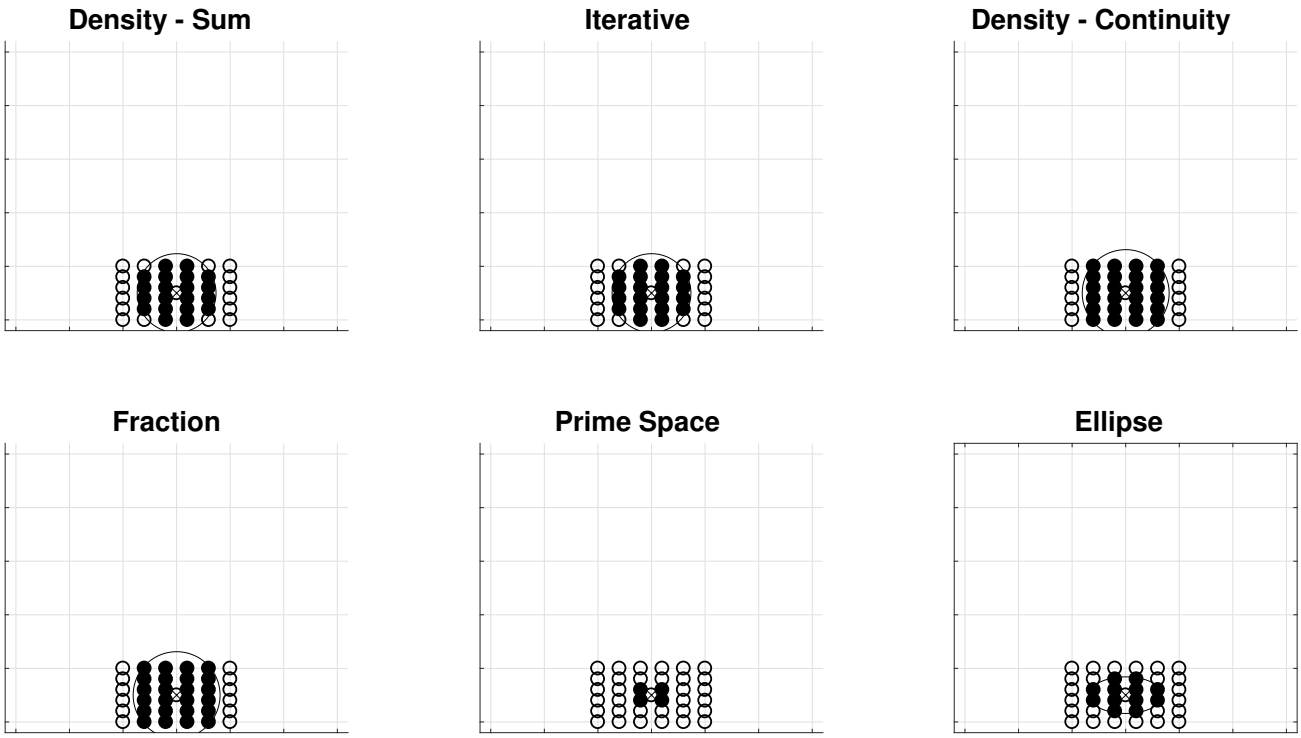

Figure 4.9: Compression test case for various methods for chosing the smoothing length h. Center point crossed and circled, candidates circled, chosen candidates filled.

neighborhood quite well. The ellipse method identifies some more neighbors. In the shear case the prime space methods misses some neighbors which seem to introduce gaps in the local connectivity.

Finally, it's hard to define what a "good" neighborhood entails. After all, the total approximation error (4.30) poses an upper as well as a lower limit on h. A final verdict on the suitability of a choice of h can only be made with respect to its implementation into a specific algorithm.

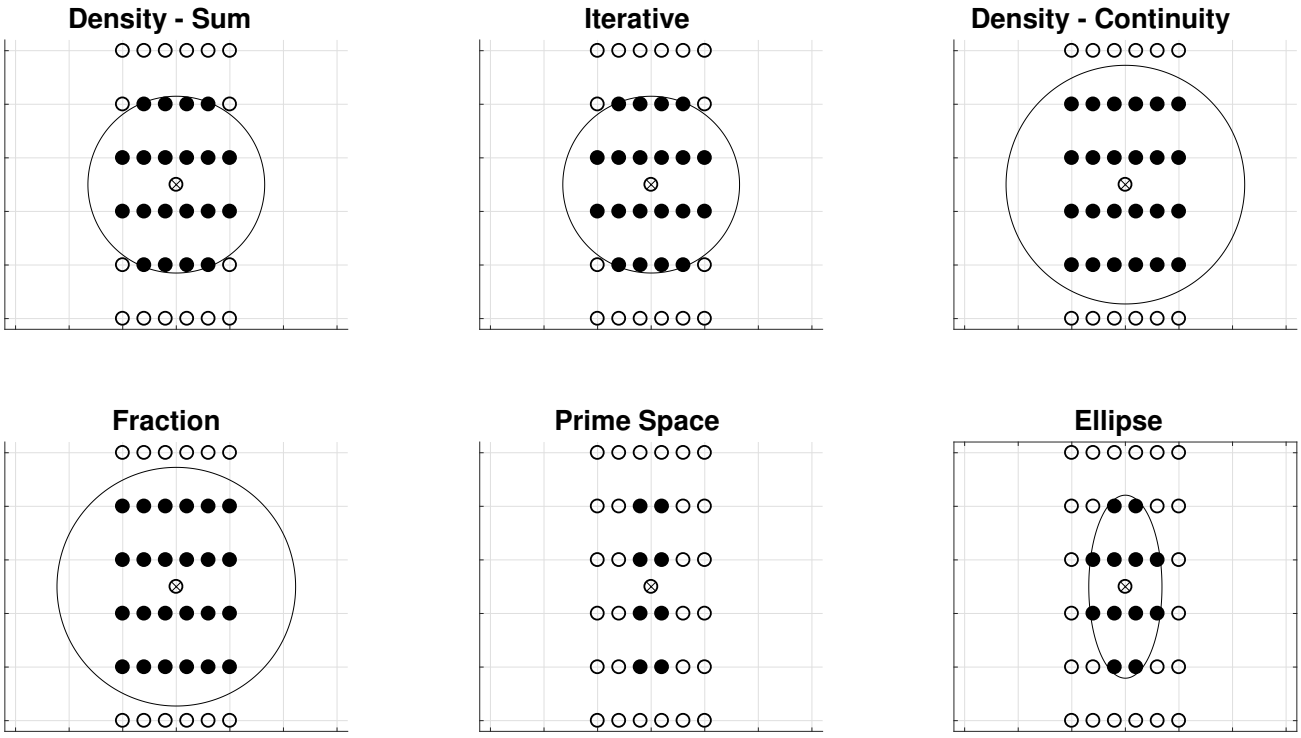

Figure 4.10: Tension test case for various methods for chosing the smoothing length h. Center point crossed and circled, candidates circled, chosen candidates filled.

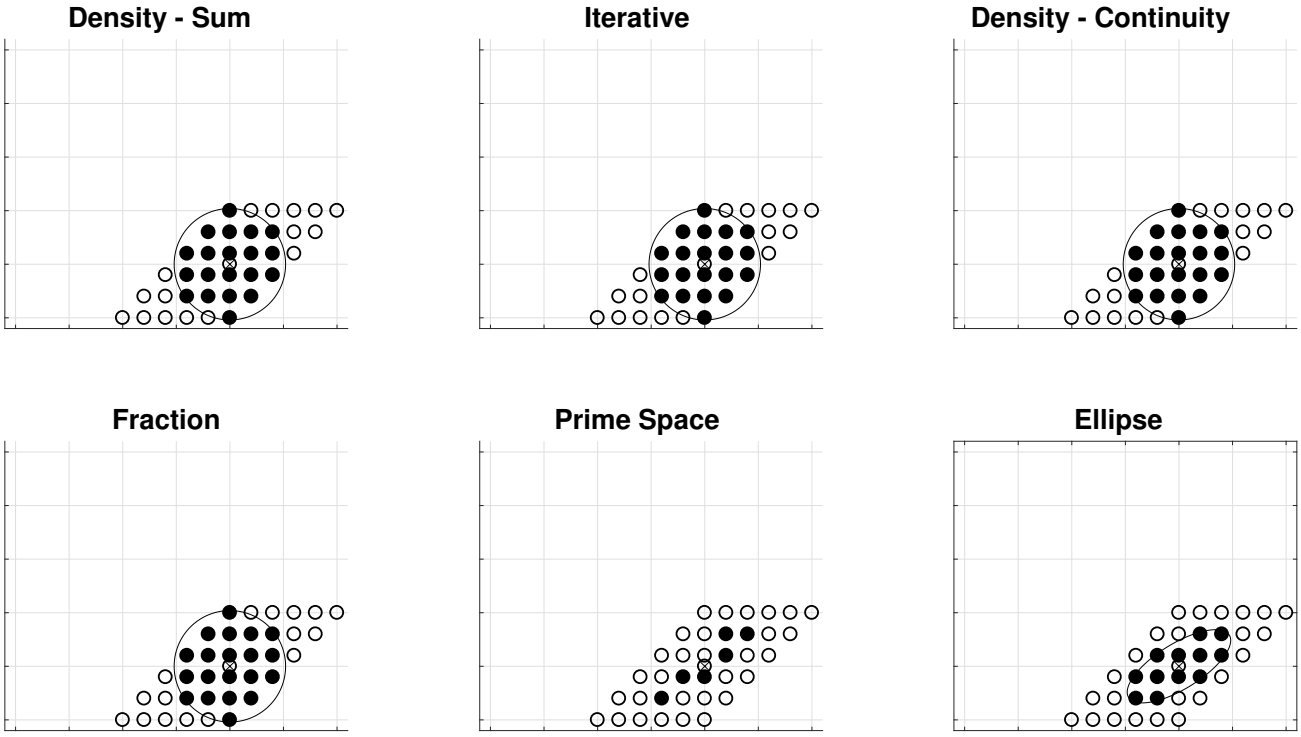

Figure 4.11: Shear test case for various methods for chosing the smoothing length h. Center point crossed and circled, candidates circled, chosen candidates filled.

4.5 HIGHER ORDER DERIVATIVES

So far the discussion has been limited to partial derivatives of first order, i.e. the gradient. Higher order derivatives, especially second order derivatives in the form of the Laplacian arise in a number of important PDEs however. For example in heat / diffusion equations or the viscosity term in the Navier Stokes equation. In weak form algorithms those higher order derivatives are often treated by partial integration. However, since most meshless methods operate on the strong form, higher order derivatives can not be neglected.

The discussion in this chapter shall be limited to second order derivatives. Second order derivatives for meshless methods are no straight forward task and have been subject to constant research since the inception of the SPH. A first trial at approximating the Laplacian of some function $f(.)$ might be to simply derive the kernel function twice:

$$\nabla(\nabla f(\underline{x})) \approx \int_\Omega f(\underline{x}')\nabla(\nabla W_h(\underline{x}-\underline{x}'))d\underline{x}' \tag{4.68}$$

This is of course a highly questionable approach that is solely motivated by intuition if no formal justification similar to (4.9) with regard to (4.10) is presented. However, if the same idea is applied to (4.34) instead of (4.10) to arrive at:

$$\nabla(\nabla f(\underline{x})) \approx \int_\Omega (f(\underline{x})-f(\underline{x}'))\nabla(\nabla W_h(\underline{x}-\underline{x}'))d\underline{x}' \tag{4.69}$$

the scheme can be shown to be stable for continuous initial conditions [77]. The scheme, called "Fisheloves" scheme was shown to be unstable using initial conditions with a jump in [74], however. In the same work it is recommended to use a difference type scheme like the one by Brookshaw [30]:

$$\nabla(\nabla f(\underline{x})) \approx \sum_{j=1}^{N} 2\left(\left(\frac{f(\underline{x})-f(\underline{x}_j)}{|\underline{x}-\underline{x}_j|}\right)\underline{e}_{ij}\cdot\nabla W_h(\underline{x}-\underline{x}_j)\right)\omega_j \tag{4.70}$$

which is essentially a finite difference scheme applied to (4.32). This method, along with the similar scheme given in [47] have the favorable property that they do not contain the second order derivative of the kernel function. This is especially important when using kernels of lower differentiability like the cubic spline, see figure 4.12.

A host of other schemes exist. Both the RKPM and CSPM methodology offer straight forward, albeit algebraically tedious, extensions to any degree of derivative. These options were explored in [43] and [41], respectively. Recently, completely new schemes have been proposed by Fatehi in [75] and by Korzilius in [132], underlining that the issue of even second order derivatives is still a heavily investigated one.

A completely different approach to higher order derivatives is the Particle Strength Exchange (PSE) method. It was introduced by [46] and [52], but formalized in the seminal work by Eldredge [71]. All schemes mentioned so far are based on the idea of function approximation, than applying the differential operator to that approximation in some way, resulting in one or two-fold differentiation of the mollification kernel W_h. The idea behind PSE is to design a

kernel that inherently performs the desired degree of differentiation. That is, W will be a completely different function depending on the degree of differentiation to be performed.

The PSE can be motivated by a Taylor series expansion. The discussion is kept one dimensional for simplicity and ease of notation:

$$f(x') = f(x) + (x' - x)\frac{\partial f(x')}{\partial x}|_{(x)} + \frac{1}{2}(x' - x)^2\frac{\partial^2 f(x')}{\partial^2 x}|_{(x)} + \frac{1}{6}(x' - x)^3\frac{\partial^3 f(x')}{\partial^3 x}|_{(x)} + \dots \tag{4.71}$$

After subtraction of x, multiplication with the PSE kernel to be determined W_h^{PSE} and integration over the whole domain:

$$\begin{aligned}\int_{-\infty}^{+\infty} (f(x') - f(x))W_h^{\text{PSE}}(x' - x)dx' \\ &= \int_{-\infty}^{+\infty} (x' - x)\frac{\partial f(x')}{\partial x}|_{(x)} W_h^{\text{PSE}}(x' - x)dx' \\ &+ \int_{-\infty}^{+\infty} \frac{1}{2}(x' - x)^2\frac{\partial^2 f(x')}{\partial^2 x}|_{(x)} W_h^{\text{PSE}}(x' - x)dx' \\ &+ \int_{-\infty}^{+\infty} \frac{1}{6}(x' - x)^3\frac{\partial^3 f(x')}{\partial^3 x}|_{(x)} W_h^{\text{PSE}}(x' - x)dx'\end{aligned} \tag{4.72}$$

W_h^{PSE} is now chosen such that:

$$\frac{1}{2}\frac{\partial^2 f(x')}{\partial^2 x}|_{(x)} \underbrace{\int_{-\infty}^{+\infty} (x' - x)^2 W_h^{\text{PSE}}(x' - x)dx'}_{\stackrel{!}{=}2} \tag{4.73}$$

If W_h^{PSE} is chosen to be symmetric as well all integrals over terms containing $(x' - x)^p$ with odd p will vanish, yielding a method in $\mathcal{O}(x - x')^4$. The canonical choice (in 1D) is:

$$W_h^{\text{PSE}}(x' - x) = \frac{1}{2h\sqrt{\pi}} e^{\frac{|x'-x|^2}{4h^2}} \tag{4.74}$$

Only schemes of the form (4.70) and (4.74) will be considered in this thesis.

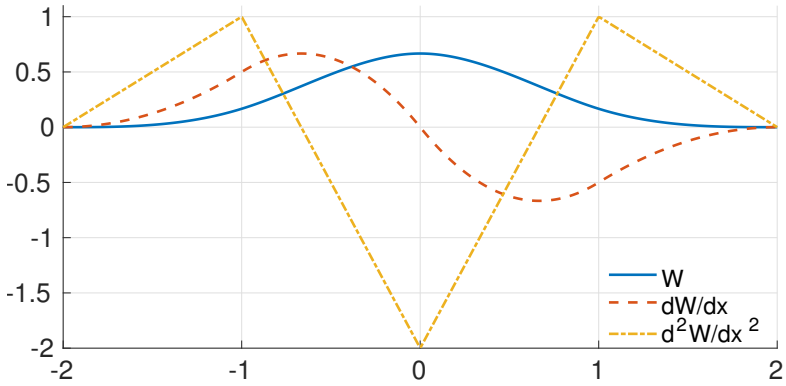

Figure 4.12: The cubic spline function (solid line) and its first two derivatives (dashed and dash-dotted line, respectively). It can be seen that the second derivative is not smooth anymore. That is, the third derivative of the cubic spline is not continuous.

4.6 INSTABILITY MODES

So far the meshless methods discussed were presented in a function approximation framework only. However, having rapid convergence with increasing number of particles when approximating the spatial derivative of a function is not enough to solve partial differential equations. Other issues to consider include, amongst many others, stability and conservation.

While stability considerations are subject to the specific algorithm at hand, and what PDEs are to be solved, there are still two frequently cited instability modes that seem to permeate the SPH/meshless literature, which will be discussed in the following. The next section 4.7 deals with what can be done to prevent such modes.

4.6.1 *The Tensile Instability*

The tensile instability was first formally described by Swegle in [255]. Swegle performed a von Neumann analysis [187] in one spatial dimension x and found that the SPH grows unstable in regions where:

$$\frac{\partial^2 W_h(x)}{\partial^2 x} \cdot \sigma > 0 \tag{4.75}$$

where σ is the stress. It was also shown that this result holds independent of any artificial viscosity terms, see [255], as well as the choice of kernel function. Expression (4.75) reveals that the term "tensile instability" is a misnomer, since the condition might just as well hold true for negative σ (compression) but positive $\partial^2 W_h(x)/\partial^2 x$. In fact, condition (4.75) might even be true if only hydrostatic pressure is present, i.e. in fluids. See for example [172] for considerations. Belytschko and coworkers perform a similar stability analysis in [23] but focus their efforts on differences between updated and total Lagrangian formalisms, determining that the instability mode is not present in total Lagrangian formalisms.

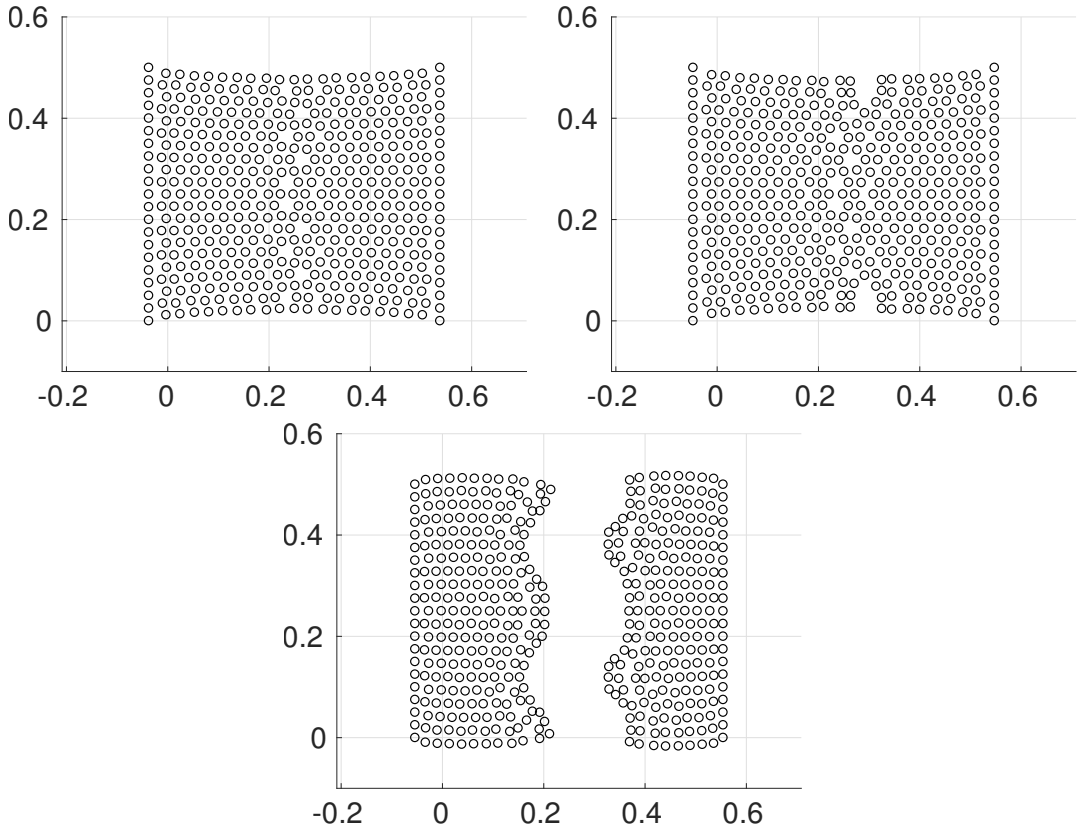

Figure 4.13: Three time steps of an elastic tensile test as simulated by SPH are shown. Progress left to right, top to bottom. Particles cluster in columns, gaps open and the material finally ruptures due to numerical fracture

It has to be noted that performing a von Neumann stability analysis is an involved process. The whole analytical expression for all state variables from an old time step to the next has to be subjected to a Fourier analysis, making the von Neumann analysis at hand valid only for the specific time stepper, constitutive equations including material model, type of particle ensemble etc. It is thus hard to tell if a method shown stable in one dimension assuming linear elasticity will be stable in two spatial dimensions using hypoelasticity, say. This is also the reason why more general analyses of this kind, involving irregular particle spacings, material models, correction terms and the like seem absent from literature.

An interesting note is given in [74] with regard to heat conduction. It is derived that schemes of the form (4.69) are only stable if the second derivative of the kernel function is positive at the nearest neighbors, i.e. the inflection point of the kernel function takes place before the first neighbor(s). Interestingly enough this is again an instability mode controlled by the second derivative of the kernel (4.75), but with regard to the diffusion equation instead of the momentum equation.

4.6.2 *Zero Energy Modes*

A zero energy mode is a mode in any field variable that spuriously does not induce energy into the equation system due to the discretization scheme chosen, even though in the physical

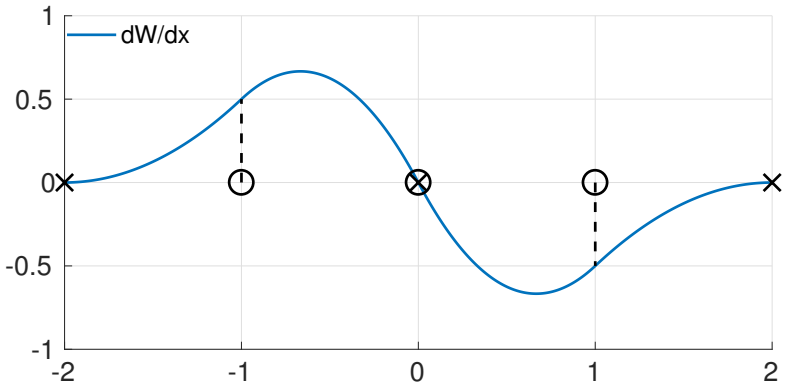

Figure 4.14: Illustration regarding a zero energy mode. Center particle i is crossed and circled. The closest neighbors are circled and contribute the same amount to the gradient of a field variable, but with opposite sign. The neighbors of degree two (crosses) do not contribute since h was chosen to be Δx.

sense, it would. These modes, while broader in scope, are very similar to hour glassing modes in the FEM. That is, the problem of zero energy modes is not inherent to meshless methods but afflicts most any numerical method. A simple example is constructed to illustrate the problem at hand. Consider a one dimensional, regular particle ensemble. The smoothing length equals the particle spacing, $h = \Delta x$. The cubic spline (4.31) is used as the kernel function, hence only the closest two particles contribute to any SPH approximation. Consider particle i with neighbors $i-1$ and $i+1$ with displacements $u(x_i) = 0$, $u(x_{i+1}) = u_{i+1}$ and $u(x_{i-1}) = u_{i-1}$. To approximate the gradient (4.32) is applied:

$$\begin{aligned}\left\langle \frac{\partial u}{\partial x} \right\rangle |_{x_i} &= \sum_{j=1}^{N} u_j \frac{\partial u}{\partial x} W_h(x_i - x_j)\omega_j \\ &= u_{i-1}\frac{\partial u}{\partial x} W_h(x_i - x_{i-1})\omega_j + u_{i+1}\frac{\partial u}{\partial x} W_h(x_i - x_{i+1})\omega_j \\ &= u_{i-1}\frac{\partial u}{\partial x} W_h(x_i - x_{i-1})\omega_j - u_{i+1}\frac{\partial u}{\partial x} W_h(x_i - x_{i-1})\omega_j \\ &= (u_{i-1} - u_{i+1})\frac{\partial u}{\partial x} W_h(x_i - x_{i-1})\omega_j \end{aligned} \tag{4.76}$$

where it was used that $\omega_{i-1} = \omega_{j+1}$ due to the uniform spacing and $W(x_i - x_{i-1}) = -W(x_i - x_{i+1})$, see figure 4.14. Now if $u_{i-1} = u_{i+1} = c$ with $c > 0$ there should clearly be some strain $\varepsilon = \partial u/\partial x$ measured since $u_i = 0$, but the above equates to zero. Hence, a zero energy mode is present.

A more formal discussion of zero energy modes can for example be found in [25], albeit with application to finite elements instead of meshless methods.

4.7 STABILIZATION MEASURES

In the previous section some instability modes where briefly discussed. In this section some counter measures are presented. This list is by no means exhaustive but covers all the

stabilization procedures employed in this thesis and some other broadly used or notable ones.

4.7.1 *Artificial Viscosity*

The artificial viscosity is one of the oldest and most broadly used stabilization measures. It was originally proposed in [169] by Gingold and Monaghan to simulate shock tubes. The idea is to suppress excessive oscillatory modes if shocks are introduced into the simulation. This is not only relevant in shock wave simulation but is more general, since direct application of Dirichlet boundary conditions or sudden particle movements due to (penalty) contact constitutes a shock, albeit an arguably weaker one than a shock wave, as well.

Artificial Viscosity works by enhancing the standard momentum equation by an artificial viscosity term:

$$\underline{\ddot{x}}_i = \sum_{j=1}^{N} m_j \left(\frac{\underline{\underline{\sigma}}_i}{\varrho_i} + \frac{\underline{\underline{\sigma}}_j}{\varrho_j} + \Pi_{ij} \underline{\underline{I}} \right) \nabla W_h^{ij} \tag{4.77}$$

where the shorthand $W_h^{ij} = W(\underline{x}_i - \underline{x}_j)$ was introduced. The momentum equation is discretized using SPH approximator (4.32). Π_{ij} is the artificial viscosity in question and is defined as:

$$\Pi_{ij} = \begin{cases} \frac{-\alpha^{av} \overline{c}_{ij} \mu_{ij} + \beta^{av} \mu_{ij}^2}{\overline{\varrho}_{ij}}, & \underline{v}_{ij} \cdot \underline{x}_{ij} < 0 \\ 0, & \underline{v}_{ij} \cdot \underline{x}_{ij} \geq 0 \end{cases} \tag{4.78}$$

where the condition on the right ensures that artificial viscosity is only to be introduced to particles that are moving towards each other. In the above expression

$$\mu_{ij} = \frac{h \underline{v}_{ij} \cdot \underline{x}_{ij}}{|\underline{x}_{ij}|^2 + \epsilon^{av}} \tag{4.79}$$

where α^{av}, β^{av} and η^{av} are parameters to be defined by the user. A popular choice, due to experience, would be $\alpha^{av} = \beta^{av} = 1$ and $\eta^{av} = 0.1$. A quantity with an over bar $\overline{x}_{ij}$ denotes the average of that quantity, i.e. $\overline{x}_{ij} = 0.5 \cdot (x_i + x_j)$. A vector with two subscripts denotes the difference vector, e.g. $\underline{v}_{ij} = \underline{v}_i - \underline{v}_j$. Even though the use of the corrector is widespread, it is for example present in the first solid mechanics paper using SPH [141] as well as in the commercial codes `Abaqus` and `LSDYNA`, it is not without problems:

- The introduction of three non-physical parameters that are tunable seems quite excessive
- Especially because they can be shown to alter the physics at hand quite severely. In [122] it was shown that the penetration depth in an SPH impact simulation can be altered three-fold by choice of artificial viscosity parameters

- The introduction of a viscosity term into a solid mechanical simulation seems conceptually questionable. There is no viscosity between to infinitesimal solid elements (as opposed to fluid elements).

The last point is addressed in a work by Vidal and collegues [267]. She proposed a purely numerical approach to introduce viscosity of the form:

$$\langle \nabla f_i \rangle = \sum_{j=1}^{N} f_j \overset{\circ}{W}{}_h^{ij} \omega_j + \underbrace{\eta \left[\underline{\underline{Y}}_i - \langle \nabla (\nabla f_i) \rangle \right] \underline{h}}_{\text{art. viscosity}} \tag{4.80}$$

here η controls the intensity of the stabilization used. $\langle \nabla (\nabla f_i) \rangle$ is the Hessian found by RKPM, i.e. by deriving equations (4.44) and (4.45) again to arrive at double and mixed derivatives. For the two dimensional case, $\underline{h} = [h, h]$ and $\underline{\underline{Y}}_i$ is a novel approximator for the Hessian:

$$\underline{\underline{Y}}_i = \sum_{j=1}^{N} f_j \left[\underline{\underline{Y}}_W^{ij} + \delta_{ij} \underline{\underline{B}}_i + \underline{\underline{\underline{A}}}_i \cdot (\underline{x}_i - \underline{x}_j) \right] \omega_j \tag{4.81}$$

$\underline{\underline{Y}}_W^{ij}$ is simply the Hessian of the kernel function directly:

$$\underline{\underline{Y}}_W^{ij} = \nabla (\nabla W_h^{ij}) \tag{4.82}$$

the correction terms $\underline{\underline{B}}_i$ and $\underline{\underline{\underline{A}}}_i$ are found by demanding that constant and linear functions have null Hessian:

$$\begin{aligned} \sum_{j=1}^{N} \left[\underline{\underline{Y}}_W^{ij} + \delta_{ij} \underline{\underline{B}}_i + \underline{\underline{\underline{A}}}_i \cdot (\underline{x}_i - \underline{x}_j) \right] \omega_j &= \underline{\underline{0}} \\ \sum_{j=1}^{N} \underline{x}_j \left[\underline{\underline{Y}}_W^{ij} + \delta_{ij} \underline{\underline{B}}_i + \underline{\underline{\underline{A}}}_i \cdot (\underline{x}_i - \underline{x}_j) \right] \omega_j &= \underline{\underline{0}} \end{aligned} \tag{4.83}$$

This leads to equation systems:

$$\begin{aligned} \underline{\underline{\underline{A}}}_i &= \left[\sum_{k=1}^{N} \underline{\underline{Y}}_W^{ik} \otimes (\underline{x}_i - \underline{x}_k) \omega_k \right] \cdot \left[\sum_{k=1}^{N} (\underline{x}_k - \underline{x}_i) \otimes (\underline{x}_k - \underline{x}_i) \omega_k \right]^{-1} \\ \underline{\underline{B}}_i &= \left[\sum_{k=1}^{N} (\underline{\underline{Y}}_W^{ik} + \underline{\underline{\underline{A}}}_i \cdot (\underline{x}_k - \underline{x}_i)) \omega_k \right] \frac{-1}{\omega_i} \end{aligned} \tag{4.84}$$

This is the same approach taken to construct the Randles Libersky correction, see (4.48), but applied to the Hessian. This artificial viscosity term has some coneceptual advantages to the one proposed by Monaghan et al.: It only introduces one additional non-physical parameter and it is motivated by numerical considerations only. It also reuces to zero as the particle number grows to infinity since both the newly proposed approximator for the Hessian as well as the Hessian found by RKPM are of the same, i.e. first order.

Unfortunately the numerical overhead for the corrector is, contrary to what is stated in the original publication [267], quite intense: Finding the Hessian by RKPM necessitates computation and inversion of two 3x3 matrices, the scheme by Vidal an additional computation of

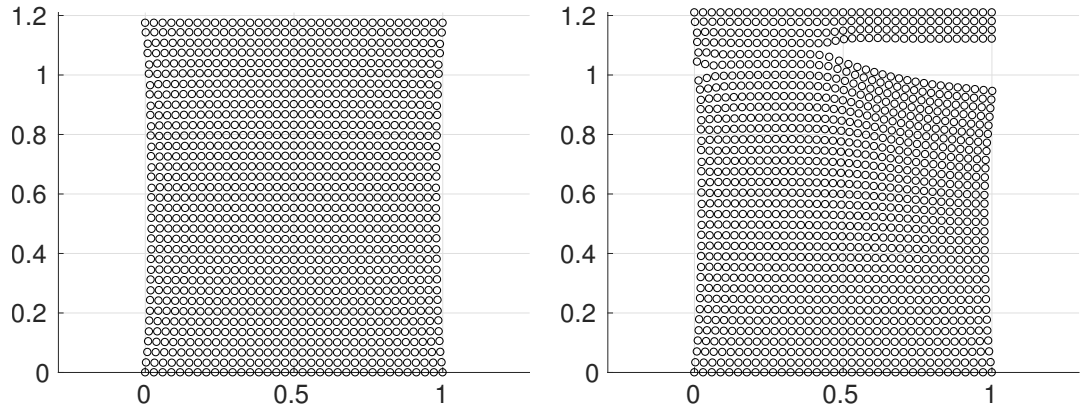

Figure 4.15: A tensile test using the artificial viscosity term by Vidal et al., implemented into the simulation framework given in [91]. Tensile instability causes unphysical fracture at strains larger than 0.2

a 2x2 matrix, computation of a 2x2x2 tensor of third order and computation and inversion of yet another 2x2 matrix. Also, while shown to be stable in impact problems, preliminary test simulations showed that the stabilization does not prevent numerical fracture due to tensile instability modes in a tensile test, consider figure 4.15. It was thus not considered further in this work.

4.7.2 *Artificial Stress Terms*

While the artificial viscosity was designed to dampen excessive shocks there exists another corrector that was specifically designed to counteract the tensile instability in the form of artificial stress terms. The technique was introduced by Gray and Monaghan in [91]. Again, the standard momentum equation is enhanced by an additional term:

$$\underline{\ddot{x}}_i = \sum_{j=1}^{N} m_j \left(\frac{\underline{\underline{\sigma}}_i}{\varrho_i} + \frac{\underline{\underline{\sigma}}_j}{\varrho_j} + \Pi_{ij} \underline{\underline{I}} + (\underline{\underline{\Theta}}_i + \underline{\underline{\Theta}}_j) f^n \right) \nabla W_h^{ij} \tag{4.85}$$

here, f is a term regulating the influence of the artificial stress terms. It confines the effect of the artificial stress to the nearest neighbors. n is again a tunable parameter that regulates how quickly the effect of the artificial stress vanishes. To determine $\underline{\underline{\Theta}}$ the Cauchy stress is diagonalized to find the principal stress(es) $\overline{\underline{\underline{\sigma}}} = \underline{\underline{R}} \cdot \underline{\underline{\sigma}} \cdot \underline{\underline{R}}^T$. Then for each diagonal component:

$$\overline{\Theta}_{xx} = \begin{cases} -\epsilon^{\text{Stress}} \frac{\overline{\sigma}_{xx}}{\varrho^2} & \overline{\sigma}_{xx} > 0 \\ 0 & \overline{\sigma}_{xx} \leq 0 \end{cases} \tag{4.86}$$

where ϵ^{Stress} regulates the intensity of the artificial stress. To find $\underline{\underline{\Theta}}$, $\overline{\underline{\underline{\Theta}}}$ is rotated back into the original frame $\underline{\underline{\Theta}} = \underline{\underline{R}}^T \cdot \overline{\underline{\underline{\Theta}}} \cdot \underline{\underline{R}}$. In summary, an analysis of the principal stress state is done. If the stress state is tensile along any direction an artificial compressive stress is added to the stress state in that specific direction.

This procedure might raise the concern if or to what degree the introduction of the additional stress terms influences the macro scale solution of a solid mechanics problem. The authors show in [91] that the effect is negligible on the macro scale by means of a dispersion analysis. Another concern lies in the fact that the artificial stress only steps into effect in the tensile state while it was shown in the previous section 4.6.1 that the tensile instability can also happen in compressive stresses. Indeed, the authors show in [91] that while instability modes are eliminated in a lot of situation they are still present in a high strain tensile test.

4.7.3 *Smoothing Schemes*

Yet another correction measure introduced by Monaghan [166] is the XSPH. The idea of XSPH is quite simple: Move the particles with a weighted average velocity instead of the individual particle velocity. More precisely

$$\dot{\underline{x}}_i = \underline{v}_i + \epsilon^{\text{XSPH}} \sum_{j=1}^{N} \frac{m_j}{\varrho_i + \varrho_j} (\underline{v}_j - \underline{v}_i) W_{ij} \tag{4.87}$$

i.e. the SPH approximator is used as weighted averaging. $0 < \epsilon^{\text{XSPH}} < 1$ again controls the intensity of the corrector, or, better phrased controls the ratio of smoothed particle velocity and "original" particle velocity.

Mohammadi and co-workers [193] [192] suggested a smiliar scheme. They smooth the velocity field directly using CSPM approximation with $h' \stackrel{!}{=} h$:

$$\underline{v}_i = \frac{\sum_{j=1}^{N} \underline{v}_j W_{h'}^{ij} \omega_j}{\sum_{j=1}^{N} W_{h'}^{ij} \omega_j} \tag{4.88}$$

In their work it was shown that this smoothing scheme can stabilize some test problems in solid mechanics. However, the smoothing introduced by this scheme is quite strong, especially for $h' \sim h$. So it is possible that excessive numerical dissipation might be introduced, see appendix 9.4.

One last point to consider regarding smoothing schemes is that they are active throughout the solution at all points in time, whereas the artificial viscosity and stress schemes are only active under certain conditions, i.e. tensile stresses for the artificial stress scheme and particles about to penetrate each other for the artificial viscosity scheme.

4.7.4 *Stress Points*

The idea of stress points is quite simple: Instead of a single particle set two particle sets are considered. The two particle types are initialized on a staggered grid, and the field variables are split between the two sets. As the name suggests, it is common to have one set to carry the stress, i.e. the stress points, and the other set to carry the rest of the field variables, i.e.

the particles. If a variable has to be known at a set not carrying it it is simply interpolated by SPH. For example: Let $j = 1 \dots N$ be the set of particles and $k = 1 \dots M$ be the set of stress points. A stress point at $\underline{x}$ can then be moved by:

$$\underline{v}(\underline{x})_k = \sum_{j=1}^{N} \underline{v}_j W_h^{ij} \omega_j \tag{4.89}$$

Of course, one is free to replace W with some higher order scheme like RKPM oder CSPM. Indeed, such a method was explored by Randles and Libersky in [213] employing a Moving Least Squares scheme.

However, stress points were first introduced by Dyka and colleagues in [66], [67] and [68] as a measure to prevent the tensile instability. It was then shown by Belytschko and co-workers in [23] that stress points can't prevent tensile instability in an updated Lagrangian framework, even in one dimension.

Vignjevic and colleagues re-introduced stress points in [268] as a means to suppress zero energy modes. They argue that they prevent such modes since, employing two different sets of collocation points, zero energy modes would need to be present at two different discretizations on two different field variables. While no formal proof was given to this end, the argument at least makes sense on an intuitive level. They show some quite involved two dimensional impact simulations to demonstrate the viability of their approach.

Stress points are not considered in this work since they where shown to be unable to prevent zero energy modes, and weak form methods, wich *are* considered in this work, de-collocate the field variables in the same fashion if a separate set of quadrature points is employed for the integration of the weak form. Note: stress points and quadrature points in weak forms might be similar concepts, however they are clearly separated by the fact that stress points carry their own smoothing kernel W, i.e. $\exists W_h^{ki}$, but $\nexists W_h^{qi}$ if $q = 1 \dots K$ is the set of quadrature points.

4.7.5 *Transport Velocity Formulation*

A quite new development in meshless methods are so called Transport Velocity Formulations (TVF). They stem from the CFD community and were introduced by Adami et al. in [2]. In essence, their idea is to introduce a "transport velocity" $\underline{\tilde{v}} \neq \underline{v}$ with which the particles are moved:

$$\underline{x}_i = \underline{\tilde{v}}_i \tag{4.90}$$

the transport velocity is

$$\underline{\tilde{v}}_i = \tilde{v}_i + \Delta t \left(\underline{\dot{v}}_i - \langle \nabla p_b \rangle / \varrho_i \right) \tag{4.91}$$

where the standard momentum equation for the physics at hand is used for $\underline{\dot{v}}$ and p_b is a background pressure, tunable to the problem at hand. It is constant throughout the field,

thus $\nabla p_b = 0$. However $\langle p_b \rangle \neq 0$ since it is approximated using SPH. That is, the method would not work using RKPM or CSPM for that term. It is argued that this extra term prevents tensile instabilities since it is repulsive in clustered regions with negative pressure.

Recently, this approach was generalized and applied to solid mechanics in [282]. It was shown to outperform the artificial stress approach by [91]. However, a definite tensile test proving that the approach completely eliminates the tensile instabiliy has so far been absent from literature.

5

ALGORITHMS

So far the discussion of the physics to be implemented, that is, the PDE's to be solved, was kept separate from the remarks on meshless methods, wherever possible. In this section this gap will be closed. Specific meshless methods will be applied to solve the PDE systems 3.1 in section 5. Afterwards a short discussion on the radial return algorithm follows, specifically what root solver is to be used. This is followed by a discussion on contact algorithms that is mainly focused on how to efficiently search for contact pairs. Some special attention is given on how to enable contact in a material separation context for Total Lagrangian methods as well as on how to avoid highly oscillatory friction forces.

5.1 THE ELASTIC STAGE

A wide array of algorithms and their implementation details is presented:

- Updated Lagrangian, Strong Form
  - Gray and Monaghans stress corrected algorithm in [91], with some variations.
  - A Transport Velocity Formulation (TVF) algorithm for solid mechanics according to [282].
  - Godunov SPH according to Parshikov et al. [200]
- Total Lagrangian, Strong Form
  - Reveles Total Lagrangian algorithm presented in [217]
  - An algorithm based on the elastic potential force instead of the standard momentum equation, inspired by [178]
  - An algorithm inspired by the corotated formulation in [19]
- Updated Lagrangian, Weak Form
  - An algorithm similar to the stress point algorithm in [268]
- Total Lagrangian, Weak Form
  - The explicit RKPM given in [127]

The implementation and comparison of these algorithms, be it isolated in the elastic stage or combined with contact, plasticity and heat conduction phenomena is one of the major contributions of this thesis. Note that all possible combinations between total Lagrangian, updated Lagrangian, strong and weak forms are explored in this thesis.

A quick note on notation: Notation is kept as brief as possible in this chapter. The shorthands introduced previously, i.e. $f_i = f(\underline{x}_i)$ for function values and $W_h^{ij} = W_{ij} = W(\underline{x}_i - \underline{x}_j, h)$ for the kernel are used extensively. Particle volumes m/ϱ and integrations weights ω are used interchangeably, whenever convenient.

5.1.1 *Algorithm 1: Monaghans Artificial Stresses and Some Notes on Conservation Properties*

At the heart of Monaghans algorithm is an SPH discretization of (3.27) and (3.39) as follows:

$$\langle \dot{\underline{v}} \rangle_i = \sum_{j=1}^{N} \left(\frac{\underline{\underline{\sigma}}_i}{\varrho_i^2} + \frac{\underline{\underline{\sigma}}_j}{\varrho_j^2} \right) \cdot \nabla W_{ij} m_i + \underline{b}_i \tag{5.1}$$

$$\langle \dot{\varrho} \rangle_i = \varrho_i \sum_{j=1}^{N} \left(\underline{v}_j - \underline{v}_i \right) \cdot \nabla W_{ij} \frac{m_j}{\varrho_j} \tag{5.2}$$

$$\langle \underline{\underline{L}} \rangle_i = \sum_{j=1}^{N} \left(\underline{v}_j - \underline{v}_i \right) \otimes \nabla W_{ij} \frac{m_j}{\varrho_j} \tag{5.3}$$

that is, SPH approximator (4.32) was applied to (3.27) and SPH approximator (4.34) was applied to (3.39). The last equation (5.3) is needed for the material model, specifically for the Jaumann rate, c.f. (3.77). Why this particular choice was made becomes clear if some relevant conservation properties, i.e. the conservation of linear momentum and the Galilean invariance are studied:

- Galilean invariance is ensured readily by (5.2) as well as (5.3), since in a pure displacement field $\underline{v}_i = \underline{v}_j$, and both (5.2) and (5.3) are null in this case.
- That linear momentum is conserved can be seen by simply writing the linear momentum of a discrete particle ensemble:

  $$\underline{\mathcal{P}} = \sum_{i=1}^{N} m_i \underline{v}_i \tag{5.4}$$

  then investigating its change over time:

  $$\dot{\underline{\mathcal{P}}} = \sum_{i=1}^{N} m_i \dot{\underline{v}}_i \tag{5.5}$$

  inserting (5.1):

  $$\dot{\underline{\mathcal{P}}} = \sum_{i=1}^{N} \sum_{j=1}^{N} \left(\frac{\underline{\underline{\sigma}}_i}{\varrho_j^2} + \frac{\underline{\underline{\sigma}}_j}{\varrho_i^2} \right) \nabla W_{ij} m_j m_i \tag{5.6}$$

the sum is now rearranged:

$$\dot{\underline{\mathcal{P}}} = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left[\left(\frac{\underline{\underline{\sigma}}_i}{\varrho_i^2} + \frac{\underline{\underline{\sigma}}_j}{\varrho_j^2}\right)\nabla W_{ij} m_j m_i + \left(\frac{\underline{\underline{\sigma}}_j}{\varrho_j^2} + \frac{\underline{\underline{\sigma}}_i}{\varrho_i^2}\right)\nabla W_{ji} m_i m_j\right] \tag{5.7}$$

which is again null since $\nabla \cdot W_{ij} = -\nabla \cdot W_{ji}$.

An interesting question which naturally arises is how this particular arrangement of SPH approximators was found. Or in other words if there is a more systematic way of designing a set of SPH approximation equations that fulfills some relevant conservation properties given a set of PDEs than simple trial and error. Price [205] shows a that this is indeed possible by means of the Lagrangian of a physical system. Note the fact that the conservation of linear momentum only works out if $\nabla \cdot W_{ij} = -\nabla \cdot W_{ji}$. That is, if a corrected kernel, e.g. $\overset{\circ}{W}$ is used, $\nabla \cdot \overset{\circ}{W}_{ij} \neq -\nabla \cdot \overset{\circ}{W}_{ji}$ and linear momentum is not conserved. However, it turns out that a corrected kernel of at least first order is needed to correctly resolve rigid body rotations, since a rotational field is linear in each coordinate and a first order kernel is needed to obtain a null gradient for linear fields. Otherwise (5.3) will not be null in rotational fields and induce stresses by means of (3.77). This can only be fixed by co-rotational schemes, see the discussion by Solenthaler [243] [230] and section 5.1.5. Still, this reveals one of the most troublesome conundrums with regard to particle methods: conservation of linear momentum and linear/first order completeness *is mutually exclusive*! Price arrives at the same conclusion in [205]. There are other reasons why choosing $\overset{\circ}{W}_{ij}$, or any other linear complete kernel, over W_{ij} may be detrimental:

- If (5.1) is used with uncorrected kernel W; unphysical particle arrangements with zero pressure revert to regular ensembles, a fact which is exploited by TVF algorithms, see section 5.1.2
- Again, if (5.1) is used with uncorrected kernel W stress free boundaries $\underline{\underline{\sigma}} \cdot \underline{n} = 0$ are fulfilled implicitly due to the boundary deficiency of W.

An investigation into these matters is subject of section 7.1. This concludes a short digression into conservation properties and kernel functions.

Back onto the topic of Monaghans algorithm, which is kept uncorrected for now: In order to suppress the tensile instability, not only is (5.1) extended by artificial viscosity Π but also artificial stresses $\underline{\underline{\Theta}}$, see sections 4.7.1 and 4.7.2:

$$\langle \dot{\underline{v}} \rangle_i = \sum_{j=1}^{N}\left(\frac{\underline{\underline{\sigma}}_i}{\varrho_i^2} + \frac{\underline{\underline{\sigma}}_j}{\varrho_j^2} + \Pi_{ij}\underline{\underline{I}} + \underline{\underline{\Theta}}_{ij}\right)\cdot \nabla W_{ij} m_j \tag{5.8}$$

To further stabilize the solution the advection equation is replaced by its XPSH version, see section 4.7.3:

$$\dot{\underline{x}}_i = \underline{v}_i + \epsilon^{\text{XSPH}}\sum_{j=1}^{N}\frac{m_j}{\varrho_i + \varrho_j}(\underline{v}_j - \underline{v}_i)W_{ij} \tag{5.9}$$

5.1.2 *Algorithm 2: A Transport Velocity Formulation for Solid Mechanics*

This algorithm is quite similar to in the sense that the same set of SPH approximators is used, and both artificial viscosity and XSPH are employed. What differs is that each particle carries two velocities. Its "normal" velocity, evolved by the momentum equation, relabeled $\underline{v}^{\text{mom}}$ in this section for clarity, and a transport-, or advection velocity $\underline{v}^{\text{adv}}$. The advection velocity is used in all balance equations. This turns (5.1) - (5.3) into:

$$\dot{\underline{x}}_i = \underline{v}_i^{\text{adv}} + \epsilon^{\text{XSPH}} \sum_{j=1}^{N} \frac{m_j}{\varrho_i + \varrho_j} (\underline{v}_j^{\text{adv}} - \underline{v}_i^{\text{adv}}) W_{ij} \tag{5.10}$$

$$\langle \dot{\underline{v}}^{\text{mom}} \rangle_i = \sum_{j=1}^{N} \left(\frac{\underline{\underline{\sigma}}_i}{\varrho_i^2} + \frac{\underline{\underline{\sigma}}_j}{\varrho_j^2} + \Pi_{ij} \underline{\underline{I}} \right) \cdot \nabla W_{ij} m_j + \underline{b}_i \tag{5.11}$$

$$\langle \dot{\varrho} \rangle_i = \varrho_i \sum_{j=1}^{N} \left(\underline{v}_j^{\text{adv}} - \underline{v}_i^{\text{adv}} \right) \cdot \nabla W_{ij} \frac{m_j}{\varrho_j} \tag{5.12}$$

$$\langle \underline{\underline{L}} \rangle_i = \sum_{j=1}^{N} \left(\underline{v}_j^{\text{adv}} - \underline{v}_i^{\text{adv}} \right) \otimes \nabla W_{ij} \frac{m_j}{\varrho_j} \tag{5.13}$$

the transport velocity is obtained by modifying the momentum velocity:

$$\underline{v}_i^{\text{adv}} = \underline{v}_i^{\text{mom}} + \Delta t \left(\langle \dot{\underline{v}}^{\text{mom}} \rangle_i - \left\langle \nabla p^0 \right\rangle_i / \varrho_i \right) \tag{5.14}$$

where the p^0 is a background pressure. Its gradient is discretized by:

$$\left\langle \nabla p^0 \right\rangle_i / \varrho_i = p_i^0 / \varrho_i^2 \sum_{j=1}^{N} m_j \cdot W \left(\underline{x}_i - \underline{x}_j,\ 0.5 \cdot h \right) \tag{5.15}$$

and $p_i^0 = min\left(10 \cdot |p_i|, p_{\text{ref}}\right)$. Both this particular choice of background pressure and discretization of its gradient (5.15) are modifications of the original algorithm [2] introduced by Zhang and co-workers [282]. While employing a zero order gradient of a constant field to regularize particle positions, as was originally suggested in [2], certainly seems reasonable, at least for fluids, the same seems at least dubious for elastic solids, where the hydrostatic pressure only describes a part of the dynamics at hand. There is also no reasoning given why the particular choice of *non*-constant background pressure is a good one. Results reported in [282] seem to at least be en par with the artificial stress correction by Monaghan [91] from the previous section 5.1.1.

The Transport Velocity Formulation for solid mechanics is a recent development, and the applications in the original work [282] and in this thesis are the first of their kind.

5.1.3 *Algorithm 3 - Godunov SPH*

The realization that SPH can be elegantly combined with a Riemann solver was first made by Inutsuka in 1994 [115], but greatly popularized by Monaghan in 1997 [167], who does not

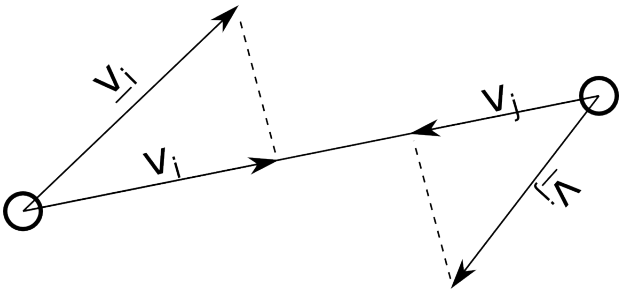

Figure 5.1: Vector quantities like the velocity are projected onto the inter-particle distance in Godunov SPH and can then be regarded scalar

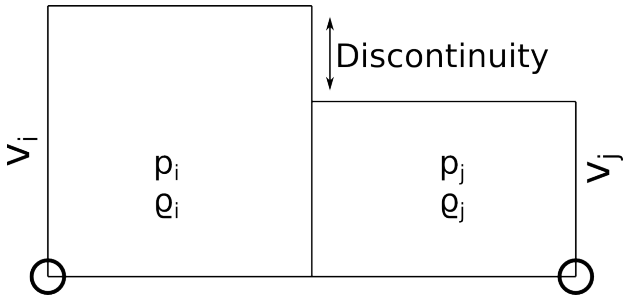

Figure 5.2: Alternative view of an SPH particle interaction: two one dimensional bodies with different velocities, densities and pressures are in contact. The contact point constitutes a discontinuity in the field variables.

seem to be aware of Inutsukas' work at the time of writing. The first application to solid mechanics followed quite a bit later in 2002 by Parshikow [200]. The algorithm therein is first order, but was extended to second order in [50], [49]. However, their approach requires a custom time splitting procedure, leaving unclear how to incorporate the scheme into standard time integrators. Occasionally there are still works published employing Godunov type SPH schemes. To cite a few examples: Cha simualtes the Kelvin-Helmholtz instability using Godunov SPH in [35], Puri benchmarks different Riemann solvers for Godunov SPH applied to Gas Dynamics in [208] and Mehra shows that Godunov SPH is superior to standard SPH in impact problems in [160]. However, it seems that there are no major works on Godunov SPH regarding solid mechanics since Conollys second order extension [50] [49].

To illustrate the idea of Godunov SPH the continuity equation (3.27) discretized by (4.32) is considered:

$$\langle \dot{\varrho} \rangle_i = \varrho_i \sum_{j=1}^{N} \left(\underline{v}_j - \underline{v}_i \right) \cdot \nabla W_{ij} \frac{m_j}{\varrho_j} \tag{5.16}$$

If this is rewritten expanding the inner derivative of ∇W_{ij}:

$$\langle \dot{\varrho} \rangle_i = \varrho_i \sum_{j=1}^{N} \left(\underline{v}_j - \underline{v}_i \right) \cdot \underbrace{\left(\frac{\underline{x}_{ij}}{|\underline{x}_{ij}|} \frac{\partial W_{ij}}{\partial \underline{x}_{ij}} \right)}_{\nabla W_{ij}} \frac{m_j}{\varrho_j} \tag{5.17}$$

Rearranging the brackets slightly

$$\langle \dot{\varrho} \rangle_i = \varrho_i \sum_{j=1}^{N} \underbrace{\left(\left(\underline{v}_j - \underline{v}_i \right) \cdot \frac{\underline{x}_{ij}}{|\underline{x}_{ij}|} \right)}_{A} \frac{\partial W_{ij}}{\partial \underline{x}_{ij}} \frac{m_j}{\varrho_j} \tag{5.18}$$

reveals that the term marked with A is a projection operation. That is, the velocities $\underline{v}_i$, $\underline{v}_j$ are projected on inter particle distance $\underline{x}_{ij}$. The velocity can now be regarded as scalar,

the length of each projection that is. This is illustrated in figure 5.1. The situation at hand can now be regarded as an initial value problem with regard to projected velocities v, with a single discontinuity along line $\underline{x}_{ij}$, combined with a conservation law (3.27), i.e. mass conservation/the continuity equation. These are exactly the conditions enabling a solution by a Riemann solver. An introduction to Riemann solvers can be found in standard texts such as [263] and [223]. A sketch of this kind of problem is given in figure 5.2.

It is not immediately clear why recasting the SPH approximators into Riemann problems would be beneficial. However, it is commonly argued, for example in [49], that algorithms constructed in this fashion do not need artificial viscosity schemes or smoothing schemes like XSPH. This is attractive because it greatly reduces the number of user tunable parameters. In light of the findings in [122], showing that the choice of these paremeters can severly influence the physics simulated, this is certainly an attractive property. Additionaly, as already mentioned, Godunov SPH was shown to be a good match for impact problems in [160]. While metal cutting and impact problems are quite different situations, there are still some common features, like the very large and violent deformations encountered. This suggests that algorithms that are a good match for impact problems might do well in metal cutting as well.

It has been clarified that (5.18) is, indeed, a Riemann problem. It remains to show how the solution of the Riemann problem can be incorporated into (5.18). A simple solution to the Riemann problem at hand is the acoustic approximation [218]:

$$v_{ij}^{*R} = \frac{v_j^R \varrho_j c_j + v_i^R \varrho_i c_i - p_j + p_i}{\varrho_j c_j + \varrho_i c_i} \tag{5.19}$$

where $(\cdot)^R$ is a projected quantity, and $(\cdot)^*$ denotes a Riemann solution. For example for the projected velocity would read:

$$v^R = \underline{v} \cdot \frac{\underline{x}_{ij}}{|\underline{x}_{ij}|} \tag{5.20}$$

Hence the underline was dropped. Now, it holds that:

$$v_{ij}^{*R} = 1/2 \left(v_i^R + v_j^R \right) \tag{5.21}$$

using (5.19) in (5.18) and substituting (5.21):

$$\begin{aligned}
\langle \dot{\varrho} \rangle_i &= \varrho_i \sum_{j=1}^{N} \left(v_j^R - v_i^R \right) \frac{\partial W_{ij}}{\partial \underline{x}_{ij}} \frac{m_j}{\varrho_j} \\
\langle \dot{\varrho} \rangle_i &= \varrho_i \sum_{j=1}^{N} \left(v_i^R + v_j^R - 2v_i^R \right) \frac{\partial W_{ij}}{\partial \underline{x}_{ij}} \frac{m_j}{\varrho_j} \\
\langle \dot{\varrho} \rangle_i &= \varrho_i \sum_{j=1}^{N} \left(2v_{ij}^{*R} - 2v_i^R \right) \frac{\partial W_{ij}}{\partial \underline{x}_{ij}} \frac{m_j}{\varrho_j}
\end{aligned} \tag{5.22}$$

In summary, equation set (5.1) - (5.3) is turned into:

$$\dot{\underline{x}}_i = \underline{v}_i \tag{5.23}$$

$$\langle \dot{\underline{v}} \rangle_i = -\frac{2}{\varrho_i} \sum_{j=1}^{N} \left(\underline{\sigma}_{ij}^{*R} \right) \frac{\partial W_{ij}}{\partial \underline{x}_{ij}} \frac{m_j}{\varrho_j} + \underline{b}_i \tag{5.24}$$

$$\langle \dot{\varrho} \rangle_i = -2\varrho_i \sum_{j=1}^{N} \left(v_i^R - v_{ij}^{*R} \right) \frac{\partial W_{ij}}{\partial \underline{x}_{ij}} \frac{m_j}{\varrho_j} \tag{5.25}$$

$$\langle \underline{\underline{L}} \rangle_i = \sum_{j=1}^{N} \left(\underline{v}_j - \underline{v}_i \right) \otimes \nabla W_{ij} \frac{m_j}{\varrho_j} \tag{5.26}$$

The advection equation (5.23) was stated explicitly to highlight that the algorithm manages without any smoothing schemes like XSPH. Note that (5.26) does not contain a Riemann solution since there is no balance equation attached to the computation of a velocity gradient. Also note that the procedure to find σ_{ij}^{*R} is quite a bit more involved than for v_{ij}^{*R}, since projection by $\underline{x}_{ij}/|\underline{x}_{ij}|$ yields a vector quantity in this case. The exact procedure is given in summary 5.2.3.

5.1.4 *Algorithm 4 - Total Lagrangian SPH according to Reveles*

So far all algorithms presented work on the updated Lagrangian framework. As discussed, a total Lagrangian formulation might be advantageous, since it is free from tensile instability, see the discussion in 4.6 and the publication [23]. Reveles presents a detailed account of a total Lagrangian SPH solver in his PhD thesis, see [217]. His solver was re-implemented for this work, with some changes. In the core of the method remain the following semidiscrete equations:

$$\underline{u}_i = \underline{x}_i - \underline{X}_i \tag{5.27}$$

$$\langle \dot{\underline{v}} \rangle_i = \sum_{j=1}^{N} \left(\frac{\underline{\underline{P}}_i}{\varrho_i^2} + \frac{\underline{\underline{P}}_j}{\varrho_j^2} + \Pi_{ij} \cdot \underline{\underline{I}} \right) \cdot \nabla_0 \overset{\triangle}{W}_{ij} m_i + \underline{b}_i \tag{5.28}$$

$$\langle \varrho_i \rangle = J \cdot \varrho_i^0 \tag{5.29}$$

$$\langle \dot{\underline{\underline{F}}} \rangle_i = \sum_{j=1}^{N} \left(\underline{v}_j - \underline{v}_i \right) \otimes \nabla_0 \overset{\triangle}{W}_{ij} \frac{m_j}{\varrho_j^0} \tag{5.30}$$

$$\langle \underline{\underline{F}} \rangle_i = \sum_{j=1}^{N} \left(\underline{u}_j - \underline{u}_i \right) \otimes \nabla_0 \overset{\triangle}{W}_{ij} \frac{m_j}{\varrho_j^0} + \underline{\underline{I}} \tag{5.31}$$

where (5.30) is used to find $\underline{\underline{L}} = \dot{\underline{\underline{F}}} \cdot \dot{\underline{\underline{F}}}^{-1}$, which goes into (3.77) to compute the Jaumann rate and (5.31) is used to find $J = det\left(\underline{\underline{F}}\right)$ in (5.29). Note that the Randles-Libersky correction was used for all derivatives. Also note that all derivatives are with regard to the reference coordinates. That is, the neighbor set of each particle, the kernel and its derivatives need to be computed only once in the beginning of the simulation and can subsequently be cached. This promises a highly efficient algorithm.

As for changes to the original work: Reveles uses a non-standard timestepper that is not given by name in his work, that features seemingly arbitrary forward and backward integration to half timesteps. This was changed to either a Runge-Kutta scheme or the Leap Frog algorithm, depending on the application. Furthermore, the material model was changed to use the Jaumann rate instead of direct computation of the stress by means of the Green-St. Venant strain tensor.

5.1.5 *Algorithm 5 - Total Lagrangian SPH with Co-Rotation*

Co-rotated methods are well known within the FEM community, see for example standard texts such as [25]. Co-rotated methods do not employ an objective stress rate such as the Jaumann rate but still retain frame invariance by computing the stress tensor, e.g. the Cauchy stress, in co-rotated coordinates. The amount of rotation contained in the current deformation is needed. This is conventionally achieved by considering the polar decomposition of $\underline{\underline{F}}$, see also chapter 3:

$$\underline{\underline{F}} = \underline{\underline{R}} \cdot \underline{\underline{U}} \tag{5.32}$$

The quantities going into the computation of the stress tensor can then be un-rotated, for example:

$$\hat{\underline{\underline{L}}} = \nabla \left(\underline{\underline{R}}^T \cdot \underline{v} \right) \tag{5.33}$$

where the convention that $\hat{(\cdot)}$ denotes a co-rotated quantity is introduced. If these co-rotated quantities are used, the momentum equation (3.39) now reads:

$$\hat{\underline{a}} = 1/\varrho \nabla \hat{\underline{\underline{\sigma}}} + \underline{b} \tag{5.34}$$

$$\dot{\underline{v}} = \underline{\underline{R}} \cdot \hat{\underline{a}} \tag{5.35}$$

Obtaining $\underline{\underline{R}}$ by means of the polar decomposition of $\underline{\underline{F}}$ is not the only option, see for example [274] and [21]. Becker notes in [19] that the current rotation of a particle can also be estimated using a shape matching procedure, originally proposed by Müller in [177], which is well suited for meshless methods. An SPH approximation to a local transformation matrix $\underline{\underline{\mathcal{A}}}$ is given as:

$$\underline{\underline{\mathcal{A}}}_i = \sum_{j=1}^{N} W_{ij} \cdot \left(\underline{x}_j - \underline{x}_i \right) \otimes \left(\underline{X}_j - \underline{X}_i \right) m_j \tag{5.36}$$

Note that the bracketed term $\left(\underline{x}_j - \underline{x}_i \right)$ is not the argument to W. It is now suggested to use the polar decomposition on $\underline{\underline{\mathcal{A}}}$ instead of $\underline{\underline{F}}$ to find $\underline{\underline{R}}$. Hence, for this algorithm:

$$\underline{\underline{\mathcal{A}}} = \underline{\underline{R}} \cdot \underline{\underline{U}} \tag{5.37}$$

This rotation matrix can be used to compute co-rotated deformation gradient:

$$\hat{\underline{u}}_{ij} = \underline{\underline{R}}_i^T \left(\underline{x}_j - \underline{x}_i \right) - \left(\underline{X}_j - \underline{X}_i \right) \tag{5.38}$$

$$\hat{\underline{\underline{F}}}_i = \sum_{j=1}^{N} \hat{\underline{u}}_{ij} \nabla_0 W_{ij} \frac{m_j}{\varrho_j^0} + \underline{\underline{I}} \tag{5.39}$$

this replaces (5.30) in Reveles' algorithm.

As mentioned, co-rotated methods work without using an objective stress rate. Hence, the usual material model (3.77) does not apply in this case. Instead, (5.39) is used to build the linear Cauchy-Green strain tensor, see (3.11):

$$\underline{\underline{\hat{E}}}^{\text{lin}} = \frac{1}{2}\left(\underline{\underline{\hat{F}}}^{T} + \underline{\underline{\hat{F}}}\right) + \underline{\underline{I}} \tag{5.40}$$

which is then used to compute $\underline{\underline{\hat{\sigma}}}$ directly:

$$\underline{\underline{\hat{\sigma}}} = \underline{\underline{\underline{\underline{C}}}} : \underline{\underline{\hat{E}}}^{\text{lin}} \tag{5.41}$$

which simplifies to

$$\underline{\underline{\hat{\sigma}}} = 2G \cdot \underline{\underline{\hat{E}}}^{\text{lin}} + \lambda \cdot tr(\underline{\underline{\hat{E}}}^{\text{lin}}) \tag{5.42}$$

for the materials considered. This is transformed using (3.18) to find $\underline{\underline{\hat{P}}}$. This would enable the procedure outlined in (5.34)- (5.35), with (3.42) instead of (3.39). However, the authors in [19] chose to do an additional symmetrization step:

$$\underline{\dot{v}}_i = 1/2 \sum_{j=1}^{N} \left(-\underline{\underline{R}}_i \cdot \underline{\dot{v}}_{ji} + \underline{\underline{R}}_j \cdot \underline{\dot{v}}_{ji}\right) \underline{b}_i \tag{5.43}$$

where $\underline{\dot{v}}_{ij}$ is the acceleration on particle i caused by particle j. The above (5.43) entails that that $\underline{\dot{v}}_{ij} = -\underline{\dot{v}}_{ji}$, i.e. that linear momentum is conserved. This is not the case if any linear correction scheme was used. The algorithm at hand is remarkable in the sense that it is not first order complete, does not use a strain tensor which correctly measures rotations, but still resolves rigid body modes by using a co-rotated formulation by means of a shape matching procedure.

Note that in the original implementation, the authors [19] do not solve the momentum equation but use elastic potential forces as outlined in 5.1.6

5.1.6 *Algorithm 6 - Total Lagrangian SPH by Elastic Potential*

Another algorithm quite popular with the computer graphics community is given by Müller in [178]. The algorithm is quite different from the others considered in the sense that it does not solve neither momentum equation (3.39) nor (3.42), but starts at the more fundamental equation:

$$\mathcal{U} = 1/2 \left(\underline{\underline{\sigma}} : \underline{\underline{E}}\right) \tag{5.44}$$

where $\mathcal{U}$ is the strain energy density function. This is true for any strain measure $\underline{\underline{E}}$. Now, the force per unit volume $\underline{\bar{f}}$ due to some potential function Ξ at some point $\underline{x}_i$ that was displaced by displacement $\underline{u}_i$ is the directional gradient in direction $\underline{u}_i$, i.e.:

$$\underline{\bar{f}}_i = -\nabla_{\underline{u}_i} \Xi \tag{5.45}$$

Setting $\Xi = \mathcal{U}$:

$$\underline{\bar{f}}_i = -1/2 \nabla_{\underline{u}_i} \left(\underline{\underline{\sigma}} : \underline{\underline{E}} \right) \tag{5.46}$$

inserting Hooke's law for isotropic media:

$$\underline{\bar{f}}_i = -1/2 \nabla_{\underline{u}_i} \left[\left(2G \cdot \underline{\underline{E}} + \lambda \cdot tr(\underline{\underline{E}}) \cdot \underline{\underline{I}} \right) : \underline{\underline{E}} \right] \tag{5.47}$$

distributing $\underline{\underline{E}}$, then $\nabla_{\underline{u}_i}$:

$$\underline{\bar{f}}_i = -1/2 \left[\nabla_{\underline{u}_i} \left(2G \cdot \underline{\underline{E}} : \underline{\underline{E}} \right) + \nabla_{\underline{u}_i} \left(\lambda \cdot tr(\underline{\underline{E}}) \cdot \underline{\underline{I}} : \underline{\underline{E}} \right) \right] \tag{5.48}$$

which equals

$$\underline{\bar{f}}_i = -1/2 \left[4G \cdot \underline{\underline{E}} : \nabla_{\underline{u}_i} \underline{\underline{E}} + 2\lambda \cdot tr(\underline{\underline{E}}) \cdot \underline{\underline{I}} : \nabla_{\underline{u}_i} \underline{\underline{E}} \right] \tag{5.49}$$

While

$$\nabla_{\underline{u}_i} \left(2G \cdot \underline{\underline{E}} : \underline{\underline{E}} \right) = 4G \cdot \underline{\underline{E}} : \nabla_{\underline{u}_i} \underline{\underline{E}} \tag{5.50}$$

is quite straight forward

$$\nabla_{\underline{u}_i} \left(\lambda \cdot tr(\underline{\underline{E}}) \cdot \underline{\underline{I}} : \underline{\underline{E}} \right) = 2\lambda \cdot tr(\underline{\underline{E}}) \cdot \underline{\underline{I}} : \nabla_{\underline{u}_i} \underline{\underline{E}} \tag{5.51}$$

is best understood by simply expanding the components, e.g. in 2D:

$$\begin{aligned} \lambda \nabla_{\underline{u}_i} \left((E_{xx} + E_{yy}) E_{xx} + (E_{xx} + E_{yy}) E_{yy} \right) = \\ \lambda \begin{bmatrix} 2 \cdot (\partial / (\partial u_x^i) E_{xx}) E_{xx} + 2 \cdot (\partial / (\partial u_x^i) E_{xx}) E_{yy} \\ 2 \cdot (\partial / (\partial u_y^i) E_{yy}) E_{xx} + 2 \cdot (\partial / (\partial u_y^i) E_{yy}) E_{xx} \end{bmatrix} = \\ 2\lambda \cdot tr(\underline{\underline{E}}) \cdot \underline{\underline{I}} : \nabla_{\underline{u}_i} \underline{\underline{E}} \end{aligned} \tag{5.52}$$

Factoring out $2 \cdot \nabla_{\underline{u}_i} \underline{\underline{E}}$ from (5.49) yields:

$$\begin{aligned} \underline{\bar{f}}_i &= - \left[2G \cdot \underline{\underline{E}} + \lambda \cdot tr(\underline{\underline{E}}) \cdot \underline{\underline{I}} \right] : \nabla_{\underline{u}_i} \underline{\underline{E}} \\ &= -\underline{\underline{\sigma}} : \nabla_{\underline{u}_i} \underline{\underline{E}} \end{aligned} \tag{5.53}$$

Having arrived at the above (5.53), the Green St.-Venant strain tensor is used for $\underline{\underline{E}}$. The original work actually makes a stronger claim than what was just shown, i.e. that the above is true for the general Hooke's law $\underline{\underline{\sigma}} = \underline{\underline{\underline{C}}} : \underline{\underline{\hat{E}}}$, with no proof given. To this date, such a proof could not be found.

After a lot of quite involved simplification, see appendix 9.1, the following expression is retrieved for force $\underline{f}_j$ on particle j enacted by particle i:

$$\underline{f}_i = -2 \sum_{j=1}^{N} \underline{\underline{E}}_i \cdot \underline{\underline{\sigma}}_i \cdot \nabla \overset{\circ}{W}_{ij} \cdot m_i / \varrho_i \tag{5.54}$$

i.e. the RKPM corrected kernel function was used instead of the Moving Least Squares kernel suggested by [178]. Note that the multiplication by the volume of particle i is merely to turn the specific force $\underline{\bar{f}}$ into force $\underline{f}$. Additionally, the symmetrization procedure (5.43) used, albeit without rotations, i.e. $\underline{\underline{R}} = \underline{\underline{I}}$. What is remarkable about this algorithm is how little work needs to be done to simulate an elastic body with large deformations. Basically, $\underline{\underline{E}}$ is estimated at each particle using the pre-computed kernel derivatives, then (5.54) is computed at each particle, which completes the algorithm.

5.1.7 *Algorithm 7 - Weak Forms of the Constitutive Equations and an Updated Lagrangian Algorithm on the Weak Form*

So far, all algorithms presented operate on the strong form of the constitutive equations. This is as opposed to the FEM, which, in most cases, operates on the weak form. A misconception occasionally encountered is that there are only weak form FEM methods. See for example [262] for a counter example. A wide array of weak form methods in a meshless setting exist, often coined Meshfree or Element-Free Galerkin Methods (EFG). While a complete overview of such methods is beyond the scope of this chapter it is still worth mentioning that the first such method was introduced by Belytschko in 1994 [22]. In this section such a method is presented. The method works with a passively advected set of quadrature points, on which a simple trapezoidal rule / Riemann sum is performed instead of Gauss quadrature. While sacrificing accuracy of the integrand this enables the method to work without a background mesh, which would possibly need remeshing if large deformations are encountered, voiding the benefits of the meshless approach.

In any case, in order to understand the method, the weak form of the constitutive equation (3.39) needs to be developed. (3.39) is integrated over the whole domain and multiplied by a test function $\delta\underline{v}$:

$$\begin{aligned}\int_\Omega \delta\underline{v}(\underline{x}')\varrho(\underline{x}')\dot{\underline{v}}(\underline{x}')d\underline{x}' &= \int_\Omega \left[\delta\underline{v}(\underline{x}')\nabla\underline{\underline{\sigma}}(\underline{x}') + \underline{b}(\underline{x}')\varrho(\underline{x}')\right] d\underline{x}' \\ \int_\Omega \varrho\left(\delta\underline{v}\cdot\dot{\underline{v}}\right) d\underline{x}' &= \int_\Omega \delta\underline{v}\cdot\left(\nabla\underline{\underline{\sigma}} + \varrho\underline{b}\right) d\underline{x}'\end{aligned} \tag{5.55}$$

The test function and its gradient is known. Hence the above (5.55) needs to be manipulated to shift the gradients from unknown quantities $\underline{v}$, $\underline{\underline{\sigma}}$ to $\delta\underline{v}$. To this end, the first term on the right hand side of (5.55) is expanded using the product rule. Some care has to be taken to apply the product rule correctly in this case due to the vector and tensorial quantities. In a first step, this is expressed using index notation:

$$\int_\Omega \delta v_i \frac{\partial \sigma_{ji}}{\partial x_j} d\Omega = \int_\Omega \left[\frac{\partial}{\partial x_j}(\delta v_i \sigma_{ji}) - \frac{\partial(\delta v_i)}{\partial x_j}\sigma_{ji}\right] d\Omega \tag{5.56}$$

Which reveals that the products on the left hand side and the first term on the right hand side are combinations of dot products. The last term on the right hand side is quite cumbersome to write. There seems to be no way to express the product with standard vector notation The only way that was found to express this in vector notation is by introduction of the Hammard product:

$$\underline{a}\odot\underline{b} = \begin{bmatrix} a_x\cdot b_x \\ a_y\cdot b_y \\ a_z\cdot b_z \end{bmatrix} \tag{5.57}$$

i.e. component wise multiplication. Which allows to write the term as follows:

$$\frac{\partial(\delta v_i)}{\partial x_j}\sigma_{ji} = \left((\nabla\odot\delta\underline{v})\cdot\underline{\underline{\sigma}}\right)\cdot\underline{1} \tag{5.58}$$

However, most texts express this with the ill-defined notation:

$$\left((\nabla \odot \delta\underline{v}) \cdot \underline{\underline{\sigma}}\right) \cdot \underline{1} = (\nabla \delta\underline{v}) \cdot \underline{\underline{\sigma}} \tag{5.59}$$

In order to keep the expressions tractable, especially after discretization, this text follows this convention. Hence:

$$\int_\Omega \delta\underline{v} \cdot \left(\nabla \underline{\underline{\sigma}}\right) d\underline{x}' = \int_\Omega \left[\nabla \left(\delta\underline{v} \cdot \underline{\underline{\sigma}}\right) - (\nabla \delta\underline{v}) \cdot \underline{\underline{\sigma}}\right] d\underline{x}' \tag{5.60}$$

Now, the first term on the right hand side can now be subjected to Gauss' theorem:

$$\int_\Omega \nabla \left(\delta\underline{v} \cdot \underline{\underline{\sigma}}\right) d\underline{x}' = \int_\Gamma \delta\underline{v} \left(\underline{\underline{\sigma}} \cdot \underline{n}\right) d\underline{x}' \tag{5.61}$$

where $\bar{\underline{t}} = \underline{\underline{\sigma}} \cdot \underline{n}$ are the traction boundary conditions. Reinserting (5.60) into (5.55):

$$\int_\Omega \varrho \left(\delta\underline{v} \cdot \dot{\underline{v}}\right) d\underline{x}' + \int_\Omega (\nabla \delta\underline{v}) \cdot \underline{\underline{\sigma}}\, d\underline{x}' - \int_\Omega \delta\underline{v} \varrho \underline{b}\, d\underline{x}' - \int_\Gamma \delta\underline{v}\, \bar{\underline{t}}\, d\underline{x}' = 0 \tag{5.62}$$

This is the weak form of (3.39). It is not complete in the sense that discontinuous stresses inside the domain was ignored, see [25] for a derivation including this matter.

This weak form, defined over the continuum Ω, is now discretized into elements or particles i. Depending on the choice for $\delta\underline{v}$, different methods can be derived. If, for some elements or particles $\underline{x}_i$, $\underline{x}_j$, the test function $\delta\underline{v}_i(\underline{x}_j) = 0$ and $\delta\underline{v}_i(\underline{x}_i) = 1$, i.e. if the test function is interpolating in the strict sense and the test functions are not allowed to overlap, the FEM is retrieved. If the opposite is true, EFG methods are found. In the thesis at hand:

$$\delta\underline{v}(\underline{x}') = \delta\underline{v}_i \overset{\circ}{W}(\underline{x}_i - \underline{x}') = \delta\underline{v}_i \overset{\circ}{W}_i \tag{5.63}$$

$$\underline{v}(\underline{x}') = \underline{v}_i \overset{\circ}{W}(\underline{x}_i - \underline{x}') = \underline{v}_i \overset{\circ}{W}_i \tag{5.64}$$

that is, the RKPM kernel is used for both test and trial function. Inserting (5.64) and (5.63) into (5.62):

$$\delta\underline{v}_i \int_\Omega \varrho \left(\overset{\circ}{W}_i \overset{\circ}{W}_j \dot{\underline{v}}_j\right) d\underline{x}' + \delta\underline{v}_i \int_\Omega \left(\nabla \overset{\circ}{W}_i\right) \cdot \underline{\underline{\sigma}} d\underline{x}' - \delta\underline{v}_i \int_\Omega \overset{\circ}{W}_i \varrho \underline{b} d\underline{x}' + \delta\underline{v}_i \int_\Gamma \overset{\circ}{W}_i \varrho \bar{\underline{t}}\, d\underline{x} = 0 \tag{5.65}$$

since $\delta\underline{v}_i$ is arbitrary and $\underline{v}_j$ is not dependent on $\underline{x}'$, i.e. not bound by the integral, this simplifies to:

$$\int_\Omega \varrho \left(\overset{\circ}{W}_i \overset{\circ}{W}_j\right) d\underline{x}' \dot{\underline{v}}_j + \int_\Omega \left(\nabla \overset{\circ}{W}_i\right) \cdot \underline{\underline{\sigma}} d\underline{x}' - \int_\Omega \overset{\circ}{W}_i \varrho \underline{b} d\underline{x}' + \int_\Gamma \overset{\circ}{W}_i \varrho \bar{\underline{t}}\, d\underline{x} = 0 \tag{5.66}$$

which is a linear equation system in unknowns $\underline{v}_j$, $j = 1 \ldots N$ where N is the number of particles as usual. This is often written in compact matrix form:

$$\underbrace{\int_\Omega \varrho \left(\overset{\circ}{W}_i \overset{\circ}{W}_j\right) d\underline{x}'}_{\underline{\underline{M}}} \dot{\underline{v}}_j + \underbrace{\int_\Omega \left(\nabla \overset{\circ}{W}_i\right) \cdot \underline{\underline{\sigma}} d\underline{x}'}_{\underline{f}_{\text{Int}}} - \underbrace{\int_\Omega \overset{\circ}{W}_i \varrho \underline{b} d\underline{x}' + \int_\Gamma \overset{\circ}{W}_i \varrho \bar{\underline{t}}\, d\underline{x}}_{\underline{f}_{\text{Ext}}} = 0 \tag{5.67}$$

In summary:

$$\underline{\underline{M}}\,\underline{\dot{v}} + \underline{f}_{\text{Int}} = \underline{f}_{\text{Ext}} \tag{5.68}$$

Note that these systems are global, not per particle; for example the mass matrix $\underline{\underline{M}}$ has dimension $n_{\text{SD}} \cdot N \times n_{\text{SD}} \cdot N$ where n_{SD} is the number of spatial dimensions.

It is worthwhile to inspect the external forces term $\underline{f}_{\text{Ext}}$ with regard to its contents. Two terms are present, the body forces and boundary tractions. Boundary forces are present in this thesis in the shape of the contact forces introduced in section 5.4. Benchmarks with traction forces are not considered, i.e. the boundaries are said to be stress free $0 = \underline{\underline{\sigma}} \cdot \underline{n}$. The term is thus dropped / set to null. This entails that stress free boundaries are always respected in methods constructed from weak forms, even if linear complete functions are eventually used for $\delta\underline{v}$. See the discussion in 5.1.1.

$\underline{\underline{M}}$ is often subjected to a simplification procedure to improve the runtime characteristics of the algorithm, called mass lumping. A very simple approach is to diagonalize the matrix by taking the sum of rows of $\underline{\underline{M}}$. Or, even simpler, just setting all diagonal elements equal to $\underline{\underline{M}} = diag(m_i)$. Extensive information on mass lumping can be found in lecture notes [76] and a select comparison simulation is given in fig 5.4. Differences are small between no lumping and the two lumping schemes, and even smaller between the two lumping schemes themselves. Hence, the most simple scheme using uniform masses is used, especially since lumping to a diagonal matrix increases runtime efficiency about one order of magnitude. Note that this is a very rough timing to be taken with a grain of salt. While runtime does in fact increase about a factor of 10 using the "full" mass matrix, and the resulting equation system was solved using the highly optimized `UMFPACK` [55] routines, the underlying `BLAS` package was obtained using pre-compiled binaries and not compiled for the machine the benchmark was performed on.

Having clarified how to find the weak form of constitutive equations and how to incorporate meshless interpolation kernels into the test functions it remains to discuss how the integrals in (5.66) are approximated. The three most common options include:

1. Performing the intergration on the particles directly like in standard SPH

2. Gauss quadrature

3. Introduce a second set of particles having the same features as Gauss points, but are spaced arbitrarily. Perform a Riemann sum / trapezoidal rule on this second set of particles.

The first option enables zero energy modes while the second option necessitates a background mesh, possibly prone to remeshing. The last option offers an interesting compromise in the sense that it dispenses with the accuracy offered by Gauss quadrature but should avoid zero energy modes since the field variables are now split into two sets of discretization points. This is the approach which was implemented for this algorithm.

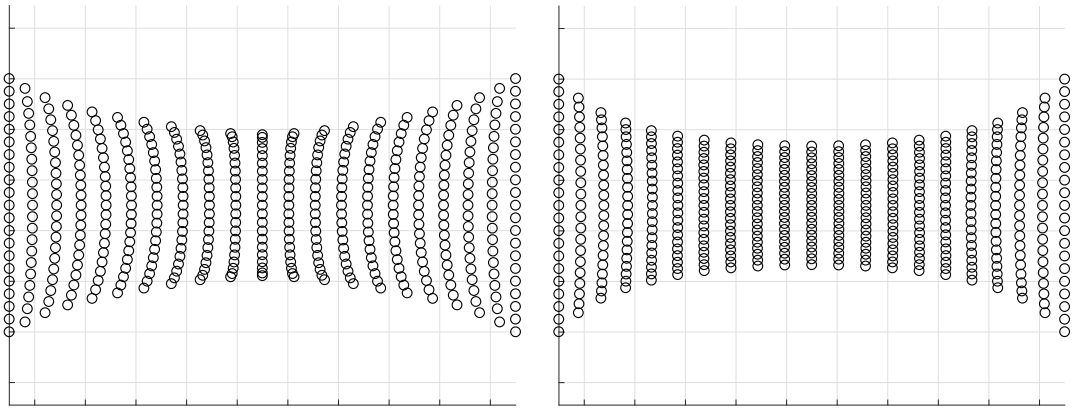

Figure 5.3: Linear elastic material model on the left and the modified Mooney-Rivlin material [81] model to the right in a tensile test using method 5.1.8/[127]. The differences are especially pronounced in the middle of the test specimen; constriction of the linear elastic material increases along x axis while the material following the Mooney-Rivlin model constricts more evenly.

5.1.8 *Algorithm 8 - Explicit Weak Form RKPM*

This algorithm is a quite close implementation of [127] by Belytschko, the main difference being the material model, which is kept linear elastic here but is extended to a large deformation modified Mooney-Rivlin model due to Johnson [81] in the original publication. The modified Mooney-Rivlin model allows for extremely large deformations, see 5.5. Since the algorithm is Total Lagrangian a background mesh of any complexity can be employed, without the need to remesh, since all computations are performed on the reference configuration. However, the implementation at hand is limited to simple quad meshes, using $\mathcal{Q}^{n_{\text{SD}}}$ quadrature points per quad element. These quadrature points are placed and weighted consistent with Gauss quadrature, making this the only algorithm in this work that enjoys a higher order quadrature scheme than the Trapezoidal rule / Riemann sum approach. The same mass lumping scheme as discussed in section 5.1.7 is used. To find the equations used by this algorithm the analysis in section 5.1.7 starting at (5.55) could be repeated based on the total Lagrangian momentum equation (3.42). However, the same can be achieved by using identities presented in chapter 3 directly on the semi-discrete equation in the updated Lagrangian frame derived above (5.66). Starting with the internal forces:

$$\int_{\Omega} \left(\nabla \overset{\circ}{W}_i \right) \cdot \underline{\underline{\sigma}} d\underline{x} = \int_{\Omega} \left(\nabla \overset{\circ}{W}_i \right) \cdot \frac{1}{J} \underline{\underline{F}} \cdot \underline{\underline{P}} d\underline{x} \tag{5.69}$$

where the transformation (3.18) from Cauchy to nominal stress was used. The integral can be transformed using the familiar relation:

$$\int_{\Omega} (\cdot)\, d\underline{x}' = \int_{\Omega_0} J\, (\cdot)\, d\underline{x}' \tag{5.70}$$

Hence:

$$\int_{\Omega} \left(\nabla \overset{\circ}{W}_i \right) \cdot \underline{\underline{\sigma}} d\underline{x} = \int_{\Omega_0} \left(\nabla \overset{\circ}{W}_i \right) \cdot \underline{\underline{F}} \cdot \underline{\underline{P}} d\underline{x} \tag{5.71}$$

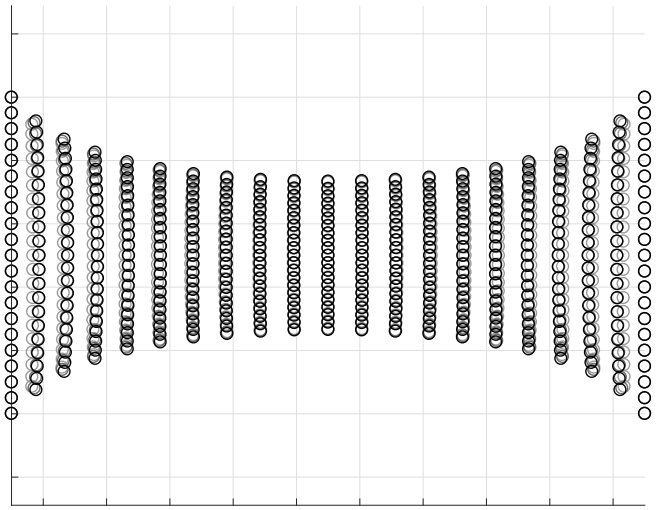

Figure 5.4: Different mass lumping schemes in a tensile test using method 5.1.8/[127]. To highlight how small the differences are the different options are plotted on top of each other instead of side by side: uniform masses in black, row sum lumping in dark gray and using the full matrix in light gray. Differences are well below 1%.

There is one more problem with this equation, the gradient in (5.69) refers to the current coordinates. This can be fixed by inspecting the term further:

$$\int_{\Omega_0} \left(\nabla \overset{\circ}{W}_i \right) \cdot \underline{\underline{F}} \cdot \underline{\underline{P}} d\underline{x} = \int_{\Omega_0} \left(\frac{\partial \overset{\circ}{W}_i}{\partial \underline{x}} \frac{\partial \underline{x}}{\partial \underline{X}} \right) \cdot \underline{\underline{P}} d\underline{x} = \int_{\Omega_0} \left(\nabla_0 \overset{\circ}{W}_i \right) \cdot \underline{\underline{P}} d\underline{x} \tag{5.72}$$

Due to mass conservation the mass matrix does not change, and can be used without alteration in either algorithm. See [25] for a formal proof regarding this matter.

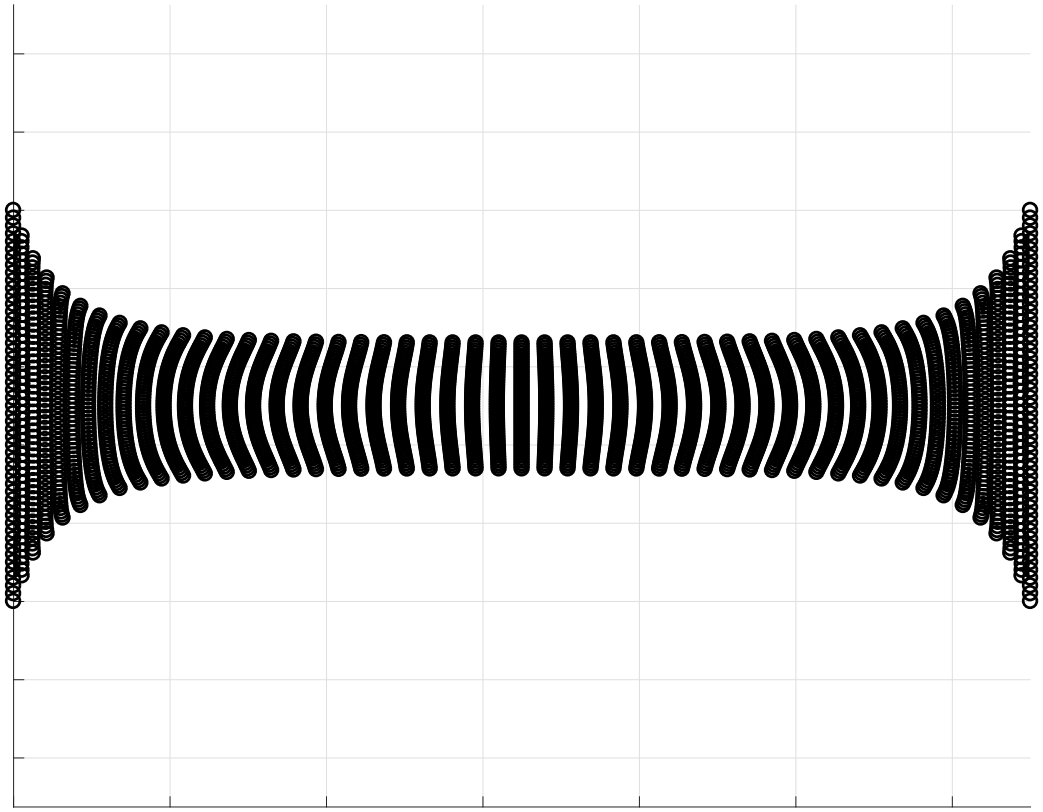

Figure 5.5: The RKPM algorithm described in conjunction with the modified Mooney-Rivlin model [81] allows for very large deformations. In fact, the stability limit of the algorithm could not be determined, tensile tests up to 20'000% extension remained stable. It is questionable how physically meaningful such extreme deformations are.

5.2 ALGORITHM SUMMARIES AND IMPLEMENTATION DETAILS

In this section all semi-discrete equations for each algorithm are summarized and some additional equation details are revealed. This is intended for quick reference and to enable comparison amongst the mathematical foundations of the algorithms at hand. The notation is kept even more compact, omitting the approximation operator $\langle\cdot\rangle$, implicitly understanding that a SPH-style sum on the right hand side does not yield an exact quantity but an approximated one.

5.2.1 *Algorithm 1*

Semi-discrete ODEs:

$$\begin{aligned}
\dot{\underline{x}}_i &= \underline{v}_i + \epsilon^{\text{XSPH}} \sum_{j=1}^{N} \frac{m_j}{\varrho_i + \varrho_j} (\underline{v}_j - \underline{v}_i) W_{ij} \\
\dot{\underline{v}}_i &= \sum_{j=1}^{N} \left(\frac{\underline{\underline{\sigma}}_i}{\varrho_i^2} + \frac{\underline{\underline{\sigma}}_j}{\varrho_j^2} + \Pi_{ij} \underline{\underline{I}} + \underline{\underline{\Theta}}_{ij} \right) \cdot \nabla W_{ij} m_i + \underline{b}_i \\
\dot{\varrho}_i &= \varrho_i \sum_{j=1}^{N} \left(\underline{v}_j - \underline{v}_i \right) \cdot \nabla W_{ij} \frac{m_j}{\varrho_j}
\end{aligned} \tag{5.73}$$

Artificial Viscosity:

$$\begin{aligned}
\Pi_{ij} &= \begin{cases} \frac{-\alpha^{av} \bar{c}_{ij} \mu_{ij} + \beta^{av} \mu_{ij}^2}{\bar{\varrho}_{ij}}, & \underline{v}_{ij} \cdot \underline{x}_{ij} < 0 \\ 0, & \underline{v}_{ij} \cdot \underline{x}_{ij} \geq 0 \end{cases} \\
\mu_{ij} &= \frac{h \underline{v}_{ij} \cdot \underline{x}_{ij}}{|\underline{x}_{ij}|^2 + \epsilon^{av}}
\end{aligned} \tag{5.74}$$

Artificial Stress:

$$\begin{aligned}
\underline{\underline{\sigma}}_i &= \underline{\underline{Q}}_i \cdot \underline{\underline{\Lambda}}_i \cdot \underline{\underline{Q}}_i^T \\
\overline{\Theta}_{\zeta\zeta}^i &= \begin{cases} -\epsilon^{\text{Stress}} \frac{\Lambda_{\zeta\zeta}^i}{\varrho_i^2} & \Lambda_{\zeta\zeta}^i > 0 \\ 0 & \Lambda_{\zeta\zeta}^i \leq 0 \end{cases} \qquad [\zeta\zeta \in \{xx, yy, zz\}] \\
\underline{\underline{\Theta}}_i &= \underline{\underline{Q}}_i^T \cdot \overline{\underline{\underline{\Theta}}}_i \cdot \underline{\underline{Q}}_i \\
\underline{\underline{\Theta}}_{ij} &= \left(\underline{\underline{\Theta}}_j + \underline{\underline{\Theta}}_i \right) \underbrace{\left(W_h(\underline{x}_i - \underline{x}_j)) / W_h(\Delta x) \right)^{n_{Stress}}}_{f^{n_{\text{Stress}}}}
\end{aligned} \tag{5.75}$$

with Eigen-decomposition $\underline{\underline{A}} = \underline{\underline{Q}} \cdot \underline{\underline{\Lambda}} \cdot \underline{\underline{Q}}^T$ and initial particle spacing Δx.

Default constants chosen are: $\alpha^{av} = \beta^{av} = 1$, $\epsilon^{av} = 0.1$, $\epsilon^{\text{XSPH}} = 0.5$, $\epsilon^{\text{Stress}} = 0.3$, $n_{\text{Stress}} = 4$

Material Model:

$$
\begin{aligned}
p_i &= c_0 \left(\varrho^i - \varrho_0^i \right) \\
\underline{\underline{\dot{S}}}_i &= 2G \cdot \left(\underline{\underline{D}}_i - 1/3 tr \left(\underline{\underline{D}}_i \right) \right) + \underline{\underline{W}}_i \cdot \underline{\underline{S}}_i - \underline{\underline{S}}_i \cdot \underline{\underline{W}}_i \\
\underline{\underline{D}}_i &= 1/2 \left(\underline{\underline{L}}_i + \underline{\underline{L}}_i^T \right) \\
\underline{\underline{W}}_i &= 1/2 \left(\underline{\underline{L}}_i - \underline{\underline{L}}_i^T \right) \\
\underline{\underline{L}}_i &= \sum_{j=1}^{N} \left(\underline{v}_j - \underline{v}_i \right) \otimes \nabla W_{ij} \frac{m_j}{\varrho_j} \\
\underline{\underline{\sigma}}_i &= \underline{\underline{S}}_i - \underline{\underline{I}} \cdot p_i
\end{aligned}
\tag{5.76}
$$

Note: W_{ij} can be replaced by $\overset{\triangle}{W}_{ij}$ to give up conservation of linear momentum but correctly resolve rigid body rotations. $\overset{\circ}{W}_{ij}$ leads to divergent simulations.

5.2.2 *Algorithm 2*

This algorithm introduces the concept of a transport or advection velocity. The usual velocity, i.e. determined by the momentum equation is denoted $\underline{v}^{\text{mom}}$, the advection velocity $\underline{v}^{\text{adv}}$:

$$
\begin{aligned}
\underline{\dot{x}}_i &= \underline{v}_i^{\text{adv}} + \epsilon^{\text{XSPH}} \sum_{j=1}^{N} \frac{m_j}{\varrho_i + \varrho_j} (\underline{v}_j^{\text{adv}} - \underline{v}_i^{\text{adv}}) W_{ij} \\
\underline{\dot{v}}_i^{\text{mom}} &= \sum_{j=1}^{N} \left(\frac{\underline{\underline{\sigma}}_i}{\varrho_i^2} + \frac{\underline{\underline{\sigma}}_j}{\varrho_j^2} + \Pi_{ij} \underline{\underline{I}} \right) \cdot \nabla W_{ij} m_j + \underline{b}_i \\
\dot{\varrho}_i &= \varrho_i \sum_{j=1}^{N} \left(\underline{v}_j^{\text{adv}} - \underline{v}_i^{\text{adv}} \right) \cdot \nabla W_{ij} \frac{m_j}{\varrho_j}
\end{aligned}
\tag{5.77}
$$

Artificial viscosity Π_{ij} is equal to (5.74), i.e. computed from $\underline{v}^{\text{mom}}$. Transport velocity:

$$
\begin{aligned}
\underline{\dot{v}}_i^{\text{adv}} &= \underline{\dot{v}}_i^{\text{mom}} - p_i^0 / \varrho_i^2 \sum_{j=1}^{N} m_j \cdot W \left(\underline{x}_i - \underline{x}_j,\ 0.5 \cdot h \right) \\
p_i^0 &= min(10 \cdot |p_i|, K)
\end{aligned}
\tag{5.78}
$$

The material model is equal to (5.76), but $\underline{\underline{L}}$ is computed from advection velocity $\underline{v}^{\text{adv}}$:

$$
\underline{\underline{L}}_i = \sum_{j=1}^{N} \left(\underline{v}_j^{\text{adv}} - \underline{v}_i^{\text{adv}} \right) \otimes \nabla W_{ij} \frac{m_j}{\varrho_j}
\tag{5.79}
$$

Correction constants with regard to XSPH and artificial viscosity are equal to the ones given in section 5.2.1.

5.2.3 *Algorithm 3*

Semi-discrete ODEs:

$$\begin{aligned}
\dot{\underline{x}}_i &= \underline{v}_i \\
\dot{\underline{v}}_i &= -\frac{2}{\varrho_i} \sum_{j=1}^{N} \left(\underline{\sigma}_{ij}^{*R} \right) \frac{\partial W_{ij}}{\partial \underline{x}_{ij}} \frac{m_j}{\varrho_j} + \underline{b}_i \\
\dot{\varrho}_i &= -2\varrho_i \sum_{j=1}^{N} \left(v_i^R - v_{ij}^{*R} \right) \frac{\partial W_{ij}}{\partial \underline{x}_{ij}} \frac{m_j}{\varrho_j}
\end{aligned} \tag{5.80}$$

Projected quantities $(\cdot)^R$ (vector $\rightarrow$ scalar, second order tensor $\rightarrow$ vector):

$$\begin{aligned}
v_i^R &= \underline{v}_i \cdot \frac{\underline{x}_{ij}}{|\underline{x}_{ij}|} \\
\underline{\sigma}_i^R &= \underline{\underline{\sigma}}_i \cdot \frac{\underline{x}_{ij}}{|\underline{x}_{ij}|}
\end{aligned} \tag{5.81}$$

Solutions to Riemann problem by acoustic approximation $(\cdot)^{*R}$. Note that the Riemann problem for the momentum equation is solved in a spherical coordinate system U, V, W with center $\underline{x}_i$. The coordinate transformation matrix from Cartesian coordinates x, y, z to spherical coordinates U, V, W is denoted $\underline{\underline{\mathcal{S}}}$:

$$\begin{aligned}
v_{ij}^{*R} &= \frac{v_j^R \varrho_j c_j + v_i^R \varrho_i c_i + p_i}{\varrho_j c_j + \varrho_i c_i} \\
\left[\sigma_i^{UU}, \sigma_i^{VU}, \sigma_i^{WU} \right]^T &= \underline{\underline{\mathcal{S}}} \cdot \underline{\sigma}_i^R \\
\underline{\sigma}_i^{*UVW} &= \begin{bmatrix} \frac{\sigma_j^{UU} \varrho_i c_i + \sigma_i^{UU} \varrho_j c_j + \varrho_i c_i \varrho_j c_j \left(v j^U - v i^U \right)}{\varrho_i c_i + \varrho_j c_j} \\ \frac{\sigma_j^{VU} \varrho_i c_i + \sigma_i^{VU} \varrho_j c_j + \varrho_i c_i \varrho_j c_j \left(v j^V - v i^V \right)}{\varrho_i c_i + \varrho_j c_j} \\ \frac{\sigma_j^{WU} \varrho_i c_i + \sigma_i^{WU} \varrho_j c_j + \varrho_i c_i \varrho_j c_j \left(v j^W - v i^W \right)}{\varrho_i c_i + \varrho_j c_j} \end{bmatrix} \\
\underline{\sigma}_{ij}^{*R} &= \underline{\underline{\mathcal{S}}}^{-1} \cdot \underline{\sigma}_i^{*UVW} = \underline{\underline{\mathcal{S}}}^T \cdot \underline{\sigma}_i^{*UVW}
\end{aligned} \tag{5.82, 5.83}$$

The material model is equal to (5.76).

5.2.4 *Algorithm 4*

Semi-discrete ODEs:

$$\underline{u}_i = \underline{X}_i - \underline{x}_i \tag{5.84}$$

$$\underline{\dot{v}i} = \sum_{j=1}^{N} \left(\frac{\underline{\underline{P}}_i}{\varrho_i^2} + \frac{\underline{\underline{P}}_j}{\varrho_j^2} + \Pi_{ij} \cdot \underline{\underline{I}} \right) \cdot \nabla_0 \overset{\triangle}{W}_{ij} m_i + \underline{b}_i$$

$$\varrho_i = J \cdot \varrho_i^0 \tag{5.85}$$

The material model stays the same as in (5.76), but some additional steps are required. For reference, the complete set of equations is stated here:

$$\begin{aligned}
p_i &= c_0 \left(\varrho^i - \varrho_0^i \right) \\
\dot{\underline{\underline{S}}}_i &= 2G \cdot \left(\underline{\underline{D}}_i - 1/3 tr \left(\underline{\underline{D}}_i \right) \right) + \underline{\underline{W}}_i \cdot \underline{\underline{S}}_i - \underline{\underline{S}}_i \cdot \underline{\underline{W}}_i \\
\underline{\underline{D}}_i &= 1/2 \left(\underline{\underline{L}}_i + \underline{\underline{L}}_i^T \right) \\
\underline{\underline{W}}_i &= 1/2 \left(\underline{\underline{L}}_i - \underline{\underline{L}}_i^T \right) \\
\dot{\underline{\underline{F}}}_i &= \sum_{j=1}^{N} \left(\underline{v}_j - \underline{v}_i \right) \otimes \nabla_0 \overset{\triangle}{W}_{ij} \frac{m_j}{\varrho_j^0} \\
\underline{\underline{F}}_i &= \sum_{j=1}^{N} \left(\underline{u}_j - \underline{u}_i \right) \otimes \nabla_0 \overset{\triangle}{W}_{ij} \frac{m_j}{\varrho_j^0} + \underline{\underline{I}} \\
\underline{\underline{L}}_i &= \dot{\underline{\underline{F}}}_i \cdot \underline{\underline{F}}_i^{-1} \\
\underline{\underline{\sigma}}_i &= \underline{\underline{S}}_i - \underline{\underline{I}} \cdot p_i \\
\underline{\underline{P}}_i &= J_i \cdot \underline{\underline{\sigma}}_i \cdot \underline{\underline{F}}_i^{-1} \\
J_i &= det(\underline{\underline{F}}_i)
\end{aligned} \tag{5.86}$$

The default artificial constants are the same as in 5.2.4, however, in some situations both α and β need to be increased quite a bit.

5.2.5 *Algorithm 5*

Semi-discrete ODEs:

$$\underline{u}_i = \underline{X}_i - \underline{x}_i \tag{5.87}$$

$$\underline{\dot{v}}_{ij} = \left(\frac{\underline{\underline{\hat{P}}}_i}{\varrho_i^2} + \frac{\underline{\underline{\hat{P}}}_j}{\varrho_j^2} + \Pi_{ij} \cdot \underline{\underline{I}} \right) \cdot \nabla_0 \overset{\triangle}{W}_{ij} m_i$$

$$\dot{\underline{v}}_i = 1/2 \sum_{j=1}^{N} \left(-\underline{\underline{R}}_i \cdot \underline{\dot{v}}_{ji} + \underline{\underline{R}}_j \cdot \underline{\dot{v}}_{ji} \right) + \underline{b}_i \tag{5.88}$$

Note that the continuity equation is absent in (5.88). This is because the stress tensor was not split into deviatoric and hydrostatic part, and the density is only needed when using

equation of state (3.70). Of course, ϱ can stil be updated like in (5.85) if it is needed, e.g. for visualization.

Shape matching procedure to find $\underline{\underline{R}}$:

$$\begin{aligned}\underline{\underline{\mathcal{A}}}_i &= \sum_{j=1}^{N} W_{ij} \cdot \left(\underline{x}_j - \underline{x}_i\right) \otimes \left(\underline{X}_j - \underline{X}_i\right) m_j \\ \underline{\underline{\mathcal{A}}}_i &= \underline{\underline{R}}_i \cdot \underline{\underline{U}}_i \end{aligned} \tag{5.89}$$

where $\underline{\underline{\mathcal{A}}} = \underline{\underline{R}} \cdot \underline{\underline{U}}$ is the polar decomposition. The material model is co-rotated, $(\hat{\cdot})$ denotes a co-rotated quantity:

$$\begin{aligned}\hat{\underline{u}}_{ij} &= \underline{\underline{R}}_i^T \left(\underline{x}_j - \underline{x}_i\right) - \left(\underline{X}_j - \underline{X}_i\right) \\ \hat{\underline{\underline{F}}}_i &= \sum_{j=1}^{N} \hat{\underline{u}}_{ij} \nabla_0 W_{ij} \frac{m_j}{\varrho_j^0} + \underline{\underline{I}} \\ \hat{\underline{\underline{E}}}_i^{\text{lin}} &= \frac{1}{2}\left(\hat{\underline{\underline{F}}}_i^T + \hat{\underline{\underline{F}}}_i\right) + \underline{\underline{I}} \\ \hat{\underline{\underline{\sigma}}}_i &= 2G \cdot \hat{\underline{\underline{E}}}_i^{\text{lin}} + \lambda \cdot tr(\hat{\underline{\underline{E}}}_i^{\text{lin}}) \\ \hat{\underline{\underline{P}}}_i &= J_i \cdot \underline{\underline{\sigma}}_i \cdot \underline{\underline{F}}_i^{-1} \end{aligned} \tag{5.90}$$

5.2.6 *Algorithm 6*

Semi discrete ODEs:

$$\dot{\underline{v}}_i = -\,2 \sum_{j=1}^{N} \underline{v}_i \cdot \underline{\underline{F}}_i \cdot \underline{\underline{\sigma}}_i \cdot \nabla \overset{\triangle}{W}_{ij} \tag{5.91}$$

Material Model:

$$\begin{aligned}\underline{\underline{F}}_i &= \sum_{j=1}^{N} \left(\underline{u}_j - \underline{u}_i\right) \otimes \nabla_0 \overset{\triangle}{W}_{ij} \frac{m_j}{\varrho_j^0} + \underline{\underline{I}} \\ \underline{\underline{E}}_i &= \frac{1}{2}\left(\underline{\underline{F}}_i^T \cdot \underline{\underline{F}}_i - \underline{\underline{I}}\right) \end{aligned} \tag{5.92}$$

The same remarks as in 5.2.5 with regard to density ϱ apply.

5.2.7 *Algorithm 7*

Algoithms 7 and 8 are a bit more involved in the sense that there are two sets of discretization points; the particles and the quadrature points. Again, quadrature points do not "carry" their own kernel functions, i.e. W_{ij} exists if $i \in N$ but does not exist if $i \in M$, where N is the set of particles and M the set of quadrature points. Respecting correct order of the

indices is thus of utmost importance for a correct implementation. This also means that the variables are split; the velocity is located on the particles, density and stress is located on the quadrature points. This is the same split as suggested in stress points algorithms, [268].

For all quadrature points q do:

$$\underline{\underline{L}}_q = \sum_{i=1}^{N} \underline{v}_i \otimes \nabla \overset{\circ}{W}_{iq} \frac{m_i}{\varrho_i} \tag{5.93}$$

Carry out material model (5.76) to obtain $\underline{\underline{\sigma}}_q$. Then:

$$\dot{\underline{v}}_i \leftarrow \dot{\underline{v}}_i - \underline{\underline{\sigma}}_q \cdot \nabla \overset{\circ}{W}_{iq} \frac{m_q}{\varrho_q \cdot m_i} \tag{5.94}$$

which is a scatter operation, as opposed to the gather operation usually encountered in particle methods in general and SPH in particular. Thus, in addition to (5.94), the semi-discrete ODEs for particles are:

$$\begin{aligned} \dot{\underline{x}}_i &= \underline{v}_i \\ \dot{\varrho}_i &= \varrho_i \sum_{j=1}^{N} \left(\underline{v}_j - \underline{v}_i \right) \cdot \nabla \overset{\circ}{W}_{ij} \frac{m_j}{\varrho_j} \end{aligned} \tag{5.95}$$

And for the quadrature points:

$$\begin{aligned} \dot{\underline{x}} &= \underline{v}_q^{\text{Interp}} \\ \dot{\varrho}_q &= \varrho_q \sum_{i=1}^{N} \underline{v}_i \cdot \nabla \overset{\circ}{W}_{iq} \frac{m_i}{\varrho_i} \end{aligned} \tag{5.96}$$

The interpolated velocity is computed as:

$$\underline{v}_q^{\text{Interp}} = \sum_{i=1}^{N} \underline{v}_i \overset{\circ}{W}_{iq} \frac{m_i}{\varrho_i} \tag{5.97}$$

Note that the density needs to be integrated in time on both the particles and the quadrature points to maintain correct integration weights m_i/ϱ_i, m_q/ϱ_q, respectively.

Another issue not discussed in sections 5.1.7, 5.1.8 is that in the equation system:

$$\underline{\underline{M}}\, \dot{\underline{v}} + \underline{f}_{\text{Int}} = 0 \tag{5.98}$$

The basis functions in $\underline{\underline{M}}$, $\underline{f}_{\text{Int}}$ do not possess the Kronecker delta property. Hence, one solves in fact for virtual accelerations $\dot{\underline{v}}^{\text{Virt}}$. To retrieve the actual accelerations an additional re-approximation has to be performed, as noted by Belytschko [22] when introducing the EFG, and also explicitly stated in [127] and [181]. Interestingly enough, a lot of implementations seem to omit this step, e.g. [283]. The re-approximation procedure and correct equation system read:

$$0 = \underline{\underline{M}}\, \dot{\underline{v}}^{\text{Virt}} + \underline{f}_{\text{Int}} \tag{5.99}$$

$$\dot{\underline{v}}_i = \sum_{j=1}^{N} \dot{\underline{v}}_j^{\text{Virt}} \overset{\circ}{W}_{ij} \frac{m_j}{\varrho_j} \tag{5.100}$$

which is strikingly similar to the smoothing procedures often used as a stabilization measure, see 4.7.3, especially the smoothing scheme given in [193] [192]. It was discovered that (5.100) can indeed double as a stabilization measure by replacing $\overset{\circ}{W}$ with $\overset{\triangle}{W}$. This is exploited in the implementation at hand.

5.2.8 *Algorithm 8*

In preprocessing the particles are meshed using a quad mesh. The number of Gauss points per element is set. n_{SD}^{q} Gauss points are seeded per element, where n_{SD} is the number of spatial dimensions and q the degree of Gauss quadrature used. Gauss quadrature weights w_q are assigned.

For all quadrature points q do:

$$\underline{\underline{F}}_q = \sum_{i=1}^{N} \underline{u}_i \otimes \nabla_0 \overset{\circ}{W}_{iq} \frac{m_i}{\varrho_i}$$

$$\dot{\underline{\underline{F}}}_q = \sum_{i=1}^{N} \underline{v}_i \otimes \nabla_0 \overset{\circ}{W}_{iq} \frac{m_i}{\varrho_i^0} \tag{5.101}$$

$$\tag{5.102}$$

Carry out material model (5.86) to obtain $\underline{\underline{P}}_q$. Then perform scatter operation:

$$\dot{\underline{v}}_i \leftarrow \dot{\underline{v}}_i - \underline{\underline{P}}_q \cdot \nabla_0 \overset{\circ}{W}_{iq} \frac{w_q}{m_i} \tag{5.103}$$

Semi-discrete ODEs on particles in addition to (5.103):

$$\dot{\underline{x}}_i = \underline{v}_i \tag{5.104}$$

$$\tag{5.105}$$

Semi-discrete ODEs on quadrature points:

$$\dot{\varrho}_q = \varrho_q tr(\underline{\underline{L}}_q) \tag{5.106}$$

where $\underline{\underline{L}}_q$ was computed in course of material model (5.86). Virtual accelerations are turned in to actual accelerations by:

$$\dot{\underline{v}}_i = \sum_{j=1}^{N} \dot{\underline{v}}_j^{\mathrm{Virt}} \overset{\circ}{W}_{ij} \frac{m_j}{\varrho_j^0} \tag{5.107}$$

5.3 RADIAL RETURN: ROOT FINDING ALGORITHMS

The mathematical groundwork for the radial return algorithm was already presented in section 3.4.3. To quickly restate, the root of the nonlinear equation

$$0 = |\underline{\underline{S}}^{\mathrm{trial}}| - \sqrt{2/3} \cdot \sigma_y(\bar{\varepsilon}_{\mathrm{pl}}, \dot{\bar{\varepsilon}}_{\mathrm{pl}}, T) - \sqrt{3/2} \cdot 2G \cdot \Delta\lambda_{\mathrm{pl}} = \mathcal{G}(\Delta\lambda_{\mathrm{pl}}) \tag{5.108}$$

needs to be found, where $\sigma_y(\bar{\varepsilon}_{\text{pl}}, \dot{\bar{\varepsilon}}_{\text{pl}}, T)$ is a function of $\Delta\lambda_{\text{pl}}$ since $\bar{\varepsilon}_{\text{pl}}, \dot{\bar{\varepsilon}}_{\text{pl}}$ are functions of $\Delta\lambda_{\text{pl}}$. In this work

$$\sigma_y(\bar{\varepsilon}_{\text{pl}}, \dot{\bar{\varepsilon}}_{\text{pl}}, T) = \left(A + B(\bar{\varepsilon}_{\text{pl}})^n\right) \cdot \left(1 + C \ln\left(\frac{\dot{\bar{\varepsilon}}_{\text{pl}}}{\dot{\bar{\varepsilon}}^0_{\text{pl}}}\right)\right) \cdot \left(1 - \left(\frac{T - T_0}{T_m - T_0}\right)^m\right) \tag{5.109}$$

i.e. the Johnson-Cook flow stress model. Consider section 3.4.3 for definition of the quantities at hand and further explanations.

There are countless numerical root finding methods, and, since the maximum λ_{pl} encountered is usually in the order of $\sim 3 \cdot 10^{-5}$ while G is in the order of 10^{11} for metals, this seems like a non-trivial situation. In this section four root finding methods were evaluated with regard to their runtime and accuracy:

- Newton's method, also known as the Newton-Rhapson method. This is probably the simplest and most well known numerical root finding method. The iterative procedure is simply:

$$\Delta\lambda^{K+1}_{\text{pl}} = \Delta\lambda^{K}_{\text{pl}} - \frac{\mathcal{G}(\Delta\lambda^{K}_{\text{pl}})}{\frac{d}{d\Delta\lambda^{K}_{\text{pl}}}\mathcal{G}(\Delta\lambda^{K}_{\text{pl}})} \tag{5.110}$$

  The method is not without its problems. While the total derivative $d/(d\Delta\lambda_{\text{pl}})$ is still tractable for (5.109), this may be challenging for more involved material models like for example the Chaboche model [36], see [135] and virtually intractable for even more complex ones. Since these analytical derivatives may become very long, floating point errors may accumulate rather quickly with number of time steps, making a method without use of analytical derivatives but estimated ones a more attractive choice. An initial guess $\lambda^{K}_{\text{pl}}|K = 0$ is needed. The method may not return the root closest to λ^{0}_{pl} due to overshoot or terminate due to a stationary point where

$$\frac{d}{d\Delta\lambda^{K}_{\text{pl}}}\mathcal{G}(\Delta\lambda^{K}_{\text{pl}}) = 0 \tag{5.111}$$

- The difficulty of finding analytical derivatives may be circumvented by the secant method. The secant method approximates the derivatives using a finite difference stencil like approach. This method may even be faster since the amount of floating point operations spent calculating the derivatives of σ_y my be quite large.

- The subject of overshoot may be addressed by using a bracketing algorithm with pre-condition $f(A) < 0$ and $f(B) > 0$ or $f(A) > 0$ and $f(B) < 0$, i.e. by supplying the root finding algorithm with two endpoints A, B which guarantee to straddle a root instead of an initial guess. There is a well known implementation from the "numerical recipes" book [204] extending the Newton method, called `RTSAFE`. This method enhances the original Newton-Rhapson method by a bracketing step; if Newtons method overshoots, i.e. leaves the interval $[A, B]$ the method switches to a bisection step, returning the current iteration into the "safe" interval.

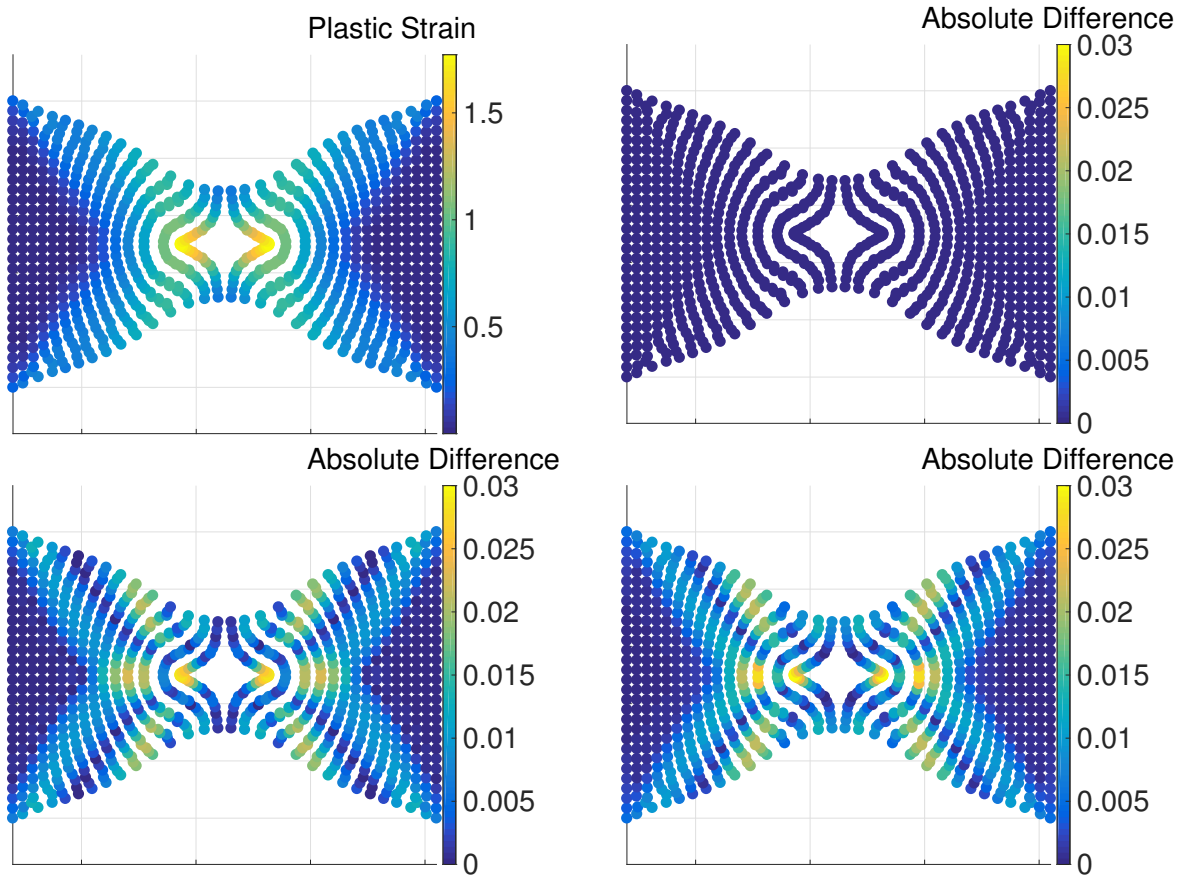

Figure 5.6: Comparing the different root finding methods. The result on the top left was found using Newton iteration and was arbitrarily chosen as reference. Absolute differences are displayed for the secant method, Brent's method and RTSAFE in clockwise order. Differences do not exceed 2%.

- The last method considered is Brent's method [29], a version of which is also implemented into the standard root finding method in MATLAB. This algorithm is focused on safety and robustness, deciding in each iteration between bisection method, inverse quadratic interpolation and the secant method, picking the safest option. Similar to RTSAFE it is a bracketed algrotihm.

These four algorithm where implemented into the total Lagrangian meshfree method by Reveles [217] / 5.1.4 and compared. Table 5.1 lists the runtimes, figure 5.6 shows the differences. Since the differences observed seem negligible, the fastest option, i.e. the Secant method was chosen for all computations in this thesis.

While these are timings represent quite typical results, it was generally observed that the particular timings depend on the workload quite a bit. Occasionally Newtons method was observed to be slightly faster than the Secant method. RTSAFE seems to be least runtime efficient consitently, but is in some cases far closer to Brents method than in this cae.

Method	Runtime [s]
Newton	122.64
Secant	110.45
RTSAFE	511.733
Brent	155.39

Table 5.1: Runtimes to simulate the tensile tests in 5.6 if the various root finding methods are employed for the radial return.

5.4 CONTACT ALGORITHM

The issue of contact can be divided into two stages, at least if penalty contact, as is the case for this work, is used.

1. Identification of contact pairs
2. Computation of contact forces

The first issue is mostly an algorithmic one, whereas the second one is more on the modeling side of things. Development of a general purpose contact algorithm is beyond the scope of this work, instead, quite the opposite route was taken and the contact algorithm was tailor made for the problem of metal cutting.

In general, the issue of contact in smoothed particle methods is a quite peculiar one. One hand hand, it is not quite clear where contact should be established, since bodies discretized by smoothed particle methods do not possess a sharp interface, but a smoothed one. On the other hand, as long as no shear or tensile stresses develop across two bodies, that is the contact is mostly a collision, updated Lagrangian particle methods handle contact naturally by particle interaction. However, this is not a reliable approach, especially in sliding motion, which is present in metal cutting. Consider figure 5.7 for an illustration. In the following, contact search will be discussed in 2D, then 3D, since the approaches are quite different. Afterwards, an approach dealing with smooth interfaces is addressed. Finally, the visibility criterion which is mostly relevant in total Lagrangian simulations is considered. The discussion will close by elaborating the implementation of Coulomb friction.

5.4.1 *Preliminaries - Hashing and Hash Maps*

In order to understand the following sections it is necessary to understand the concept of hash functions and hash maps:

- A hash function is a function that is able to map data of arbitrary size to data of fixed size. Applying a hash function to some input is called “hashing”.
- A hash map or hash table is the resulting data structure if the keys are stored at their resulting hash locations. This enables quick lookups to check if a given key is already

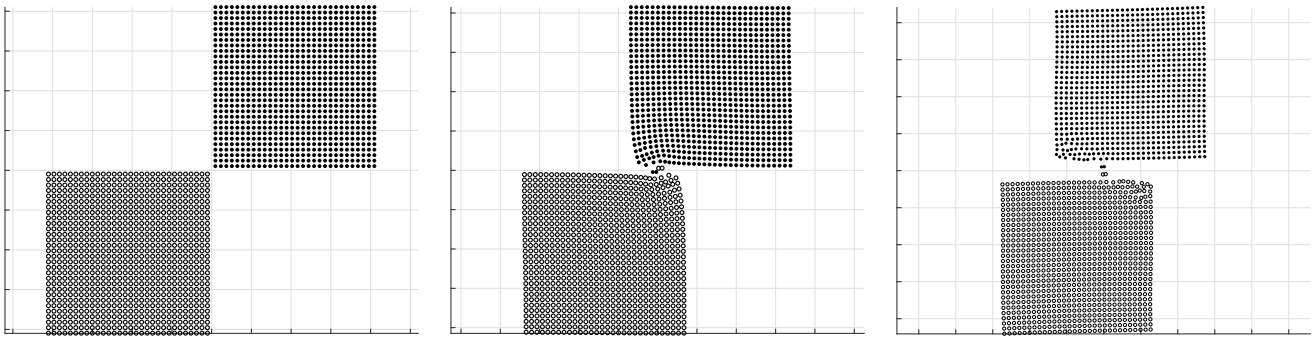

Figure 5.7: Two rubber squares move with an initial velocity of 100 m/s against each other. A small vertical air gap divides the squares, thus the squares are not expected to interact. However, due to the nature of the SPH interaction shear stresses develop across the air gap. The stresses developed are so intense that some particles are ripped from the squares. Shading simply indicates which square the particles belong to, subsequent time steps displayed from left to right.

present in the hash map. If two distinct elements happen to share the same result of the hash function this is called a "hash collision" or "hit". Hits are bound to happen sooner or later since to domain of the hash function is fixed, while its input is arbitrarily large.

This concpet is illustrated in figures 5.8 and 5.9. It is not immediately clear how these concepts relate to contact search. While this is explained in detail in section 5.4.3, suffice to say for now that there is such a thing as *spatial* hashing, where the hash function depends on geometric properties of the input, e.g. their location in space, and the input located close spatially is mapped to the same hash value.

5.4.2 *Contact Search - 2 Dimensions*

For 2 dimensions, a parametric approach was taken. That is, the tool is represented by distinct geometric entities of which the distance function can be computed analytically. The tool was restricted to be representable by 4 line segments and a circle segment, the latter representing the cutting edge radius. That is, the chip breaker geometry, if any, is neglected. The steps of the algorithm can be summarized as follows:

For each particle p at $\underline{x}_p$ within the bounding box $\mathcal{B}$ of the tool, find closest point $\underline{p}_s$ on each tool segment $s \in S$. Compute distance vector $\underline{d}_s = \underline{p}_s - \underline{x}_p$. If

$$\underline{d}_s \cdot \underline{n}_s > 0 \quad | \quad \forall s \in S \tag{5.112}$$

where $\underline{n}_s$ is the surface normal of segment s at $\underline{p}_s$, the particle is in contact with the tool and the penetration depth is reported as

$$min(|\underline{d}_s|) \quad | \quad \forall s \in S \tag{5.113}$$

A sketch illustrating the algorithm is given in figure 5.10. The approach presented enjoys two main benefits over a more classical approach, where the tool is comprised of FEM shell

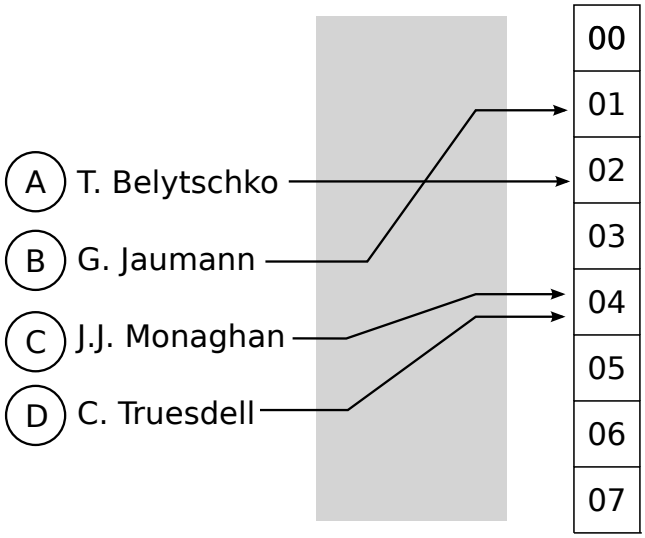

Figure 5.8: Four names, labelled A-D are mapped onto the integers 0-7.

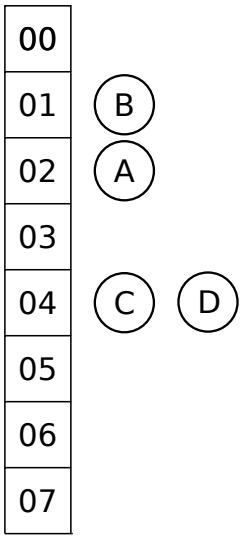

Figure 5.9: Resulting hash map from figure 5.8

elements, say: it is fast in the sense that only a small number of primitives needs to be assessed and it is accurate in the sense that the cutting edge radius is a circle exactly, as opposed to a piecewise linear approximation to a circle segment as would be customary using the more classical approach. Downsides include that the tool can only be modeled rigid and can't be of arbitrary shape, e.g. no chip breaker can currently be included as mentioned before.

5.4.3 *Contact Search - 3 Dimensions*

Modelling the tool fully parametric in 3D for all possible situations is quite challenging, albeit it might be possible by successive boolean operations on simple parametric primitives. However, for this thesis another approach was taken by modeling the tool with finite elements. Establishing contact between a general, unstructured FE mesh and particles is a challenge in itself. Generally algorithms designed to this end perform two sweeps; a candidate set of contact pairs is identified by spatial binning in a first sweep, called the broad phase, then the final contact pair is decided upon by an array of heuristics, called the close phase. An overview about how such an algorithm could be designed is found in [229].

There are other options to consider if some assumptions are made with regard to the FEM mesh. Since a general purpose solver was never the intention of this work, it was decided to restrict the contact algorithm to be able to establish contact between a triangulated surface mesh and SPH particles only. For this case, a hierarchical query in the shape of an Axis Aligned Bounding Box AABB tree is suitable, see for example [26]. An efficient implementation is found in [203], which was used as a reference.

While these algorithms are efficient on CPUs, in anticipation of what will be the topic in chapter 6, another approach fully dependent on spatial hashing will be taken, similar to

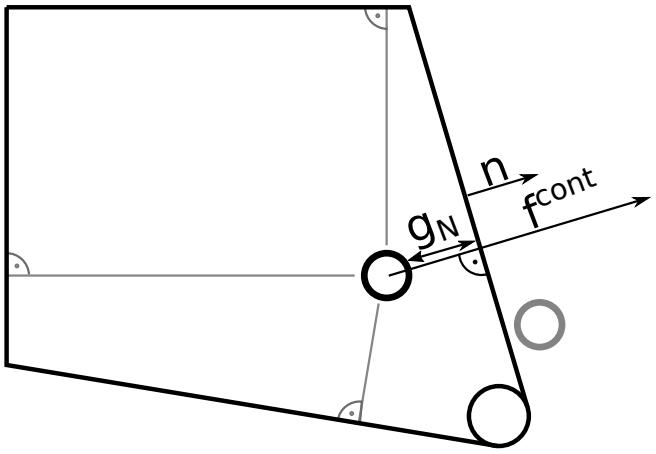

Figure 5.10: Illustration regarding the 2 dimensional contact search. A particle at old location indicated in gray penetrates the tool, resting at the position indicated in black. The distance to each segment is measured. Penetration depth g_N is recorded as the distance to the closest segment. A penalty contact algorithm would then apply a contact force parallel to the surface normal of the closest segment. Note that the sketch is not to scale, the amount of penetration displayed is greatly exaggerated for illustrative purposes.

the approach given in [260]. In 2D, the algorithm can be summarized as follows: spatially hash both triangles and particles into a hash map. This is explained in figure 5.11. Note that the hash map needs to be computed only once in the beginning, since the tool is rigid. A triangle may belong to multiple boxes and each box may contain multiple triangles. For each p at $\underline{x}_p$ within bounding box $\mathcal{B}$ of tool find hashes ix, iy. Check box $\mathcal{B}$ with index $hash(ix, iy)$. Distinguish the following two cases, which are illustrated in figures 5.12 and 5.13.

- $\mathcal{B}$ is not empty: compute closest point $\underline{p}_t$ on triangles $t \in \mathcal{B}$, as well as in neighboring boxes with $hash(ix \pm 1, iy \pm 1)$. Select

$$\underline{p}_s^{\min} \quad | \quad min(|\underline{p}_s - \underline{x}_p|) \tag{5.114}$$

  If

$$(\underline{x}_p - \underline{p}_s^{\min}) \cdot \underline{n}_t < 0 \quad | \quad \forall s \in S \tag{5.115}$$

  where $\underline{n}_t$ is the triangle surface normal of the triangle containing $\underline{p}_s^{\min}$.

- $\mathcal{B}$ is empty: Check boxes with $hash(ix \pm c, iy \pm d)$ for $c, d = 1, 2, 3, \ldots$ until a non empty box is found. Set $ix \leftarrow ix \pm c$, $jx \leftarrow jx \pm d$. Restart at previous case. Do this for all non empty boxes encountered.

The discussion was kept in 2D, but the algorithm extends naturally to 3D. There are some implementation details to be elaborated upon. One is the size of the boxes in the spatial hashing algorithm. There is no clear answer to this case, small boxes may lead to a lot of empty boxes, hence to a lot of iterations until a full box is found. Large boxes may lead to a high number of triangles per box, necessitating a lot of closest point on triangle computations. In principle a benchmark would need to be performed searching for the optimal box size for

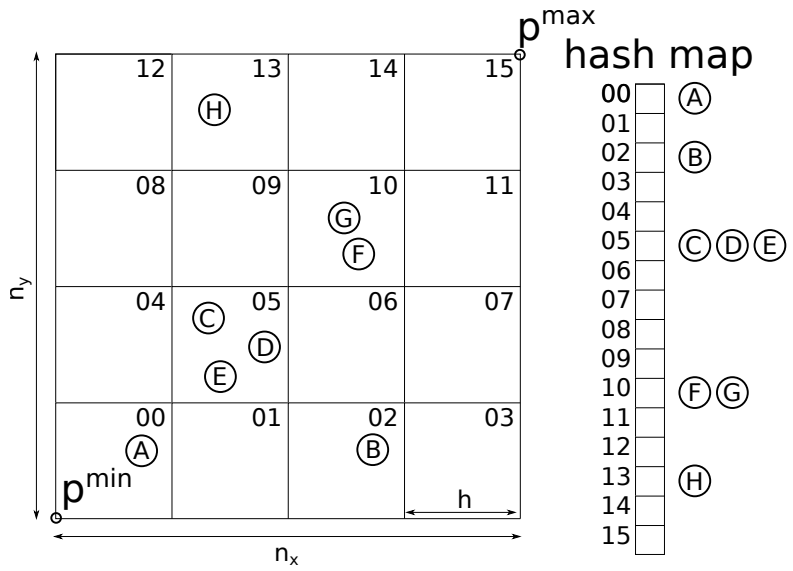

Figure 5.11: Spatial hashing illustrated: The spatial domain is discretized by some boxes of potentially arbitrary size h. The boxes are numbered by some scheme, e.g. successively by column and row. Objects A-E are then hashed into these boxes, resulting into a hash due to the numbering scheme of the boxes, in this case $hash(\underline{p}) = \left\lfloor (p_y - p_y^{\min})/h \right\rfloor \cdot n_x + \lfloor (p_x - p_x^{\min})/h \rfloor$. More sophisticated numbering schemes leading to better space locality exist, see [174] or [109] and the explanations in section 6

each mesh considered. As a starting point it was found that the average edge length of all triangles in the mesh is reasonable. This choice would be near optimal if the size of triangles contained in the mesh does vary only little.

Another crucial detail is the computation of correct triangle normals in 3D. This is relevant if the closest point is not in the interior of the triangle ABC, where the normal is uniquely defined by cross product $\underline{n}_t = \underline{AB} \times \underline{BC}$, but on an edge or corner. In this case, the normals at the edges and corners need to be computed by the angle averaging procedure outlined in [261]. Even then simply checking the if the normal points away from the distance vector to the closest point (5.112), (5.115) may give the wrong answer if the query point is far away from the triangle mesh. This may be a problem if the triangle mesh only occupies a small portion of its AABB. In this case a simple, but safe ray casting procedure, needs to be run before the closest point search:

1. From the query point, shoot a ray in a random direction
2. Count the number of intersections with the triangle mesh
3. If the number of intersections is zero or even, the point is inside the mesh and outside otherwise.

This procedure is illustrated in figure 5.14.

A benchmark was computed to measure performance and ensure correctness of the spatial hashing algorithm. `libgts` [203] was used as a reference. A realistic work load was ensured

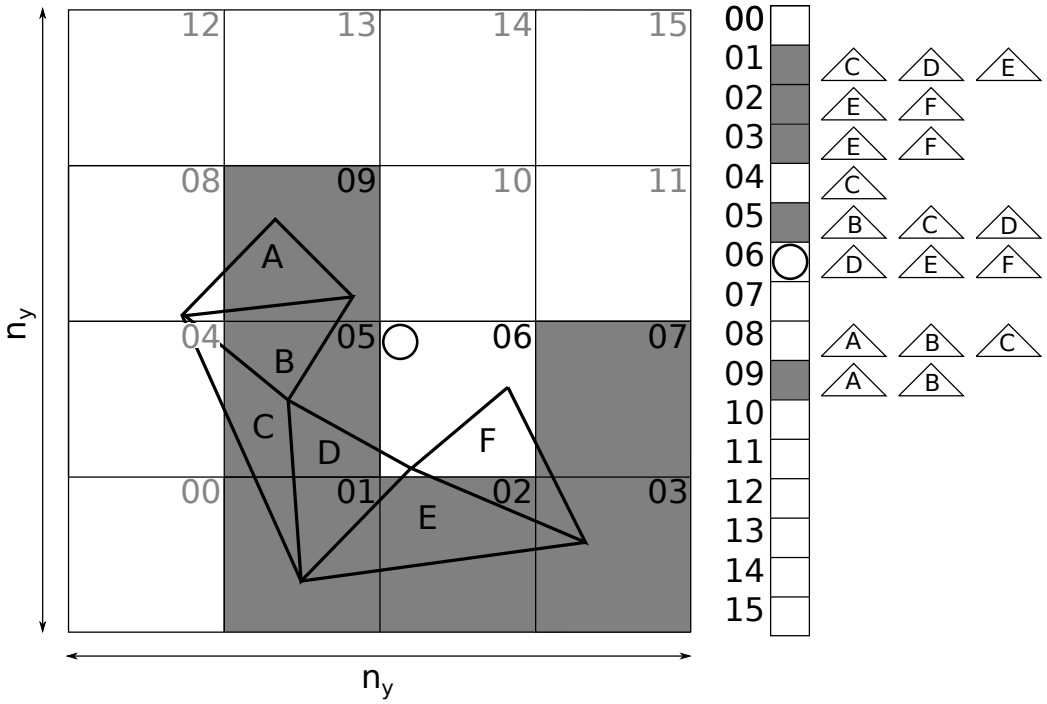

Figure 5.12: Spatial hashing algorithm for contact queries. This case illustrates why it is necessary to search the non-empty neighboring boxes indicated in gray, in addition to the box actually hit. The query particle indicated as a circle resides in box 6, where triangle F is closest. However, triangle B resting in neighboring box 5 is closer.

by seeding particles close to triangle surfaces, representing the material. A drill tool as displayed in figure 5.15 was tested, which is the most complex work load envisonaged for the solver at hand. For this model and workload, a simple check with the surface normal yielded identical results as `libgts`, which uses the safe procedure described. Some select timings are given in tables 5.2, 5.3. "Spatial Hashing (Safe)" refers to the case were the ray casting procedure was performed, "Spatial Hashing" means that the surface normal was used for the space partition query.

Method	Runtime [s]
Linear	27.1
`libgts`	0.6
Spatial Hashing	0.4

Table 5.2: Contact search run times, load case 1: 90'000 particles were seeded on a thin layer around the drill model. "Linear" refers to simply checking all triangles against all particles. Since this is unsigned distance, it does not matter whether the safe ray casting procedure is used.

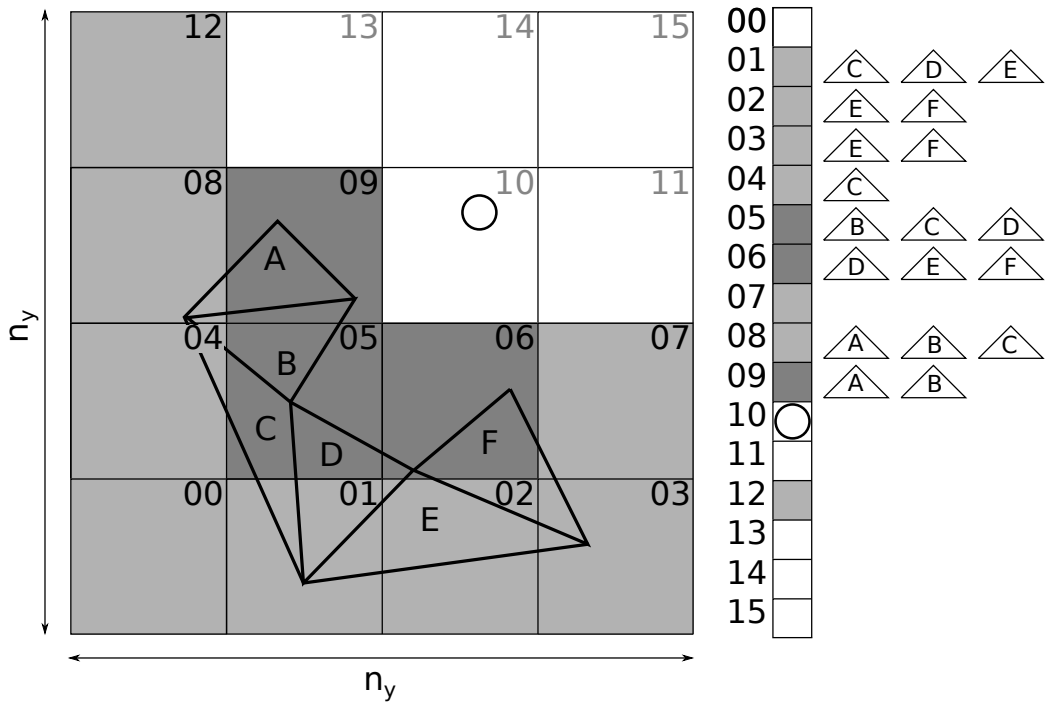

Figure 5.13: Spatial hashing algorithm for contact queries. This time the empy box with number 10 was hit. The neighborhood is scanned for non-empty boxes. These boxes, i.e. number 9, 5, 6 become new center points for the same search procedure as in figure 5.12. This constitutes a worst case situation in this toy example since all non-empty boxes had to be considered.

.

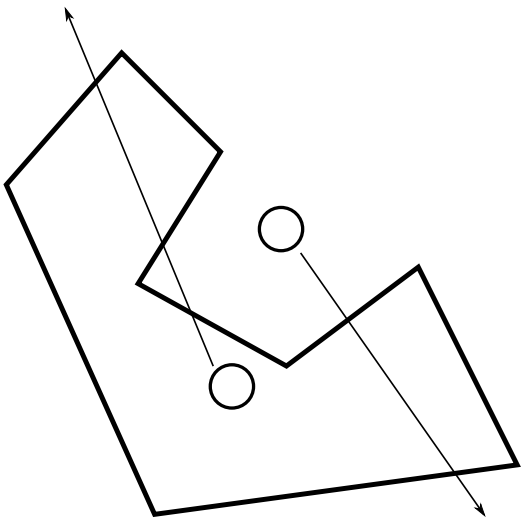

Figure 5.14: Illustration of the safe inside/outside query. A ray is shot from two particles, in a random direction. The particle outside crosses the boundary an even number of times, the particle inside an odd number of times.

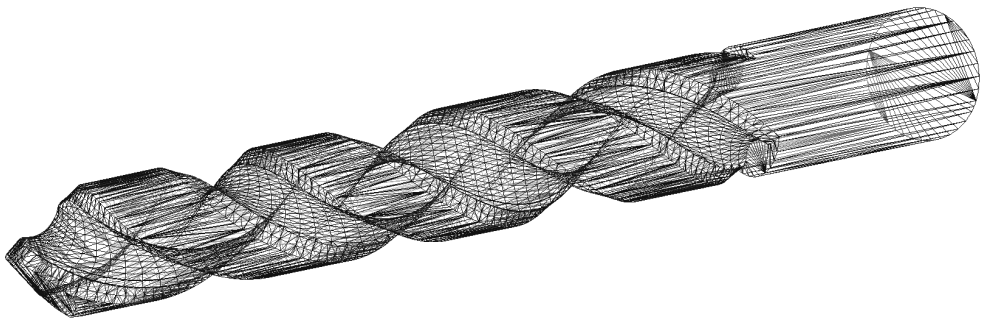

Figure 5.15: The drill model used for testing and timing the contact search in 3D. The model features a wide range of triangle sizes, some with very bad aspect ratio. Hence, arguably a very bad mesh, but at the same time a very challenging case for the contact search to handle.

Method	Runtime [s]
Linear	270.4
`libgts`	8.2
Spatial Hashing	4.6
Spatial Hashing (Safe)	5.2

Table 5.3: Contact search run times, load case 1: 900'000 particles were seeded on a thin layer around the drill model, note that this is ten times as much as in case reported in table 5.2. Signed distance to model. "Linear" refers to simply checking all triangles against all particles.

5.4.4 *The contact force*

In sections 5.4.2 and 5.4.3 it was discussed how to decide whether a particle is in contact with a rigid body, and if so, how far that particle penetrates. It remains to be discussed what is to be done upon contact detection. A popular choice is the penalty contact methodology, where a force is applied proportional to penetration depth in direction from the penetrating particle p to the closest point on the surface, which is equal to surface normal $\underline{n}$. There are various options. One choice used in metal forming applications, for example in [182], is the contact force given as:

$$\underline{f}^{\text{cont}} = \kappa \frac{m_p \cdot g_N}{\Delta t^2} \underline{n} \tag{5.116}$$

Which is simply twice the force needed to move a body with mass m_p over a distance of g_N in Δt seconds. The stiffness of the slave material is incorporated into interface stiffness constant κ. Consequently g_N denotes the penetration depth.

There are other options to chose $\underline{f}^{\text{cont}}$, for example Belytschko associates κ with physical quantities in [139], but since simulations proved to be almost universally stable across numerous length scales using $\kappa = 1$. this procedure was not applied. An entirely different approach is taken by the commercial solver `LSDYNA`, according to user manual [99]:

$$\underline{f}^{\text{cont}} = \kappa \frac{g_N \cdot K \cdot A^2}{V} \underline{n} \tag{5.117}$$

where K is the Bulk Modulus, and A, V, the area and volume of the contacting elements. Bulk modulus and area/volume can be taken from the slave/workpiece or master/tool side, and there might be situations where one is preferable over the other, see [99]. In the implementation at hand, all quantities where taken from the workpiece side, where the volume and area were derived from the particle radius.

Figures 5.16, 5.17 show a simple test simulation and the differences between the two contact forces described. Since the differences are small and the procedure is very simple, (5.116) is chosen for all simulations in this thesis. Friction is discussed in a separate section 5.4.7.

5.4.5 *A smooth interface contact algorithm*

As mentioned, bodies discretized by smoothed particle methods lack a sharp interface. Instead, the interface is blurred out by the smoothing kernel. Thus, the issue of contact between a sharp and a blurred interface may be regarded as an ill-posed one, since it is not at all clear when contact is to be established; the mass center and the zero isosurface of mass are just two of an infinitude of options. An early effort to establish smooth contact between rigid bodies and fluids is due to Monaghan [170], where the general idea to impose a boundary force similar to the Lennard-Jones potential [125] found its inception. This idea

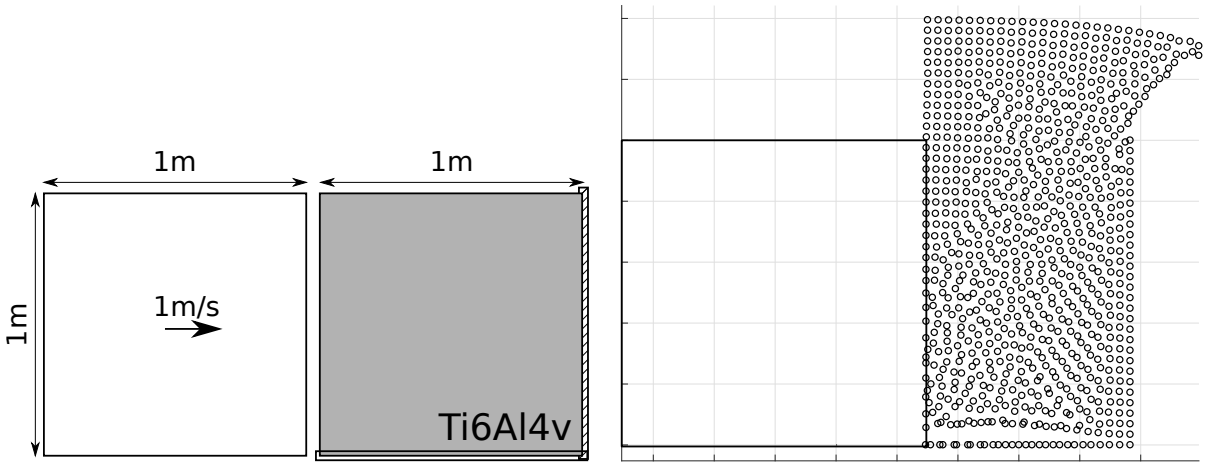

Figure 5.16: Simulation set up for all preliminary contact force and friction force benchmarks on the left: A rigid plate is forced against a deformable one with 1m/s. The deformable plate is made from a titanium alloy. The simulation setup is the one by Monaghan [91] / 5.1.1. The last frame of the simulation, after 0.3 seconds, is shown on the right.

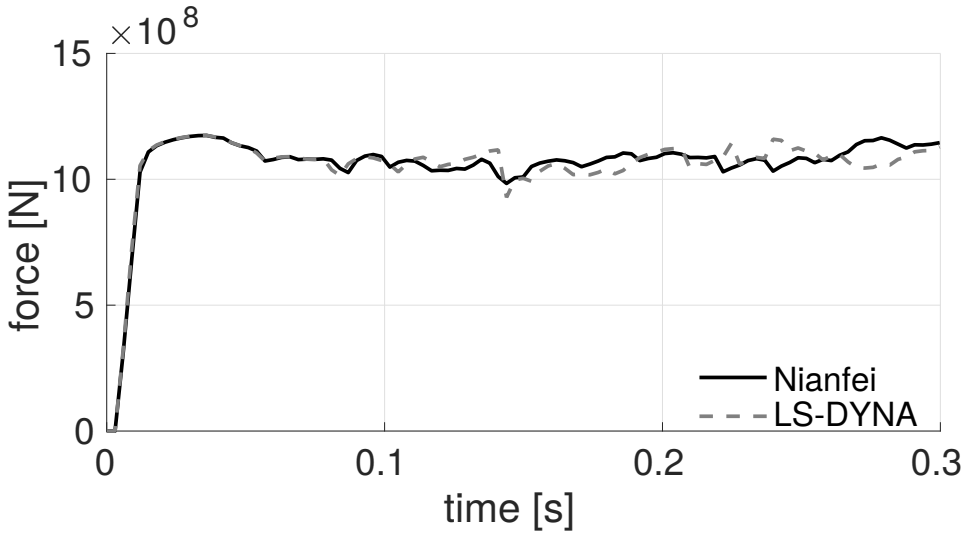

Figure 5.17: Two methods to compute the contact force are compared, i.e. (5.116) and (5.117). It can be seen that both methods yield virtually identical results

.

was further fleshed out by the computer graphics community in [176]. They seed the rigid body/bodies with repulsive particles, carrying a special kernel function:

$$W_h^{\text{cont}} = \begin{cases} \kappa \frac{(h-r)^4 - (h-r_0)^2(h-r)^2}{h^2 r_0 (2h - r_0)} & \text{if} \quad r < h \\ 0 & \text{otherwise} \end{cases} \tag{5.118}$$

where κ is the interface stiffness. While the approaches in [170] and [176] are similar, [176] was chosen because it was proven useful for metal cutting simulations in use with the meshless solver `pasimodo` [79] in [246] and [244]. It can be seen that the kernel (5.118) resembles the Lennard-Jones potential in the sense that it is repulsive up to distance r_0, then turns adhesive. This is because the authors of [176] intend their algorithm for fluid-solid contact, where sticking of fluid particles to solids may occur. Since this is not the case for solid-solid contact, r_0 is set to h. The boundary force on some particle i can then be computed as:

$$\underline{f}_i^{\text{cont}} = \sum_{b=1}^{I} (\underline{x}_i - \underline{x}_b) W_h^{\text{cont}}(|\underline{x}_i - \underline{x}_b|) \omega_b \tag{5.119}$$

The authors go on to suggest to model the friction as a viscosity term:

$$\underline{f}_i^{\text{fric}} = \tilde{\mu} \sum_{b=1}^{I} (\underline{v}_i - \underline{v}_b) \nabla^2 W_h^{\text{visc}}(|\underline{x}_i - \underline{x}_b|) \omega_b \tag{5.120}$$

with special purpose kernel

$$\nabla^2 W_h^{\text{visc}}(r) = \begin{cases} \frac{45}{\pi h^6}(h - r) & \text{if} \quad 0 \leq r \leq h \\ 0 & \text{otherwise} \end{cases} \tag{5.121}$$

while this may be reasonable for fluid-solid contact this seems less so for solid-solid interaction, especially since the friction parameter $\tilde{\mu}$ is not based on physical considerations and needs to be fit to the process forces in a sentinel simulation if used. Luckily, their algorithm is fully compatible with the friction force calculations presented in 5.4.7. A schematic overview over the algorithm is given in figure 5.18.

This algorithm leaves a lot of parameters to be set, like the density of the repulsive particles on the tool or how the smoothing length of those particles is to be chosen. Consider section 7.5.1.4 for a systematic examination to this end.

5.4.6 *The Visibility Criterion*

The smooth interface and interaction due to smoothed, spread out kernels also presents interesting consequences if a body, discretized by smoothed particles, is separated by a foreign body. Particles which are already separated by the foreign body may still interact across the body due to the nature of the interaction radius, see figure 5.19.

This problem becomes even more severe in the total Lagrangian frame. Here, particles which are on opposite sides of the material separation point may keep interacting even if one lands

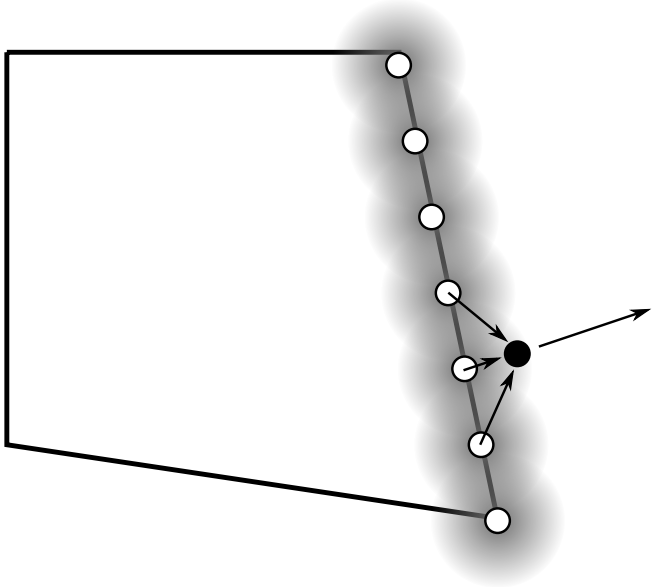

Figure 5.18: An illustration of the smooth contact algorithm described in this section 5.4.5. The contact zone, in this case the rake face of a cutting tool is enhanced with particles featuring a special kind of repulsive kernel. Any particle coming close to these contact particles is pushed back with the intensity of the sum of the repulsive kernels. Consider section 7.5.1.4 for considerations on how to chose the particle spacing and smoothing lengths.

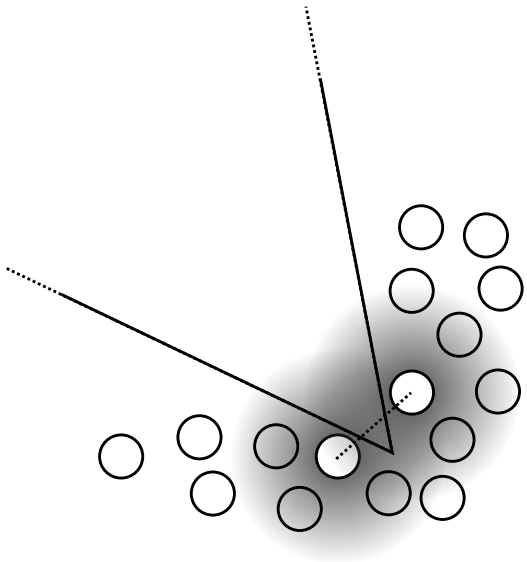

Figure 5.19: A close up of a SPH simulation of a metal cutting operation around the tool tip. The simulation features interactions across the tool tip; the two particles remain in each others support radius even though the tool moved in between them. The issue is more or less pronounced depending on the smoothing length adaption scheme chosen, see section 4.4.

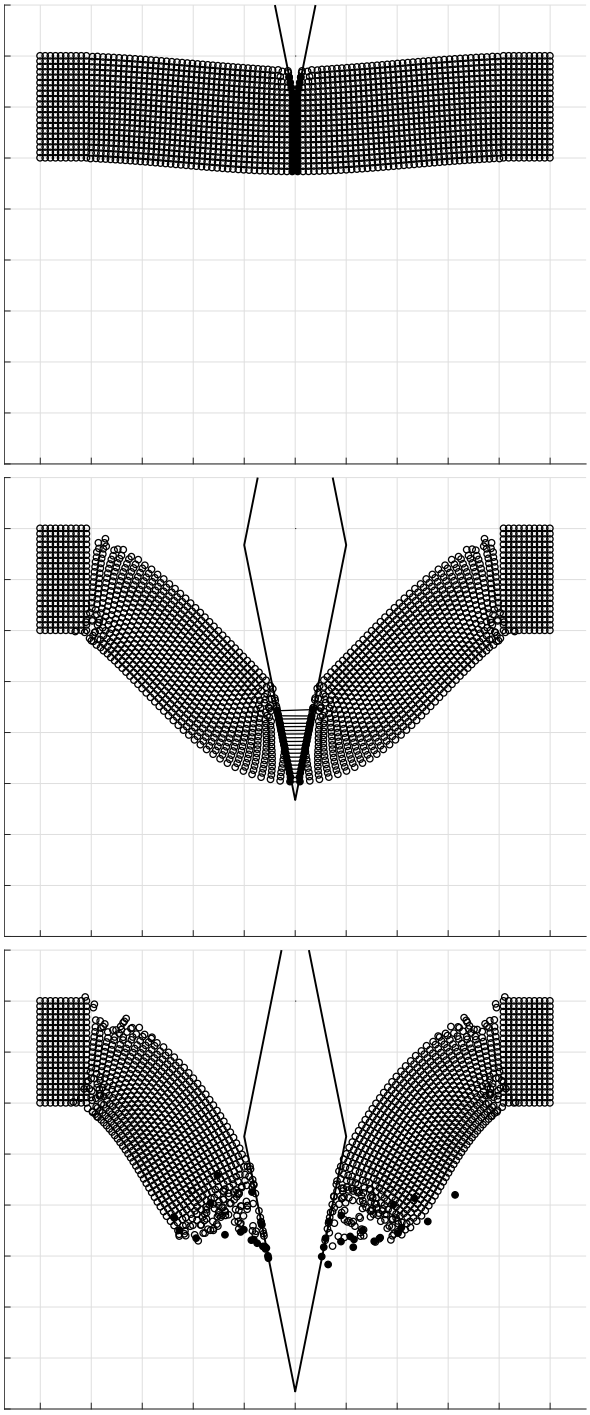

Figure 5.20: A situation roughly comparable to a knife cutting situation is considered. A sharp, rigid, chisel is forced against a slab of a titanium alloy, which is fixed on the sides. The algorithm is total Lagrangian, see [127] / 5.1.8, and no visibility criterion was imposed. Hence, particles keep interacting across the foreign body, see middle frame, leading to eventual failure exhibited in the bottom frame.

up in the chip, the other in the cutting grove. This also means that metal cutting violates the basic assumptions on continuum mechanics, see 3.1, namely that motion $\underline{x} = \phi(\underline{X}, t)$ is sufficiently smooth. Since material separation is modeled into updated Lagrangian particle methods by default, i.e. the particles move out of each others support radius, at least if an upper limit to the support radius is implemented, this is not the case for total Lagrangian particle methods, as the kernel functions are defined on the reference coordinates. This may lead to catastrophic failure of the simulation at hand, see figure 5.20. Consider also section 7.5.2 and illustration 7.44 therein.

This problem can be addressed by introduction of the visibility criterion, i.e. particles i, j only interact if a straight line from $\underline{x}_i$ to $\underline{x}_j$ does not intersect a foreign body. Care needs to be taken if a penalty contact formulation allows spurious interactions of particles i and j currently penetrating the body. Thus, the visibility problem is solved after a projection step out of the body, see figure 5.21 for an elaboration. Note that this procedure has only been

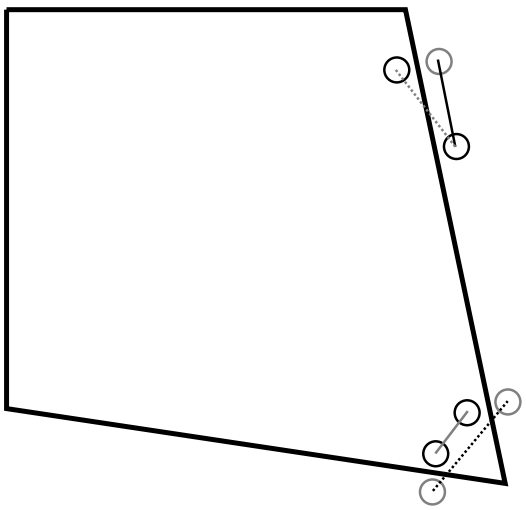

Figure 5.21: Figure highlighting issues of the visibility problem in conjunction with the penalty contact algorithm. The particle pair on top would be spuriously blocked from interacting while the particle pair on the bottom would spuriously be allowed to; gray dashed and solid line, respectively. Hence, the particles inside the tool are first reflected at the closest segment before the visibility test is done, eliminating both false positives and false negatives; black solid and dashed lines, respectively.

.

implemented for the parametrized two dimensional tool. Performing the same routine in 3D, having to check each interaction by shooting a ray through a FEM mesh is prohibitively expensive. In that situation another solution would need to be used, possibly by enhancing the simulation with level set methods. See for example [233] and [1] for explanations on level set methods.

5.4.7 *Friction*

As has been stated in section 3.5, the friction model chosen is the most simple available, i.e. Coulomb friction:

$$|\underline{f}_{\text{fric}}| = \mu |\underline{f}_{\text{cont}}| = f_{\text{clmb}} \tag{5.122}$$

where $|\underline{f}_{\text{cont}}|$ can for example be computed by either the method presented in 5.4.5 or by (5.116), then applied in relative velocity of contacting particle and tool. However, it turns out that such a naive implementation of the friction force in conjunction with the penalty contact algorithm leads to highly oscillatory behavior of the friction force, i.e. the friction force spuriously flips direction with high frequency. Thus, some means to penalize change of direction of the friction force or some kind of stick-slip behavior needs to be implemented. One quantity commonly used is the relative velocity of particle p:

$$\underline{v}_p^{\text{diff}} = \underline{v}_p - \underline{v}^{\text{tool}} \tag{5.123}$$

$$\underline{v}_p^{\text{rel}} = \underline{v}_p^{\text{diff}} - \underline{v}_p^{\text{diff}} \cdot \underline{n} \tag{5.124}$$

where $\underline{n}$ is the surface normal at the contact point. Just as computing the contact force there are are numerous ways of computing the friction force, each containing (5.122) in some shape. One of the most simple approaches is again given by the metal forming community as:

$$\underline{f}_p^{\text{fric}} = -min(f_p^{\text{clmb}}, |\underline{f}_p^T|) \cdot \underline{v}_p^{\text{rel}} / |\underline{v}_p^{\text{rel}}| \tag{5.125}$$

$$\underline{f}_p^T = -\underline{v}_p^{\text{rel}} \cdot m_s / \Delta t \tag{5.126}$$

Another interesting approach is given in [139].

$$\underline{f}_p^* = -\underline{f}_p^{\text{int}} \cdot \underline{t} - \underline{v}_p^{\text{rel}} \cdot m_s / \Delta t \tag{5.127}$$

where $\underline{t} \perp \underline{n}$ and $-f_p^{\text{int}}$ is the internal force at p in the sense of (5.68), i.e. the force resulting due to the internal stresses of the material, or put differently, the force imposed by the momentum equation (3.39) / (3.42). Then:

$$\underline{f}_p^{\text{fric}} = \begin{cases} f_p^{\text{clmb}} \cdot \underline{f}_p^* / |\underline{f}_p^*| \cdot \underline{t} & \text{if} \quad |\underline{f}_p^*| > f_p^{\text{clmb}} \\ |\underline{f}_p^*| \cdot \underline{t} & \text{if} \quad |\underline{f}_p^*| \leq f_p^{\text{clmb}} \end{cases} \tag{5.128}$$

Finally, yet another procedure is given in the LSDYNA manual [99]:

$$\underline{f}_p^* = \underline{f}_p^{\text{fric,old}} - \underline{v}_p^{\text{rel}} \cdot m_p / \Delta t \tag{5.129}$$

where $\underline{f}_p^{\text{fric,old}}$ is the friction computed in the previous time step. Then, similar to [139]:

$$\underline{f}_p^{\text{fric}} = \begin{cases} f_p^{\text{clmb}} \cdot \underline{f}_p^* / |\underline{f}_p^*| & \text{if} \quad |\underline{f}_p^*| > f_p^{\text{clmb}} \\ \underline{f}_p^* & \text{if} \quad |\underline{f}_p^*| \leq f_p^{\text{clmb}} \end{cases} \tag{5.130}$$

Note that f^* is a scalar quantity in the approach by [139] and a vector quantity in [99].

Unfortunately, not a lot of explanation or physical reasoning why these procedures might be adequate ones is given in the original works. Intuitively, the first approach presented implements some kind of stick and slip behavior, where the latter two penalize a switch in direction of the friction force either by comparing it to the current internal force or the friction force obtained by the last time step. All of these methods seem to be useful heuristics rather than descriptions of the physical phenomena at hand. In figure 5.22, the three, respectively four methods including the naive implementation of (5.122), are compared in the same simple benchmark simulation as in 5.16. It is noted that the procedure by [182] leads to highly oscillatory behavior, where the other procedures lead to a stable solution. Since the procedure [99] was proven to be suitable for a commercial FEM package, it was chosen for this work as well.

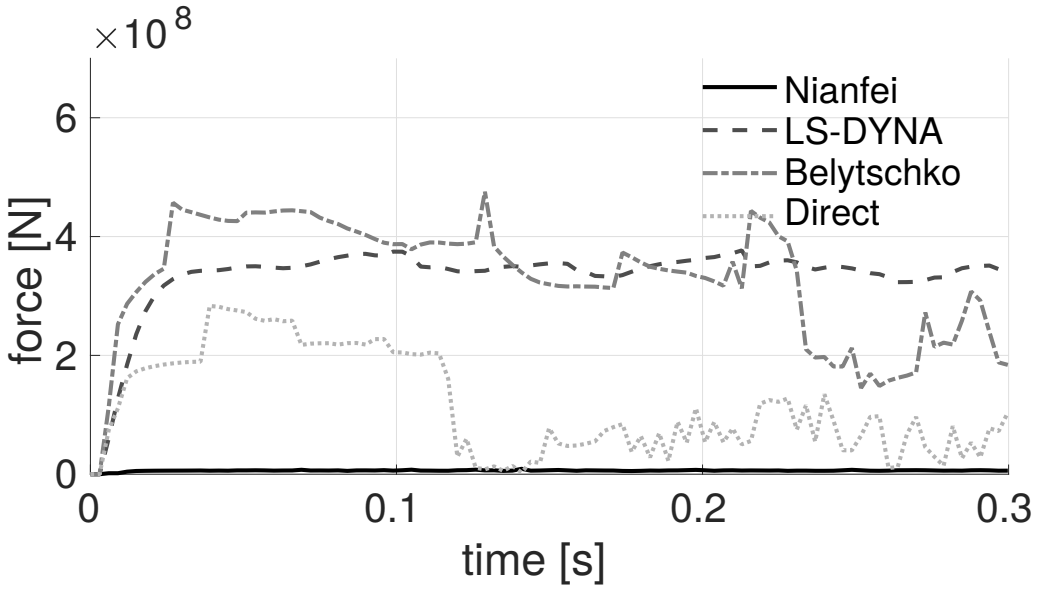

Figure 5.22: Four methods to compute the friction force are compared using the setup given in 5.16. The methods given in the LS-DYNA user manual (5.130) and by Belytschko (5.128) yield comparable results up until the end where the simulation using Belytschko's approach becomes unstable. The approach by Nianfei (5.125), (5.126) spuriously produces zero friction in the head-on collision investigated. The "direct" approach simply uses (5.122) directly. It is highly oscillatory, as stated previously.

.

6

GPGPU COMPUTING

So far the equations arising due to the physical background and how to tackle these equations using meshless techniques have been discussed, as well as incorporation of these meshless techniques into specific algorithms to implemented on a computer. The aim of this chapter is to make the connection from the abstract description of the algorithms at hand to the hardware they are to be implemented onto, with a focus on General Purpose Computing on the Graphics Computing Unit (GPGPU). In a first section, the CUDA platform, memory model and performance considerations are explained. Afterwards, these performance considerations are applied to SPH. A short discussion about single and double precision follows. Some timings and speedup discussions conclude the chapter.

6.1 NVIDIA CUDA

The CUDA platform and Application Programming Interface (API) [184] enables execution of general purpose codes on certain Nvidia GPUs by extending the C, C++ or Fortran language. That is, no knowledge of shading languages specific to graphics programming is needed, and the program does not need to be retro fit into the rendering pipeline as given, for example, by openGL or DirectX.

Even though Multiple Data Multiple Instruction (MIMD) parallelism is possible using CUDA, algorithms suitable for Single Instruction Multiple Data Parallelism (SIMD) are best candidates for parallelization using CUDA. The reason for this is two fold: One one hand, CUDA does not provide as message passing interface, making the only way to communicate between threads by writing to the shared memory. This makes MIMD parallelism inconvenient from an implementation standpoint. The other issue is due to the CUDA architecture, see figure 6.1 and section 6.3.2.

CUDA enables the programmer to run code in massively parallel fashion. These parallel code sections are coined “compute kernels”. This is a little unfortunate since the SPH mollification function is called kernel function as well. It is important to not confuse the concepts.

On the hardware side an Nvidia GPU consists of a number of Streaming Multiprocessors on a die. A multiprocessor features 32 - 192 Arithmetic Logic Units (ALU) cores , depending on the architecture. Typically dozens of Streaming Multiprocessors (SM) are contained withing a single GPU, bringing the number of CUDA cores per GPU in the range of thousands on the more high end cards, as of 2018. An example of a Streaming Multiprocessor is depicted in figure 6.1. In this example, representative of the Pascal architecture of Nvidia GPUs, two “warps” are featured. A warp consists of 32 processing units. In each clock cycle, all cores within one warp execute the same instruction, as indicated by the single

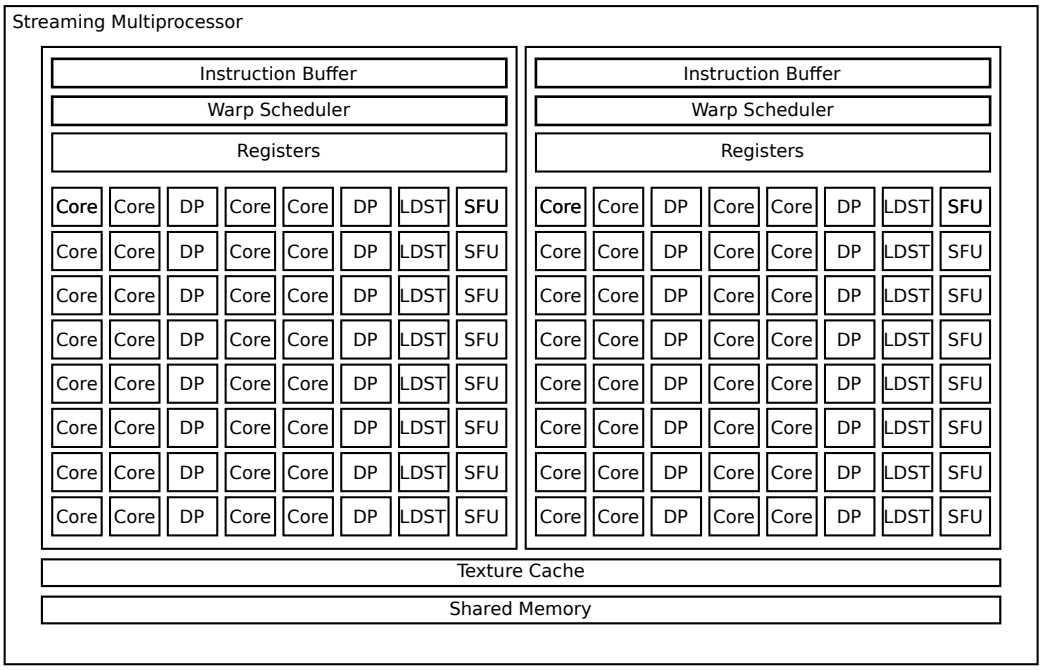

Figure 6.1: Architecture of a CUDA Streaming Multiprocessor (SM) as of Nvidia Pascal [185]. Different architectures may differ in the number of double processing units etc. per SM, but stay conceptually the same.

scheduler and instruction buffer per warp. This limits applicability to MIMD use cases, as mentioned. Double precision is handled in hardware by separate Arithmetic Logic Units (ALUs). They are smaller in number than the single precision cores, in this case the ratio is 1:2. Transcendental functions like sine or cosine are handled by separate circuitry in the Special Function Units. LoaD and STore Units (LDST) handle access to the main memory, which is not depicted in the figure. Each SM features a small portion of shared memory, enabling inter-thread communication and a texture cache for fast memory access.

All of this is in stark contrast to a Central Processing Unit (CPU) architecture, where the number of cores is per processor is small, at the time of writing at most 22, but typically much smaller at 8 or 4. A single core would feature not only an ALU but more involved logic like branch prediction and speculative execution units. Different levels of memory caching are provided per core, alleviating performance penalties imposed by cache misses. Also, each core features it's own scheduler, allwoing each core to execute different instructions in each clock cycle.

6.2 MEMORY MODEL

The memory model is sketched in figure 6.2. CUDA code is executed in what is called compute kernels during witch a grid of blocks of threads is executed. Each thread is mapped to a core and a block contains multiple warps. Thus, the number of threads needs to be divisible by the number threads per warp, i.e. 32. Some memory is thread local while other memory is block local, or even global. The following memory types are available to a CUDA programmer:

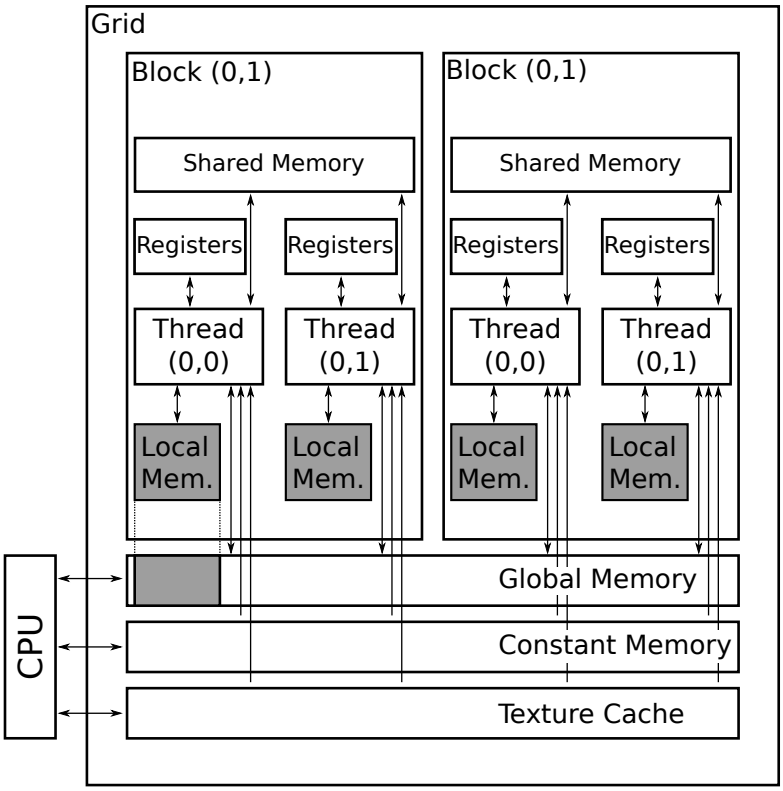

Figure 6.2: GPU memory model. Double pointed arrows indicate read/write during kernel execution, single pointed read only. The gray section in global memory indicates thread local mapped memory.

- Just like on the CPU, a number of registers is available to each thread. Access is extremely fast but the number of registers is limited. The number of registers in use per thread is subject to occupancy concerns, see 6.3.3.
- If the number of thread local variables in the program code exceeds the number of registers available, it is said to be "spilled" into global memory. Local memory is nothing but thread local mapped global memory.
- Constant memory. A small storage for constant memory, i.e. memory which stays constant throughout one kernel execution but needs to be available to all thread. It is well suited for physical constants and the like. Access is fast, size is a few KiB.
- Shared memory is block local and can thus be used to communicate between threads in a block. Access is fast if no bank conflicts are occurring. Size is still small, i.e. in the range of a few KiB.
- Global memory is similar to the main memory / RAM with regard to the CPU. Its size is in the range of a few GiB. It is high bandwidth but comparably slow, i.e. about two orders of magnitude slower than the other memory types. Access time needs to be optimized using coalescing, explained in section 6.3.1, in order for CUDA code to be efficient.
- Texture memory is a special cache / addressing mode for global memory. It can potentially speed up accesses depending on the access pattern. Factors of two up until

one order of magnitude are typical, however, as opposed to global memory, it is read only.

6.3 PERFORMANCE

Due to the architecture of the GPU special measures need be to taken to attain reasonable performance. These measures are a direct consequence of the design of the GPU, especially the single scheduler / instruction buffer per warp and the "steep" memory hierarchy. The term "steep memory hierarchy" describes the fact that on the GPU a cache miss incurs a performance hit of a few hundred cycles. On the CPU, after a cache miss there are at least two more levels of caches to be checked, with relaxed runtime penalties if the data is found on one of the three cache levels. Since these performance characteristics entail heavy consequences on the design of algorithms to be parallelized on the GPU, to most important of them are briefly discussed in this section.

6.3.1 *Coalescing*

The GPU memory controller has a granularity of 64 or 128 bytes, depending on whether the single precision cores or double precision cores are served, and it works per half warp, i.e. 16 threads. It is thus very important that memory reads and writes are aligned at 64 / 128 byte boundaries. This is illustrated in figure 6.3. Note that memory coalescing requirements used to be more strict for older devices, where reads had to be performed in sequence with possible skips. Any other pattern would entail 16 reads for a half warp

6.3.2 *Divergence*

Thread divergence is a direct consequence of the single scheduler per warp. The concepts applies to any thread that features a conditional, for example in the form of an `if` clause or `for` loop. As an example a simplified axis aligned bounding box traversal as used in ray tracing is chosen. Consider algorithm 1. If the tree is imbalanced as shown in figure 6.4, the warp will stall until all 32 rays found a leaf. In other words, if there is a conditional in a compute kernel, the runtime of a warp will be equal to the runtime of the thread taking the longest. This further emphasizes that parallelism on the GPU is best suited for the SIMD paradigm.

6.3.3 *Occupancy*

Only a limited number of registers per SM exist. If a compute kernel uses R registers, and $32 \cdot R > R^{\text{SM}}$, where R^{SM} is the maximum number of registers per SM and 32 the number of

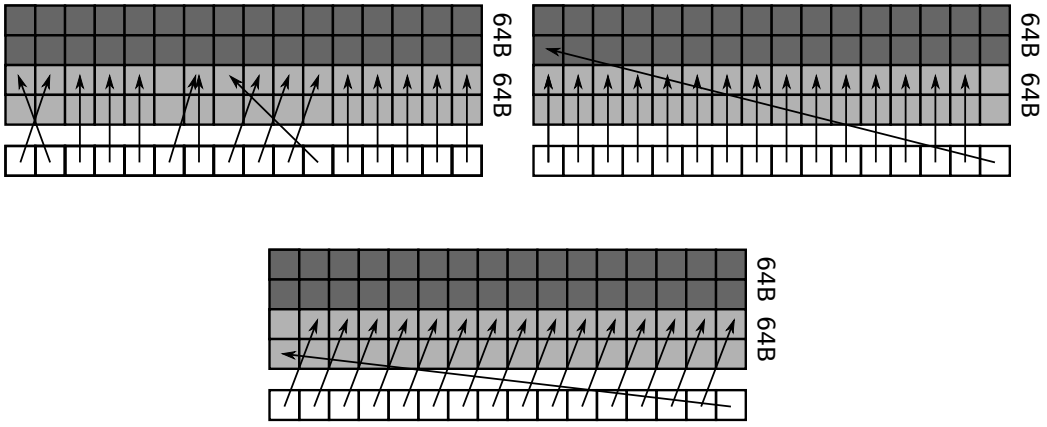

Figure 6.3: Three memory access patterns are exhibited. Two 64 byte sections of the global memory are shown in light and dark gray. Threads at the bottom access said memory. On modern architectures, i.e. after compute capability 1.2, only the top right pattern would issue two reads. Older architectures would issue 16 reads since no reads are performed in order.

Algorithm 1: Simplified tree traversal in CUDA

```
1  Function RayAABBIntersect(tree, origin, dir, outInter):
2      idx ← blockIdx.x * blockDim.x + threadIdx.x;
3      ray ← origin + dir[idx]
4      node ← tree
5      while not node is leaf do
6          if Intersect (ray,node.left) then
7              node ← node.left
8          end
9          if Intersect (ray,node.right) then
10             node ← node.right
11         end
12     end
13     outInter[idx] ← Intersect (ray,node)
```

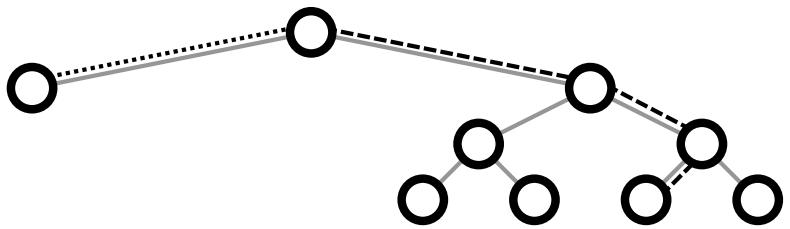

Figure 6.4: An imbalanced tree. Two traversals are shown, one immediately terminating at a leaf, dotted line to the left, another needing more iterations due to the imbalance of the tree, dashed line to the right. This will lead to heavy thread divergence in algorithm 1.

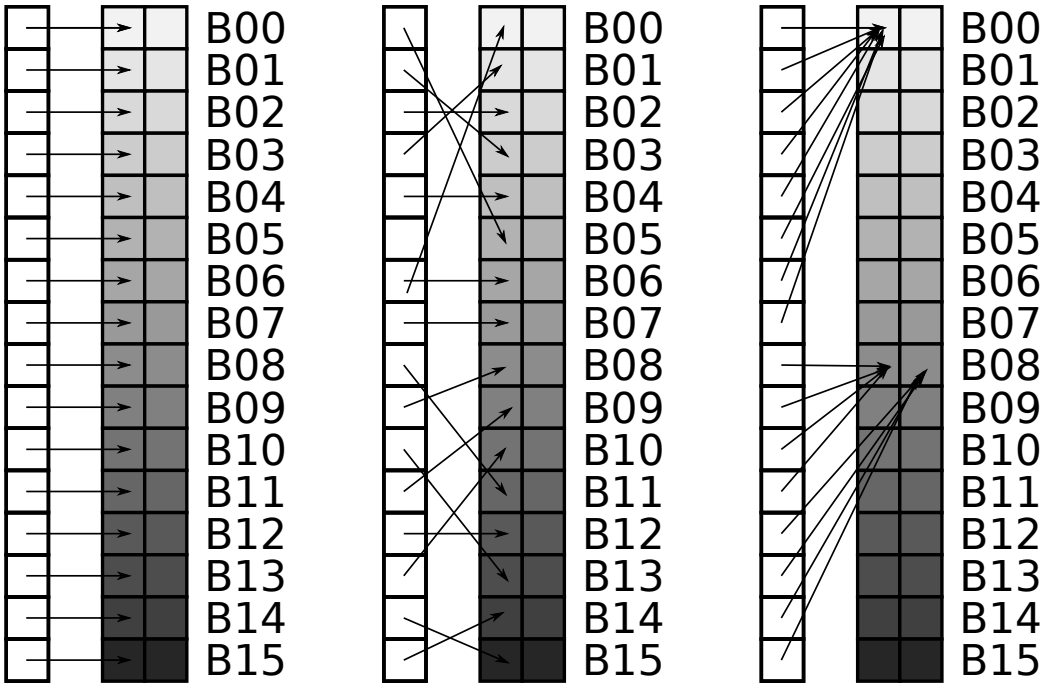

Figure 6.5: Shared memory banks and bank conflicts illustrated. Only two 4 byte words are pictured, while in reality there are one thousand. The two situations on the left do not lead to bank conflicts while the situation on the right leads to 8 serialized reads. It is important to note that the conflict is not only caused by access of the same byte, as in the top group, but also to two, or any number of bytes in the same bank, as in the lower group.

threads per warp, there are two options: either registers are spilled to local memory, which is nothing but thread local global memory or stalling another thread, borrowing its registers. The first options may impose heavy performance penalties since reads from global memory are slow, while the other option may lead to poor occupancy of the SM. This entails that best GPU efficiency is achieved if threads are "fine grained", i.e. each thread can compute its output relying on few bytes of data. This is synonym to saying that a kernel has high arithmetic intensity, as opposed to high memory intensity.

6.3.4 *Bank Conflicts*

Shared memory is divided into 16 banks, each 1 KiB of size, numbered in a stride of 4 Byte words. If two threads need to access the same bank in the same cycle, the access is serialized, i.e. takes twice as long. This is called a bank conflict. The concept is explained in figure 6.5. Hence, not only writes to shared memory need to be carefully planned in order to avoid race conditions, but also the reads, in order to avoid bank conflicts.

6.4 SINGLE AND DOUBLE PRECISION - SOME REMARKS

Single and double precision refers to whether 32 bit (4 Byte) or 64 bit (8 Byte) numbers are used. Since GPUs were originally intended for graphics applications only, single precision performance is usually a lot higher in consumer cards, since in graphics applications speed is usually more important than scientific accuracy. Factors of 24 or 32 between single and double precision FLOating Point operations per Second (FLOPS) are not uncommon. Special purpose cards with ratios 1:2 or 1:3 exist, at a much higher price. It remains to clarify whether the application at hand requires double precision calculation. Some remarks to this end:

- There are only $2^{32} \sim 4 \cdot 10^9$ single precision numbers. They are not spread out equidistant over the domain, but most dense between 0 and 1, becoming less and less dense the farther from 1. Considers Youngs modulus E, which is for example $200 \cdot 10^9$ for steel; the resolution, or spacing between numbers, is $\sim 10'000$ at this scale for single precision but $3 \cdot 10^{-3}$ for double precision.

- In fact, due to the above, the plastic stage , i.e. the root finding in the radial return, is often only convergent in double precision.

- Another problem is round off, i.e. if a small number is added to a larger one, round off of some significant bits of the smaller number may occur. If for example 100 uniform random numbers are sampled from the unit interval and added once in the original order, and once with a random permutation applied, the result will not be equal. Typically, results will differ $1 \cdot 10^{-14}$ in double precision but $1 \cdot 10^{-6}$ in single precision. Since SPH is an application heavy on sums, often with far from uniformly scaled summands, it seems imperative that such operations are handled as accurately as possible.

Figure 6.6 shows the difference in plastic strain in a metal cutting simulation that was once calculated with single precision and another time with double precision, except for the root finding in the radial return, which was double precision in both cases. Results differ quite severely. Consequently, only sentinel computations were performed using single precision, final simulations exhibited in this work are computed using double precision exclusively. The plastic stage was run in double precision at all times.

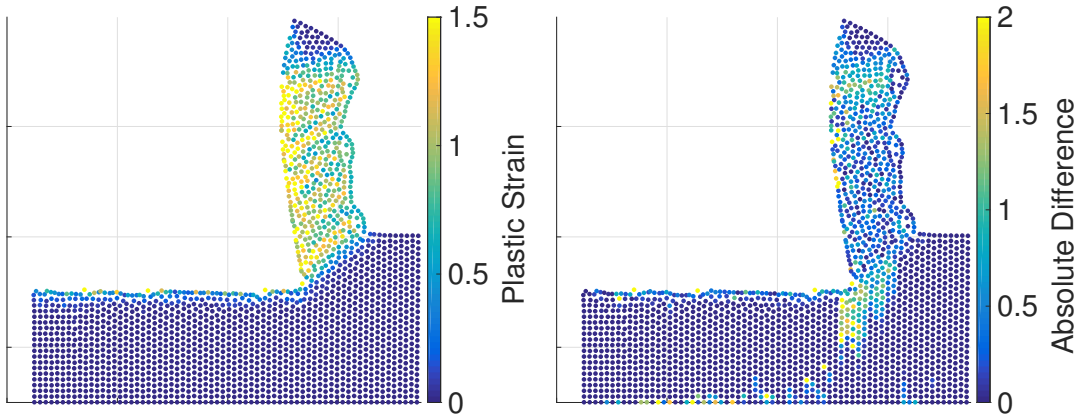

Figure 6.6: An orthogonal metal cutting simulation is run in double and single precision. The result for the plastic strain obtained with double precision is shown on the left. On the right the absolute differences to the same simulation ran with single precision are displayed. As can be seen, the differences are most pronounced in the shear zone / in ares of high strain gradients. This disagreement in the location of the shear zone between single and double precision simulations leads to the high differences observed.

6.5 OPTIMIZING PARTICLE METHODS FOR GPUS

Having discussed the GPU architecture and the performance characteristics derived thereof it is now time to apply this knowledge to implement an efficient particle solver on the GPU. Most of the ideas presented stem from the `DualSphysics` software, an industry ready CFD solver for designing coastal structures, for a general overview over the software see [54] and [53] for GPU implementation aspects. This section is divided into three topics: granularity, i.e. how much work one thread should do, how memory can be organized to minimize uncoalesced reads and thread divergence, and how the spatial hashing structure can be realized on the GPU.

6.5.1 *Granularity*

This section is concerned with the question "how much work should each thread do?". From an implementation stand point particles are hashed into boxes, and neighbors are found by cell lists, as opposed to Verlet lists. See classic texts such as [112] for explanations regarding these structures. The following threading strategies are suggested, for N particles in M boxes:

1. STPB: 1 thread per box, M threads are spooled
2. MTPB: 9 threads per box - 8 threads handle the particles in neighboring boxes while 1 thread handles particles in the box itself, $9 \cdot M$ threads are spooled
3. STPP: 1 thread per particle, N threads are spooled

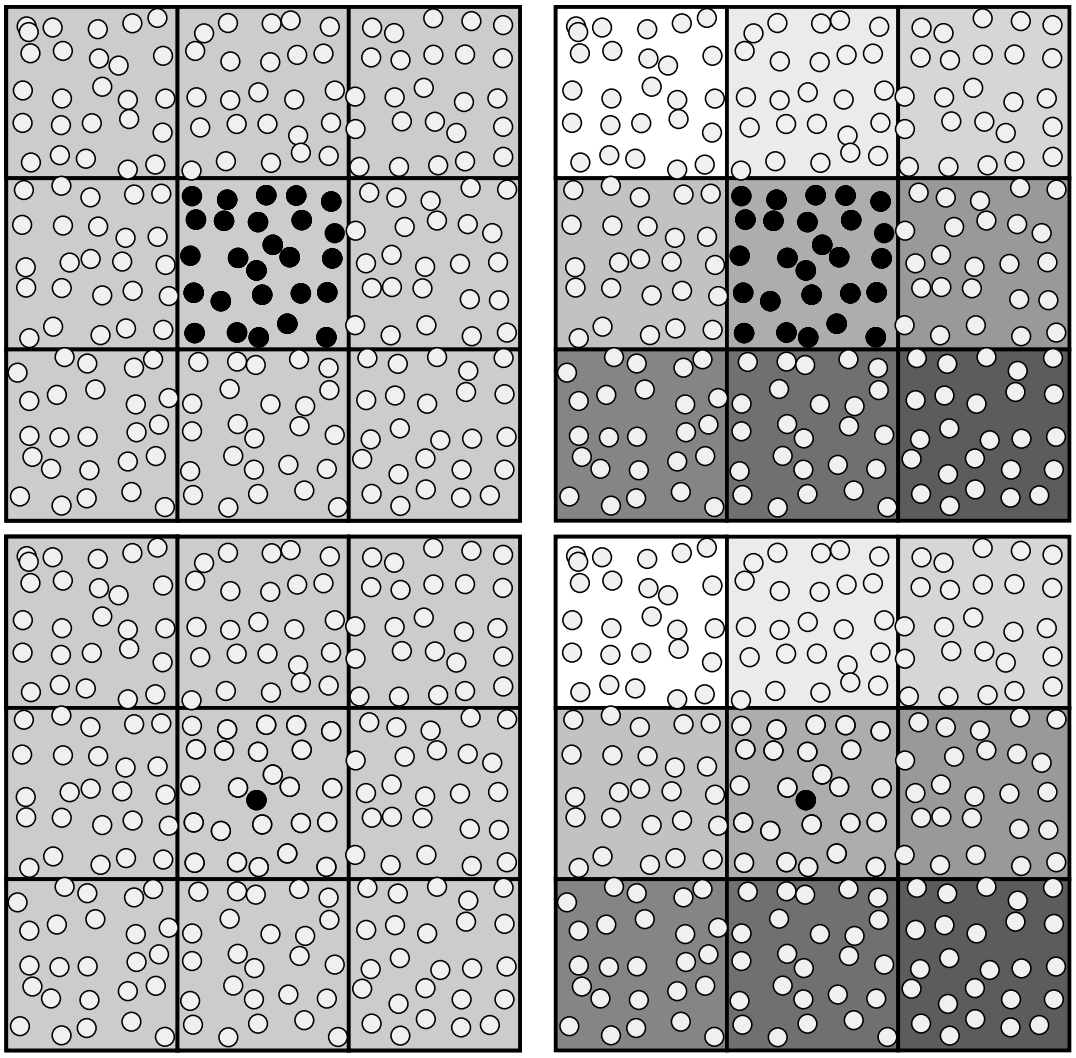

Figure 6.7: SPH threading strategies illustrated. Particles being handled, be it by 9 or 1 thread, are indicated in black. Threads are distinguished by shades of gray. Clockwise, starting from the top left: a single thread handles a full box of particles, 9 threads handle a full box of particles, a single thread handles a single particle and 9 threads handle a single particle.

4. MTPP: 9 threads per particle - 8 threads handle particles in neighboring boxes while 1 thread handles particles in the box itself, $9 \cdot N$ threads are spooled

there are some observations which can be done before benchmarking the options: since $M < N$ in most situations, option 1 will use the least amount of threads. Usually its best to oversaturate the GPU, i.e. spin up more threads than there are CUDA cores to hide latencies in scheduling. Thus, option 1 might only be useful for very large particle arrangements. Also, if there are a lot of empty boxes thread divergence might be very severe. Option two aims to feed more work to a single thread than option 3 or even 4. Option 4 will perform roughly 1/9th of the work per particle, but will also need an atomic write onto the particle itself. The same is true for option 2. The different strategies are illustrated in figure 6.7 for clarity.

A benchmark is performed for a single gradient evaluation, both for a random and for a regular particle arrangement of 250'000 particles, using both the GTX 660 and Tesla P100, c.f. table 6.4. The gradient evaluation is run a 100 times in each case since a single evaluation is very short and noise due other processes and non deterministic scheduling may mask the true trends. The results are listed in tables 6.1, 6.2 and permit quite a few interesting observations: The first thing to note is that the standard deviation does not exceed ~ 7 percent of the run times measured. The timings can thus be considered reliable.

The "winner" is clearly identified as the single thread per particle strategy. It is interesting to note that particle arrangement influences the timings quite a bit. Even on the CPU, execution time increases a considerable amount of 60%, implying that the complicated caching logic on the CPU is not a catch all for very adverse situations. The performance hit due to randomization of particles is most severe on the GTX 660. This makes sense since divergence imposes to highest performance penalty on the machine with the lowest single thread performance. Overall the threading strategy is important, causing runtime differences of up to a factor of 5. Performance takes a dive as soon multiple threads are employed per entity, be it boxes or particles. This is because in that case additions to particles need to be performed in using atomic operations, i.e. the variable needs to be locked before write access to avoid race conditions. Managing the lock and serializing the access seems to be very detrimental on both the Tesla and the GTX.

Scrutinizing literature gives the impression that there does not seem to be a clear consensus in the GPGPU community about which of the approaches is best in general. `DualSPHysics` uses the same single thread per particle strategy, while the researchers in [90] use an improved version of option 2 involving shared memory. The autors in [5] forego cell lists completely employing Verlet lists instead. While it is not clear how the other researchers decided on their threading strategy since their works do not cover any benchmarks to this end, it has to be noted that the benchmark performed here is synthetic in nature and results are subject to change would the whole application be examined in the same fashion.

6.5.2 *Organizing Memory*

Having chosen single thread per particle as the strategy to adopt for this work it is left to discuss how to avoid thread divergence and uncoalesced reads. As the method stands now, thread divergence may be quite severe in boundary and especially corners of the current simulation geometry. See figure 6.8 for an explanation. There is probably no way to avoid thread divergence in particle methods on the gpu completely as long as there are free surfaces, however, there is a quite simple strategy how to avoid at least some of it: simply subdivide the cells. At first it seems counter intuitive how this would improve the situation, since the number of non zero summands in the SPH approximation with respect to some particle stays constant. However, since the cell lists are not perfect in the sense that the kernel function is radial, and cell lists are squares. Cell lists may thus include quite a few particles not contributing to the SPH sum. It is hoped that figure 6.8 illuminates this situation. This idea was, to the best of the autors knowledge, first introduced in [53].

Table 6.1: Timings for the various graining strategies on the GTX and Tesla, c.f. 6.4. 250'000 particles in a regular arrangement, evaluation of a single SPH gradient using formula (4.32). Time is in seconds. Speedup against CPU taking 0.1 seconds, with a standard deviation σ of 0.001. Times are averaged over 100 runs and standard deviation is recorded.

Method		GTX	
	Time [s]	σ [s]	speed up
STPB	0.0046	0.0003	22.47
MTPB	0.0116	0.0003	9.02
STPP	0.0034	0.0001	31.28
MTPP	0.0129	0.0003	8.11

Method		Tesla	
	Time [s]	σ [s]	speed up
STPB	0.00055	0.0002	190.06
MTPB	0.00284	0.0009	36.91
STPP	0.00045	0.0002	232.63
MTPP	0.00281	0.0009	37.23

Table 6.2: Timings for the various graining strategies on the GTX and Tesla, c.f. 6.4. 250'000 particles in a randomly sampled arrangement, evaluation of a single SPH gradient using formula (4.32). Time is in seconds. Speedup against CPU taking 0.16 seconds, with a standard deviation σ of 0.003. Times are averaged over 100 runs and standard deviation is recorded.

Method		GTX	
	Time [s]	σ [s]	speed up
STPB	0.0270	0.0001	6.22
MTPB	0.0327	0.0004	5.13
STPP	0.0120	0.0005	13.99
MTPP	0.0239	0.0011	7.01

Method		Tesla	
	Time [s]	σ [s]	speed up
STPB	0.00081	0.0001	208.56
MTPB	0.00403	0.0006	41.74
STPP	0.00079	0.0001	212.44
MTPP	0.00403	0.0001	44.59

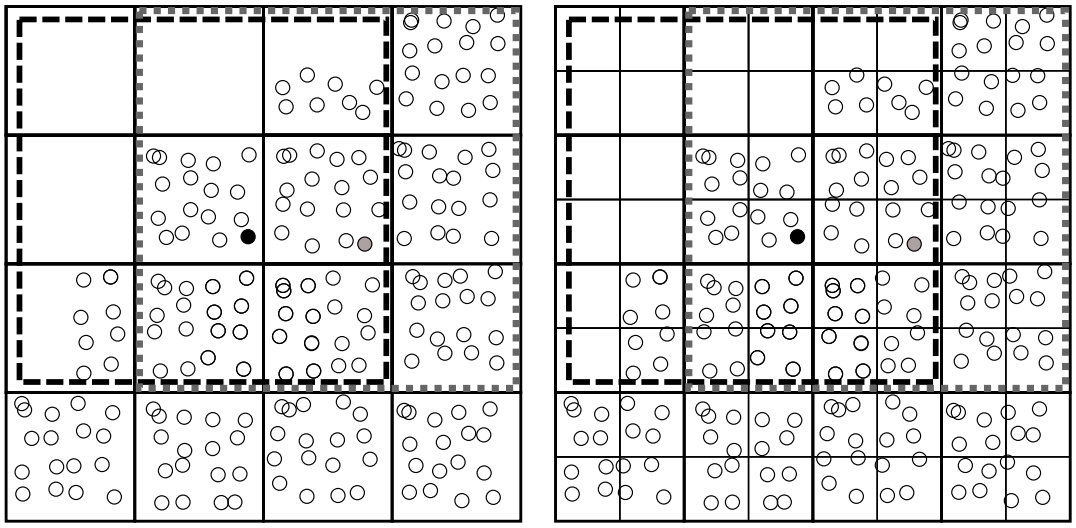

Figure 6.8: Thread divergence with regard to SPH. The black and gray particle will almost certainly end up in the same warp, at least if boxes are numbered choosing x as the "fast" index. The boxes the two particles interact with are framed using black dashed and gray dotted lines, respectively. By subdividing the grid, the black particle is not affected at all, while the gray particle can skip quite a few particles, reducing thread divergence. Put in numbers: 85 and 127 particles before subdivision, 85 and 100 particles after subdivision.

In 6.3.1 the importance of avoiding uncoalesced reads was highlighted. By using cell lists in conjunction with spatial hashing, the situation is quite pleasing in this regard from the outset, since particles belonging to the same box are adjacent in memory. However, one can still affect the situation by the numbering / hashing scheme used. The familiar hashing scheme $hash(\underline{p}) = \left\lfloor (p_y - p_y^{\min})/h \right\rfloor \cdot n_x + \lfloor (p_x - p_x^{\min})/h \rfloor$ from 5.4.3 leads to a situation as depicted in figure 6.9. This means that one can expect at least 5 uncoalesced reads for each thread. Unfortunately there is not much to be done about that, another scheme called z-ordering, see [174] was tried but proved to improve the situation only marginally, if at all.

6.5.3 *Spatial Hashing*

Plenty of remarks have been made how to efficiently handle the SPH sum evaluation. Most of the remarks made depend that cell lists are present. What was not touched upon so far is how this structure can be constructed efficiently on the GPU. To this end, it is worthwhile to examine the classic cell list construction algorithm on the CPU first. The procedure is illustrated in figure 6.10. The procedure can be concisely summarized in three steps:

1. Compute hash of each particle
2. Reorder all particle data according to hashes
3. Linearly scan particle hashes. If a hash does not equal the hash of the particle ahead, update a cell array accordingly.

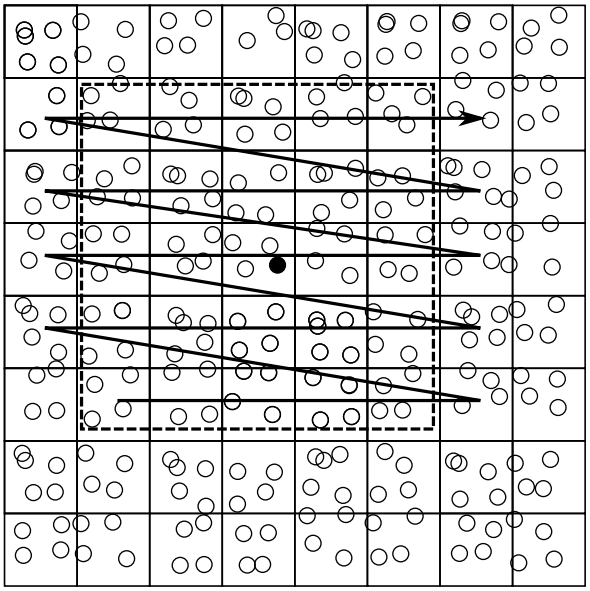

Figure 6.9: The particle in black is handled by a single thread. Memory layout is linear with x major ordering / x is the "fast" index. At least 5 uncoalesced reads are triggered in this situation.

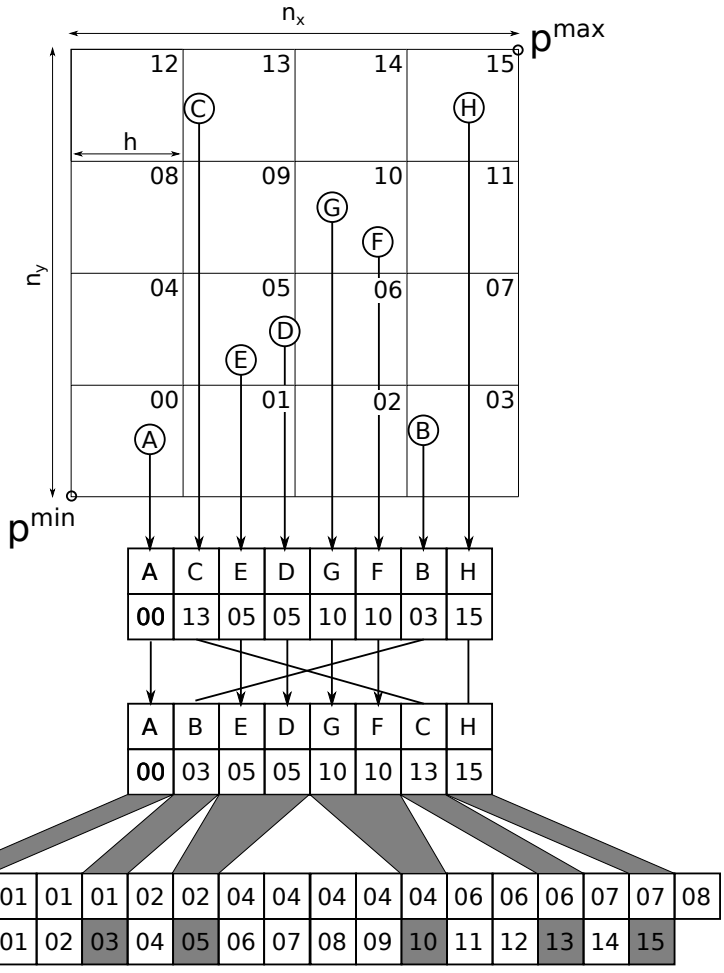

Figure 6.10: Sequential spatial hashing algorithm of particles illustrated.

The first step of this procedure can be parallelized trivially with the usual one thread per particle strategy. How data can be efficiently sorted on the GPU is beyond the scope of this thesis, suffice to say that traditional divide and conquer methods like quicksort and mergesort Originally are unsuitable. Quicksort was first descirbed in [111] while mergesort was first reported by by Goldstine and von Neumann. Their report does not seem to be public, however. Implementation details with regard to mergesort can be found in [130]. For the GPU, radix sort proved to be a highly efficient candidate for sorting data, see [228] for design and implementation aspects.

The most interesting part seems to be point 3 of the spatial hashing algorithm. The algorithm relies on a linear scan, hence it does not paralellize in straight forward manner. Two options were explored to this end. One approach is given by Green in [93], the other is published in a masters thesis [224]. A more involved approach not explored for this thesis is discussed in [113].

The approaches differ mainly in buffering strategies. Green [93] suggests to assign a thread per particle, load hashes to shared memory, then perform the check whether the hash of the next particle equals the current hash on the shared memory. Data is then reordered on the fly to an output buffer. This necessitates an output buffer of the same size as the original particles for each attribute like position, velocity, stress and the like. Hence, this approach effectively doubles the memory requirement.

[224] on the other hand suggests to compute the required permutation using a series of calls to functions in the `thrust` namespace, which provides highly efficient parallel primitives for CUDA, like gather, scatter and reduction operations, see [20]. Data is then reordered using an out of place gather operation for each particle attribute. This only necessitates additional memory for each attribute *type*, e.g. scalar, vector, tensor of second order. However, the approach is quite a bit slower since the number of gather operations required is quite high in solid mechanics applications due to the high number of particle attribtues. Some timings with this regard are given in 6.3. It can be observed that the approach by [93] runs about 15% faster, but consumes more than twice as much memory. Due to these results it was decided to employ the approach by Green as long as the simulation data fits into the GPU memory, and fall back to the approach in [224] otherwise.

Note that a hybrid approach would also be possible, i.e. the cell lists could be computed using the apporach by Green, but instead of reordering all the data on the fly, just an array of sequential integers could be reordered on the fly. This array could then be used as input for an out of place gather operation just as in [224]. This option was not explored in this thesis.

Table 6.3: The rubber rings impact is run with $\sim$ 100'000 particles, using two approaches to construct the cell lists. Timings on the GTX, c.f. table 6.4, additional memory estimated for single precision

Approach	Time [s]	Add. Memory [MiB]
Approach by [93]	210.18	11.2
Approach by [224]	243.76	5.2

6.6 BENCHMARKS

While some select timings were interspersed in the previous sections to highlight some properties, this section presents systematic timings of complete simulations, after taking a look at the hardware employed.

6.6.1 *Graphic Cards Used*

The following two graphic cards were employed for the benchmarks at hand.

- The Nvidia GTX 660
- The Nvidia Tesla P100

Performance numbers of the GPUs as well as the CPU used as reference are given in table 6.4. The single precision performance is a neat illustration of Moore's law [173], which states that computing power roughly doubles every 18 months. The same can not be observed for double precision since the GTX was never designed to crunch double precision numbers. Tesla cards avaibale in 2012 on ther other hand feature double precision performance of $\sim$ 1 TFLOP, again confirming Moore's law.

In the following, three benchmarks were selected and timed on all processing units given in listing 6.4. The Tesla P100 was run with ECC correction turned off. ECC is a memory feature which guards against bit flip errors caused by neutrons from cosmic ray secondaries. Using

Table 6.4: Performance of the GPUs and CPU used. Performance is given in Tera FLOPs (TFLOPS). Memory figure for CPU is RAM available to the CPU. The Intel i7 features 4 cores, but only one is employed for all measurements.

Name	SP	DP	Memory	Year	Cores	Purpose
GTX 660	1.8	0.078	2 GiB	2012	960	Games
Tesla P100	9.3	4.7	12 GiB	2016	3584	Scientific
Intel i7-3770	0.224	0.112	16 GiB	2012	4 (1)	Desktop

Table 6.5: Rubber ring impact timed for various particle numbers, run on the GTX 660 as well as the Intel i7, see table 6.4 for details.

N	Intel i7 DP	GTX SP		GTX DP	
	Time [s]	Time [s]	speed up	Time [s]	speed up
4232	309.23	14.86	20.80	48.97	6.31
17344	1789.54	45.32	39.49	168.81	10.60
39160	4429.00	79.26	55.88	355.81	12.45
69920	8201.59	138.20	59.34	621.91	13.19
109304	13481.19	211.61	63.71	963.96	13.99
157504	19397.01	296.33	65.46	1384.01	14.02
214656	26888.94	399.65	67.28	1877.64	14.32
280472	35537.18	527.85	67.32	2463.72	14.42
355064	45722.41	669.95	68.25	3156.39	14.49
438496	56076.33	890.39	62.98	4141.80	13.54

ECC entails a performance penalty since error correction and checking has to be performed. It is mostly used if long term persistency is required by the application, e.g. a web database. The simulations on the CPU were only run in double precision. The reason for this is twofold: One hand, the code was never intended to run in single precision and conditional compilation would need to be retro-fitted to this end, which amounts to a considerable effort. On the other hand, only neglibile performance improvements are expected would the code be run in single precision on the CPU for the reasons outlined in 6.6.5.

While the GTX seems to be an odd choice due to its age, it presents an interesting contrast to the other options: Both the GTX and the Tesla are massively parallel units, featuring thousands of cores, but the Tesla outperforms the GTX in single thread performance and number of threads. This is contrasted by the Intel i7, which is run single threaded only, featuring the highest single thread performance by some orders of magnitude. Consider the single thread performance in GFLOPS: GTX 660: 0.08 - Tesla P100: 1.31 - Intel i7: 28.

6.6.2 *Elastic Benchmark*

In this benchmark two rubber rings are collided. See section 7.2.2 for the specifics. The simulation was chosen since it is very dynamic, requiring constant re-computation of the cell lists and dynamic, uneven loads per thread. That is, particles on the boundary of the rings will encounter a lot of empty boxes while their neighbors further inside the ring but in the same warp will encounter mostly full boxes as neighbors. That is, thread divergence is expected. Runtimes and speedups are listed in tables 6.5, 6.6. The simulation was run only once for each card this time since the noise by scheduling evens out due to the very high number of time steps taken. The speed ups are additionally plotted in figure 6.11.

The first thing to notice is that the speed ups achieved by both the GTX as well as the Tesla seems overly excessive. The maximum speed up achievable in double precision can be

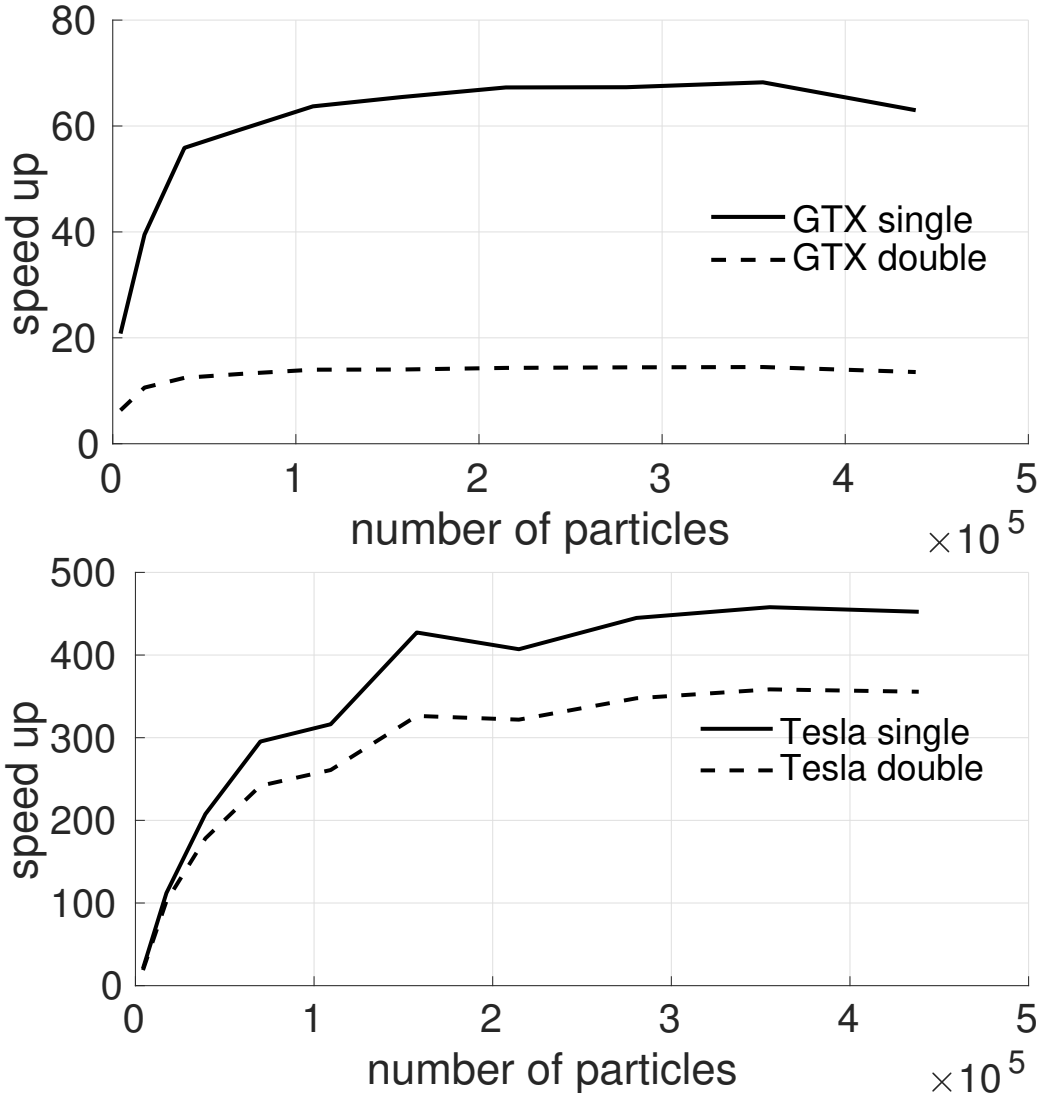

Figure 6.11: Speedup vs number of particles for both the GTX (left) as well as the Tesla (right). Note the different scaling of y axis.

Table 6.6: Rubber ring impact timed for various particle numbers, run on the Tesla P100 as well as the Intel i7, see table 6.4 for details.

N	Intel i7 DP	Tesla SP		Tesla DP	
	Time [s]	Time [s]	speed up	Time [s]	speed up
4232	309.23	14.90	20.75	16.40	18.86
17344	1789.54	15.97	112.05	17.52	102.15
39160	4429.00	21.33	207.60	24.86	178.15
69920	8201.59	27.77	295.34	33.91	241.86
109304	13481.19	42.62	316.29	51.67	260.90
157504	19397.01	45.40	427.29	59.44	326.34
214656	26888.94	66.06	407.04	83.55	321.85
280472	35537.18	79.87	444.91	102.20	347.72
355064	45722.41	99.85	457.90	127.55	358.45
438496	56076.33	123.97	452.35	157.68	355.62

computed by dividing the peak FLOP numbers $4.7/0.112 \sim 42$, see 6.4, but is measured much higher 6.6, 6.11. This means that only a fraction of the CPU power is employed. There is a number of reasons for this:

- The number of FLOPS achieved by the CPU is computed assuming that 4 (8) additions as well as 4 (8) multiplications are scheduled per cycle in double (single) precision using SSE instructions. Streaming SIMD Execution (SSE) is an extended instruction set available on some CPUs allowing for vectorization of multiple arithmetic operations in a single cycle. This is hardly ever the case in an SPH sum evaluation. In fact, profiling the CPU code with the tool `perf` [56] reveals that SSE is heavily used to copy data, but hardly used for arithmetic operations. SSE was only introduced by compiler optimization. No manual vectorization of code was attempted.

- The execution on the CPU is single threaded. This means that the maximum speed up extends to $4.7/(0.112/4) \sim 168$.

- The CPU code was hardly ever optimized. That is, maintainability, readability and consistency with source material was more of a concern when developing the code than execution time.

Otherwise the speed up graphs seem reasonable. Only moderate speed ups are observed for low particle numbers. For higher particle numbers the speed up increases since more memory latency can be hidden by the additional work. Finally, the speed up stays constant as soon as the GPU is fully saturated. It is important to note that while these graph look like an illustration of Amdahl's law [7], they are not. Amdahl's law relates the speed up to the number of available processors, not to the amount of work load.

But what about the speed up between the two GPUs? In single precision, a speed up of 7.2 is observed while in double precision a speed up of 26.3 is measured. This is in relation to theoretical maximum speed ups of 5.16 and 60, respectively. Which means that neither the

Table 6.7: Measuring the signed distance of 800'000 particles to 8'000 triangles representing a sphere, see figure 6.12. Particles are seeded in a thin layer inside and outside the triangles.

N, M	Intel i7 DP	GTX SP		GTX DP	
	Time [s]	Time [s]	speed up	Time [s]	speed up
8k, 800k	1.98	0.27	7.15	0.54	3.63

N, M	Intel i7 DP	Tesla SP		Tesla DP	
	Time [s]	Time [s]	speed up	Time [s]	speed up
8k, 800k	1.98	0.02	99	0.03	66

GTX runs at peak single precision nor does the Tesla at peak double precision. The highly optimized nbody code in CUDA toolkit exhibits speed ups of 5.5 and 26, which is not too far off from the measured numbers.

6.6.3 *Contact Search in 3 Dimensions*

In this section, the algorithm from section 5.4.3 is parallelized using CUDA. Using the findings from 6.5.1 a single thread searches for the closest triangle. That is, the safe procedure outlined in 5.4.3 was ignored for the timings, although it was implemented on the GPU, and can optionally be turned on. Since the triangle mesh is assumed rigid, no recomputation of the triangle cell list needs to be performed during the simulation. Hence, a discussion analogous to 6.5.3 with regards to triangles is mute; the cell lists are simply constructed on the CPU and subsequently copied to the GPU.

It is not straight forward to construct arbitrarily scalable test data for the contact search since the number of triangles in a mesh is not a free parameter. Unfortunately, the drill model 5.15 can not be used if single precision performance is to be tested. Some triangles are so badly conditioned that they become degenerate, i.e. a line or point in single precision. Thus, a more well behaved model representing a sphere, c.f. 6.12 is considered. Again a large number of particles are seeded along a thin layer on each side of the model. This constitutes a realistic load with regard to a penalty contact algorithm. A single timing is performed and reported in table 6.7.

It can be seen that the speed up is moderate compared to the elastic benchmark 6.6.2, especially for the GTX. This is not because of lacking size of the working set but because the application is memory bound. That is, the ratio of memory operations to arithmetic operations is worse when loading a triangle and measuring the distance than loading a particle and computing the kernel. GPUs excel in tasks with high arithmetic intensity.

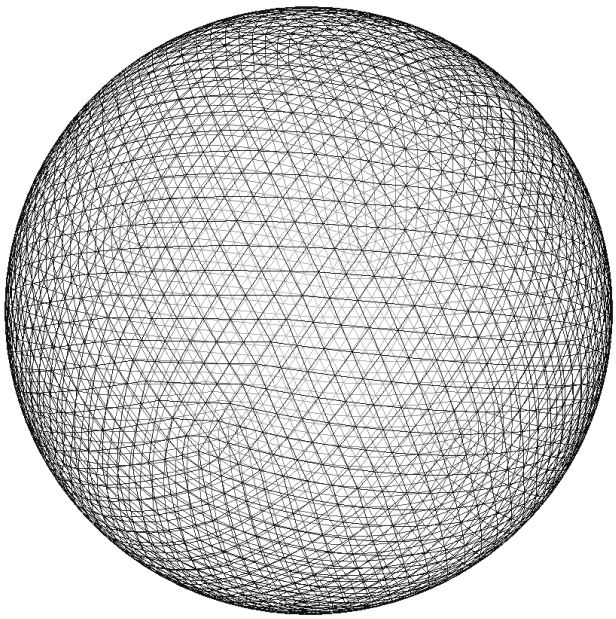

Figure 6.12: The sphere mesh used for benchmarking the 3D contact search on the GPU. The sphere is quite finely resolved employing 8'000 triangles.

6.6.4 *Orthogonal Metal Cutting*

In this final benchmark an orthogonal metal cutting simulation is considered using Monaghans algorithm 5.1.1 [91]. The cutting parameters are the same as in section 7.5, except that the simulation was run at unphyscial speads of 500m/min. This was done to bring the high resolution simulation on the CPU into the realm of feasibility.

Two situations are considered: On one hand a low resolution simulation is carried out, just enough to recreate the results obtained by LSDYNA presented in [225]. Execution time is compared to an own CPU implementation. Furthermore, a high resolution simulation is carried out, which is equal to the highest resolution used in section 7.5.3.2. Results are summarized in table 6.8. Again, it can be seen for example by the almost identical runtimes of single and double precision on the Tesla in the lwo resolution case that the cards are not saturated. Still, using the Tesla, a speed ups of about 15 and about 60 are achieved in the low and high resolution case, respectively.

Unfortunately, the runtimes can not be compared to the LSDYNA simulations presented in [225]. LSDYNA does not support contact between shell elements and SPH particles, necessating that the domain is modelled using a "2.5" dimensional approach, see figure 6.13. This entails that the computation is carried out using the 3 dimensional constitutive laws, and that the particle number has to be increased about tenfold for a result of the same quality. On current hardware using a single CPU the runtimes are about 13 hours for the low resolution case and about 40 days for the high resolution case. The 40 day simulation was not actually performed, but the runtime was extrapolated from a partial run.

Table 6.8: Timing the orthogonal metal cutting case from 7.5. Timings are in seconds.

N	Intel i7 DP	GTX		Telsa	
	DP	SP	DP	SP	DP
6'000	3'481	277	534	270	271
150'000	127'542	10'438	36'222	2'375	2'388

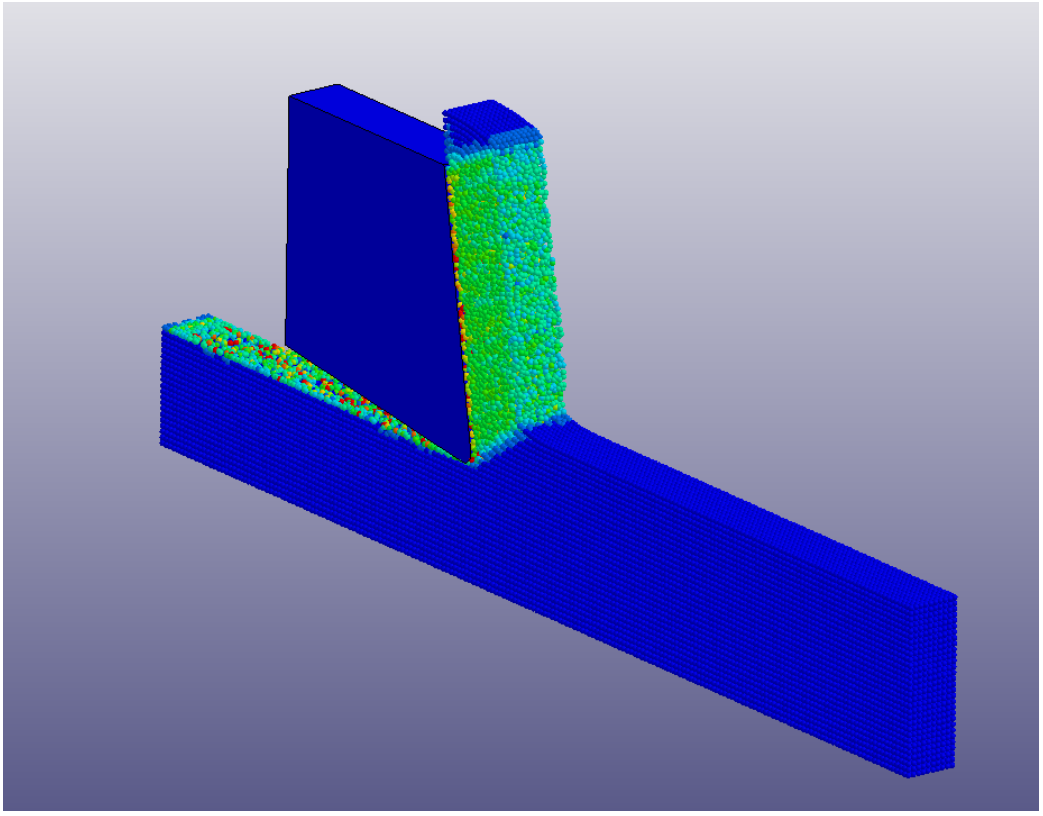

Figure 6.13: Typical LSDYNA result. The domain has to be modelled three dimensionally to establish contact with the FEM tool. Color indicates plastic strain.

6.6.5 *Assessing Performance*

While the speed ups achieved seem quite substantial it is not quite clear how they compare to the highest speed ups realistically obtainable. That is, code can not be perfectly vectorized on the CPU to achieve 16 (8) instructions per cycle. And any code computing something useful will not be completely free of conditionals, thus thread divergence, nor compppletely free of uncoalesced reads, limiting how much of the raw computing power of a GPU can be exploited. The section is concluded with some remarks to this end:

- Doing an analysis using `nvprof`, the profiling tool provided with CUDA reveals that the most important compute kernel, i.e. the one computing the particle interactions exhibits low occupancy. This is because it uses a high number of registers. This is to be expected, since a lot of particle attributes, like position, velocity, and stress tensor, just to name a few, need to be loaded. That is, threads are not fine grained enough to fully occupy current GPUs. However, in section 6.5.1 it was revealed that trying to further subdivide the work, i.e. employing multiple threads per particle, entails atomic writes which severely degrade performance. This fact seems to be the main conundrum for further improving GPU accelerated particle methods, at least for solid mechanics.

- To further optimize the kernels at hand, a roofline model analysis [276] should be performed. However, this was beyond the scope of this thesis.

- The only GPU accelerated industry ready SPH solver, i.e. `DualSphysics` speeds up with a factor of about 60, see [53]. However, the benchmarks in the report are quite dated and would need to be repeated on current hardware.

7

RESULTS

While results are interspersed into this thesis to highlight some key points or features of algorithms, this section more systematically investigates benchmarks and compares the algorithms from chapter 5. In a first step, their most basic functionality is investigated by trying to resolve rigid body motions. Afterwards, some elastic benchmarks are presented and two select algorithms are expanded to 3D. A plastic impact problem demonstrates what algorithms might be useful for metal cutting simulations, which is clearly dominated by plastic deformations. The results of these preliminary benchmarks invite a short reflection on the algorithms' performance. Orthogonal metal cutting tests follow right after. This is the most comprehensive exploration, such that the section was split in three parts: updated Lagrangian domain, total Lagrangian domain and some applications specific to the GPU, which happen to be on the updated Lagrangian domain. The section concludes with a single grain cutting test, including some features new to metal cutting simulations like use of linear complete kernels, TVF and artificial stresses and execution on the GPU.

The numbering of the Algorithms refers to chapter 5. They are quickly re-mentioned in table 7.1 for quick reference. Additionally, own implementations of updated and total Lagrangian FEM are used as reference in most situations. In cases necessitating advanced FEM features like hour glassing control, the `LSDYNA` FEM solver is employed, which works on the updated Lagrangian frame. For metal cutting benchmarks, the SPH solver in `LSDYNA` is used for comparison as well.

Table 7.1: List of algorithms benchmarked. Frame denotes Updated or Total Lagrangian frame and Form refers to Weak and Strong form. **Bold** text denotes a shorthand for the algorithm.

#	Name	Frame	Form	Pub.
1	**Monaghan**s artificial stress	U	S	[91]
2	Transport Velocity Formulation (**TVF**)	U	S	[91]
3	**Godunov** type SPH	U	S	[200]
4	**Reveles**' Lagrangian SPH	T	S	[217]
5	**Potential** Based SPH	T	S	[178]
6	**Co-rot**ated SPH	T	S	[19]
7	**Weak u**pdated **L**agr. RKPM	U	W	[-]
8	**Weak t**otal **L**agr. RKPM	T	W	[127]

7.1 RIGID BODY MODES

In this section the basic working of the algorithms is established by considering simple spinning of the unit disk. The other rigid body mode, i.e. translation is uninteresting since all methods are Galilean invariant. Either by employing a first order complete scheme or by a scheme of shape (4.32). The angular velocity is chosen as 50/s. It has to be emphasized that the angular velocity was imposed as initial velocity only, not as a boundary condition. While conceptually a simple benchmark, this situation is still interesting in combination with meshless methods, especially in light of remarks made in section 5.1.1.

A virtual, linear material is considered, with properties listed in table 7.2. It features the E-Modulus of rubber, however is slightly compressible and has an unphysical density of one. This makes the material easy to work with from a numerical point of view since $c_0 = \sqrt{K/\varrho_0}$, and the maximum time step is given by the Courant Friedrich Lewy (CFL) condition:

$$\Delta t^{\max} = CFL \frac{h^{\min}}{||\underline{v}||^{\max} + c^{\max}} \tag{7.1}$$

Where CFL is a constant with $CFL < 1$. Hence, a large speed of sound allows for large time steps, which is achieved by choosing a small ϱ. As a time stepper, the Runge Kutta method of 4th order (RK4) was chosen, see for example [31].

Table 7.2: Linear Elastic, "rubber"

Parameter	Value	Unit
Youngs modulus E	10^7	Pa
Poissons ratio ν	0.4	-
Density ϱ	1	kg/m^3

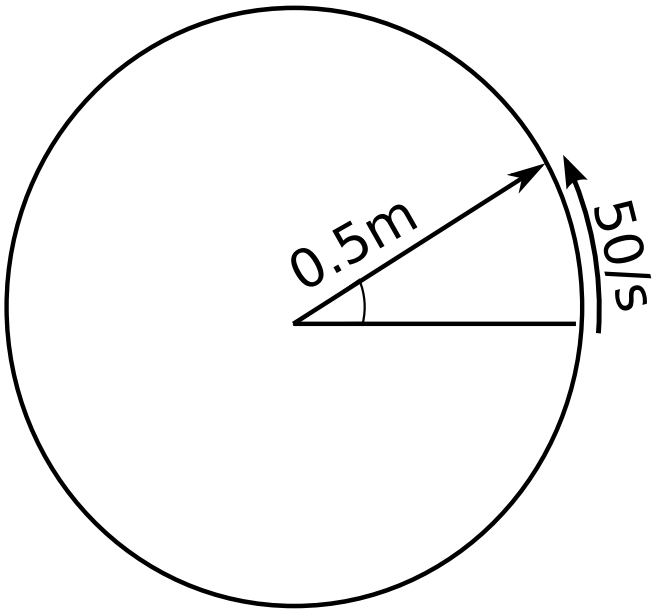

Figure 7.1: Setup of the angular momentum benchmark. The unit disk is expected to keep spinning with initial angular velocity.

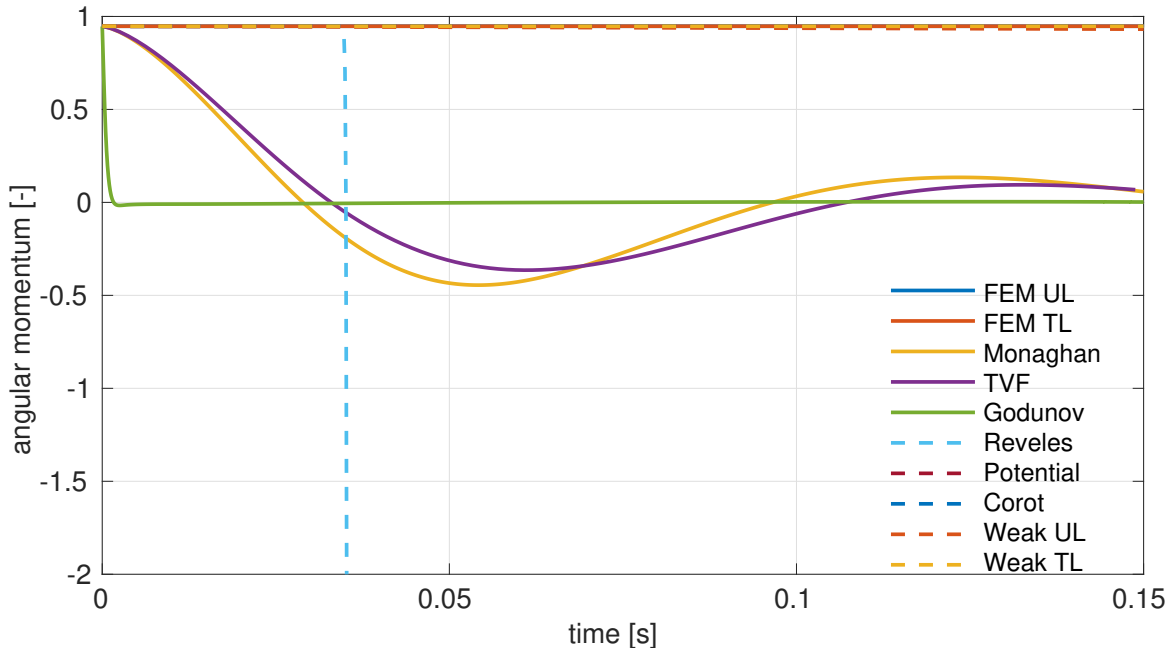

Figure 7.2: Results of the the angular conservation bench mark.

Results are summarized in graph 7.2. The current angular momentum was recorded using:

$$\mathcal{H} = \sum_{i=1}^{N} x_i m_i v_i^y - y_i m_i v_i^x \tag{7.2}$$

and normalized using the analytical expression for the angular momentum for the disk at hand:

$$\bar{\mathcal{H}} = \frac{1}{2} M R^2 \cdot \omega \tag{7.3}$$

with total mass M, disk radius R and angular velocity ω. That is $\mathcal{H}/\bar{\mathcal{H}}$ was recorded. Note that $\mathcal{H}/\bar{\mathcal{H}}$ is smaller than one from the outset, since the discretization underestimates the mass.

The results do indeed confirm the remarks made in 5.1.1. All methods which are either first order complete or use some other means to establish proper handling of rotations, that is the co-rotational SPH, conserve angular method almost exactly. One notable exception is Reveles' algorithm, which diverges in this situation after less than a quarter rotation. It is not clear why this is the case. Activating SPH formulation `FORM=8` in `LSDYNA`, which seems to be an algorithm very similar to Reveles', shows the same behavior. The Godunov algorithm is very dissipative and the rotation of the disk stops almost immediately. Neither the TVF nor Monaghans algorithm are linearly complete, thus they do not conserve angular momentum. In fact, they completely reverse the direction in which the disk spins, losing more and more energy and eventually come to a stop.

7.1.1 *Fixing algorithms 1 and 2*

As mentioned, the situation about conservation of angular momentum can potentially be rectified by replacing W with CSPM, Randles Libersky $\overset{\triangle}{W}$ or RKPM kernels $\overset{\circ}{W}$. While this will correctly resolve rotations, approximate stress free boundaries are lost because the method is linearly complete. So is conservation of *linear* momentum, since $\overset{\triangle}{W}_{ij} \neq \overset{\triangle}{W}_{ji}$, $\overset{\circ}{W}_{ij} \neq \overset{\circ}{W}_{ji}$. It is not clear what situation is preferable from a conceptual stand point. Implementing RKPM kernels into algorithms 1 and 2 makes them diverge in only a few time steps, while using CSPM or Randles-Libersky kernels makes the algorithms conserve angular momentum with the same quality as the FEM references, see figure 7.3.

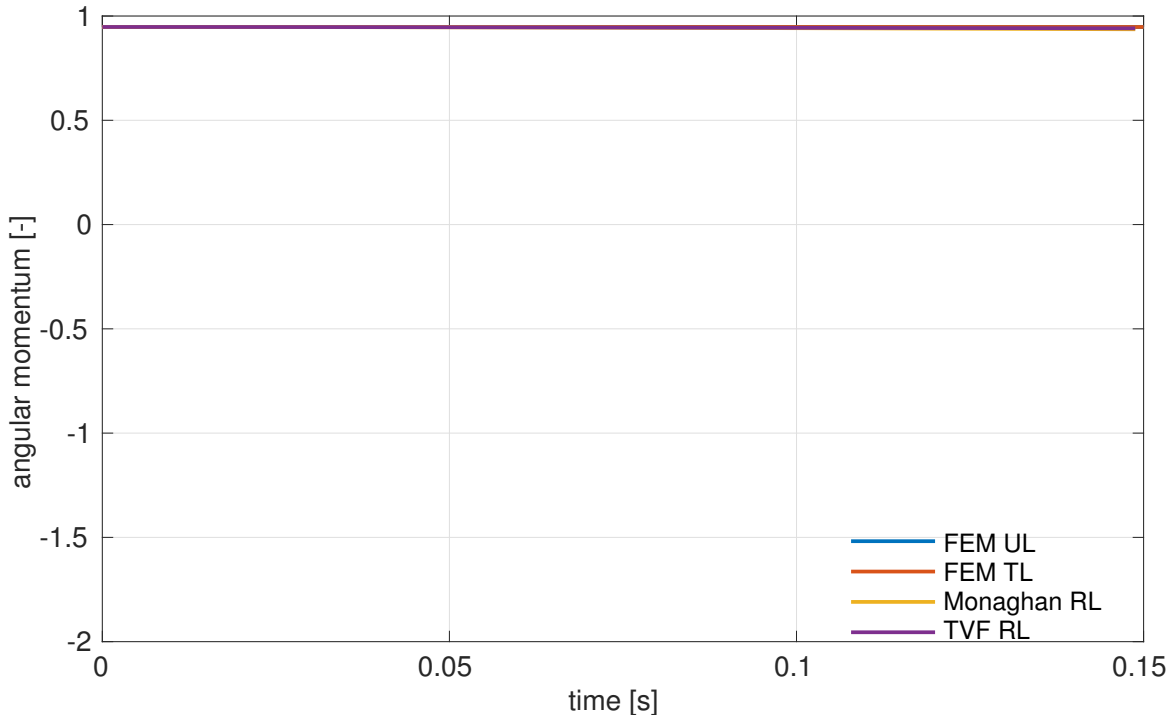

Figure 7.3: Results of the angular conservation bench mark for algorithms 1 and 2 employing Randles-Libersky kernels.

7.2 ELASTIC BENCHMARKS

In this section two elastic benchmarks are investigated. The tensile test is probably the most classic material test. It is still an interesting situation to consider in a meshless setting due to the presence of the tensile instability 4.6.1, application of Dirichlet boundary conditions and, since the ends of the tensile specimen are fixed in all directions completely, large rotations encountered at the corners.

The other benchmark considered is a more dynamic setting, where two rubber rings, or, due to the plane strain condition, rather rubber cylinders, collide violently, then reflect off each other, and vibrate freely. This benchmark was apparently introduced in a memo by Swegle in [254], although the memo is not publicly available to verify this claim. However, it has has been a stable in meshless benchmarks ever since, see for example [172], [91], [200], [267], [282].

7.2.1 *Tensile Test*

The basic setup of the tensile test considered can be seen in figure 7.4. Since the material model was chosen linear elastic, the tensile test was not modeled after a tensile test according to standard EN ISO 6892-1 [116], say. Instead, a square geometry and large dimensions where chosen. This is to ensure that the algorithms at hand are pushed to the limit what they can represent as soon as possible, due to the large vertical contraction and large forces

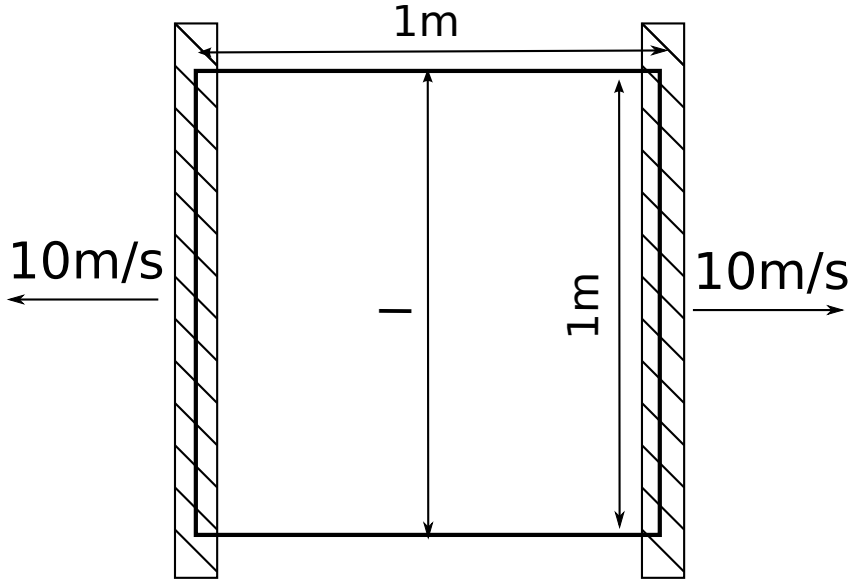

Figure 7.4: Sketch of the tensile benchmark. Ends are fixed in all directions. Contraction is recorded in the middle of the specimen.

ensured by the set up at hand. The time stepper chosen was again RK4 and the simulation was run for 0.025 seconds, such that the specimen extends 0.25 meters on both sides, which corresponds to 50% extension. This is extremely large for a linear elastic material and does not correspond to a physical situation.

All algorithms on the updated Lagrangian strong form, that is, algorithms 1-3 were excluded from this benchmark since they are not able to represent the 50% extension demanded. Some discussion to this end, and why the algorithms are still able to complete the tensile test considered in [91], indicating that they were indeed implemented correctly, is given in appendix 9.2.

For the other algorithms, two results were recorded. On one hand the normalized contraction, i.e. l/l_0, where l is the current and reference vertical distance from the two particles situated in the middle, respectively. See figure 7.4. Additionally, the last frame of the simulation for each method is exhibited in figure 7.7.

The graph of contraction 7.4 reveals some interesting results. It can be observed that the methods cluster around two final contractions, i.e. 25% and 35%. While it is not surprising that the corotational SPH and Potential based SPH stray from the FEM results, since they were designed for computer graphics applications. Thus, they do not necessarily need to follow the correct solution, but need to look plausible for the causal viewer. It is also not surprising that they produce similar results, since they were constructed in similar fashion. The real surprise is that the updated Lagrangian weak SPH follows the two computer graphics algorithms so closely. Of course, it is a very dissipative method, requiring re-approximation as well as interpolation of the velocity field onto the quadrature points. Thus, some lack of contraction is to be expected. That this lack of contraction ends up to be the same amount as the lack of contraction in algorithms 5 and 6 is certainly a curious coincident.

To shed some light on this situation, all simulations were re-run using another set of material parameters, namely the elastic parameters of steel 4430, see table 7.3. The first thing to

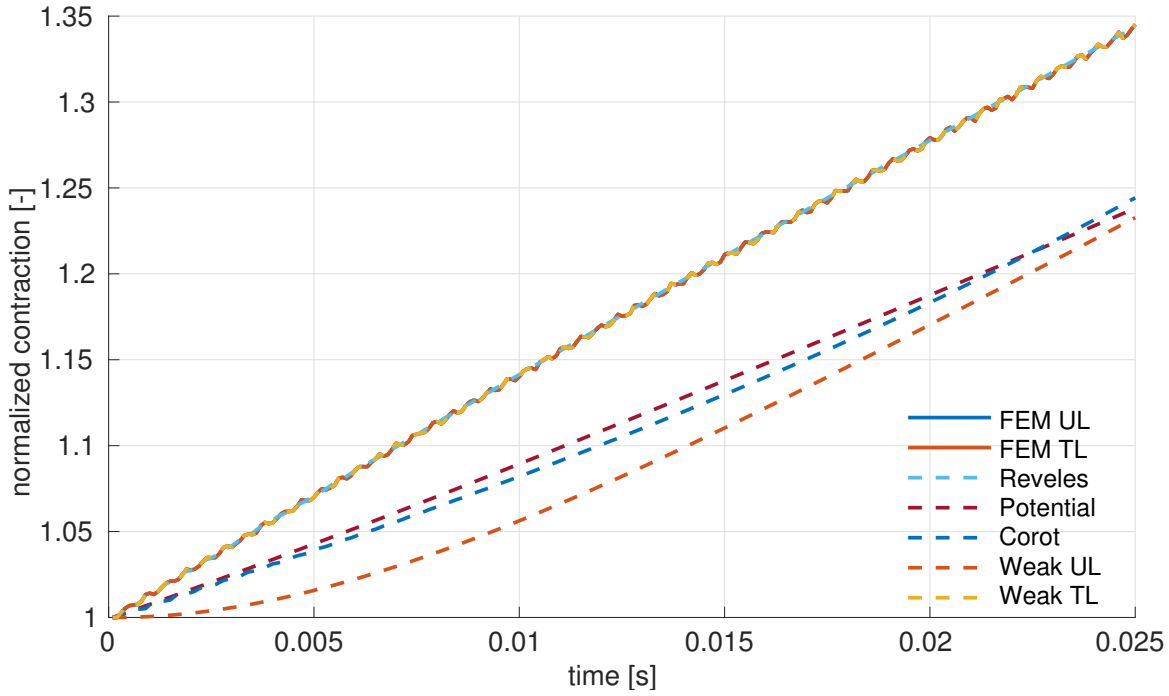

Figure 7.5: Results of the tensile benchmark. Contraction was recorded. Note the two clusters of results.

note is that the contraction is less than in the rubber-like case, due to the lower Poissons ratio. Again, the FEM results and Reveles algorithm are virtually indistinguishable. The potential based and corotational algorithm are still lacking in contraction but are closer to the cluster comprising of FEM results / Reveles, while the updated weak form algorithm lacks in contraction most severely. It seems that the potential based algorithm, corotational algorithm and the updated weak form lack contraction increasingly with increasing Poissons ratio at different rates, crossing over at $\nu = 0.4$.

The difference in contraction is hard to spot in the final frames 7.7 since it differs by 10% only. What can be seen is that the particle arrangements differ quite severely. Especially the co-rotated SPH shows some oscillatory modes. Again, since the methods primary application is computer graphics, this may not be an issue for the original purpose, since it is not the discretization points that are rendered on screen.

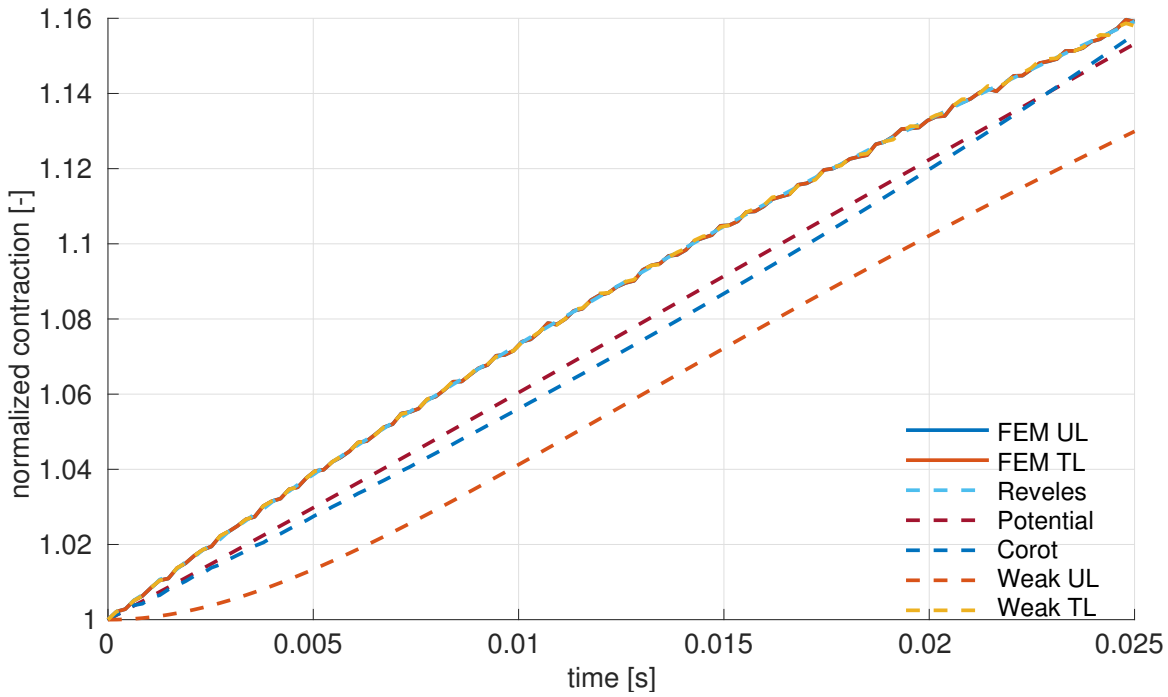

Figure 7.6: Results of the tensile benchmark with alternative material data. Contraction was recorded.

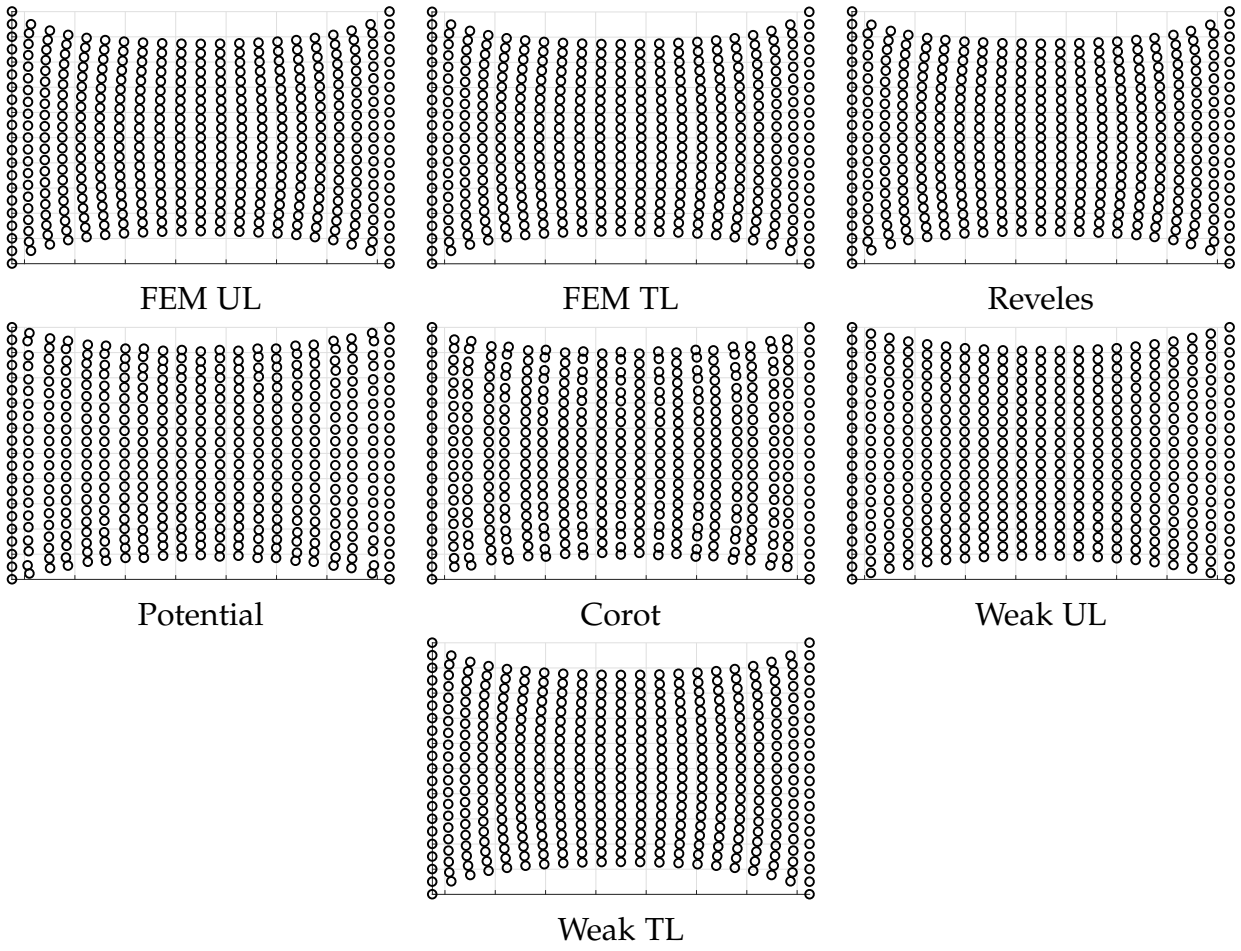

Figure 7.7: Final particle distribution of the tensile test for all methods. For the FEM methods, only the nodes are displayed such that the visualization is comparable to the particle methods.

7.2.2 *Rubber Ring Impact*

In this section two rubber rings are collided with a velocity of $180m/s$ each. That is, with relative velocity of $360m/s$. It would be possible to run the updated Lagrangian simulations without a contact algorithm, i.e. resolving the collision by SPH particle interaction only, since the collision is mostly head on and free from sliding motion. However, to keep all algorithms comparable, the contact algorithm from section 5.4 is employed in all cases. Time stepping is again handled by the RK4 time stepper. However, the contact algorithm restricts the value of the CFL constant quite heavily, such that lower order schemes offer the better trade off between performance and accuracy.

Again, two kind of results were recorded. Figure 7.9 shows the height of the left ring, over time, while figure 7.10 exhibits the frame where the height was at maximum. Note that the potential based method was excluded from this benchmark since the contact algorithm used proved unsuitable for the method, necessitating unreasonably small timesteps for operation.

The history of compression reveals that the results are quite different this time around. This makes sense since this is the most dynamic benchmark. The two FEM methods are indistinguishable, while Monaghans and the TVF algorithm follow closely. So does Reveles method, however, Reveles method required quite large artificial viscosity constants in order to remain stable after the impact.

Both meshless weak form algorithms are numerically dissipative in this case. The updated Lagrangian one more so than the total Lagrangian one. In fact, the updated Lagrangian algorithm does not exhibit free vibrations after the impact. It makes sense that the weak form algorithms are prone to amplitude decay since they both require a reapproximation step, see the discussion in 5.2.7. The updated Lagrangian algorithm needs to interpolate velocities onto the gauss points, which is smoothing the velocity field even more. The Godunov type algorithm is the worst offender with regard to numerical damping, a fact which is acknowledged in the original publication [200], where the same benchmark is performed.

The last thing to note is that the co-rotated SPH is out of phase with regard to the FEM references. This means that it reacts very sensitively to the contact algorithm. Due to the

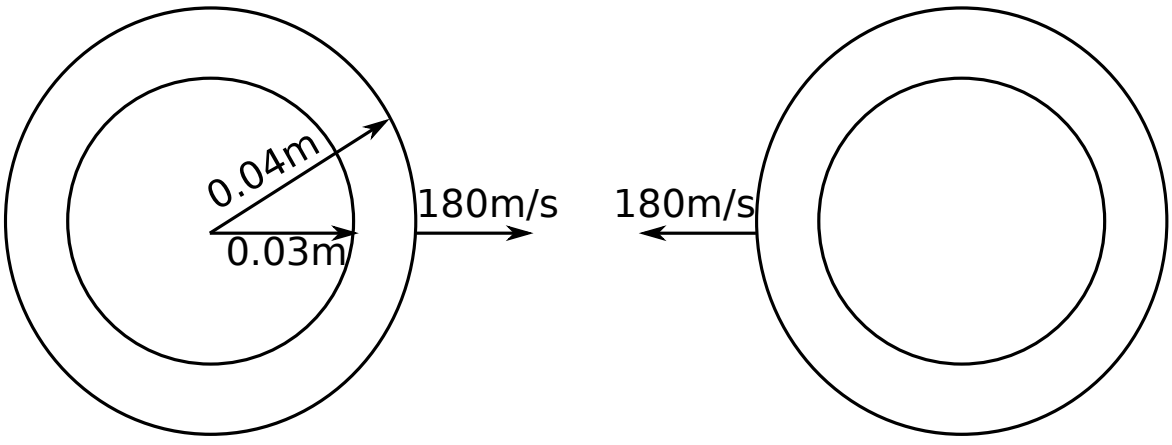

Figure 7.8: Two rings fly against each other with a velocity of 180m/s.

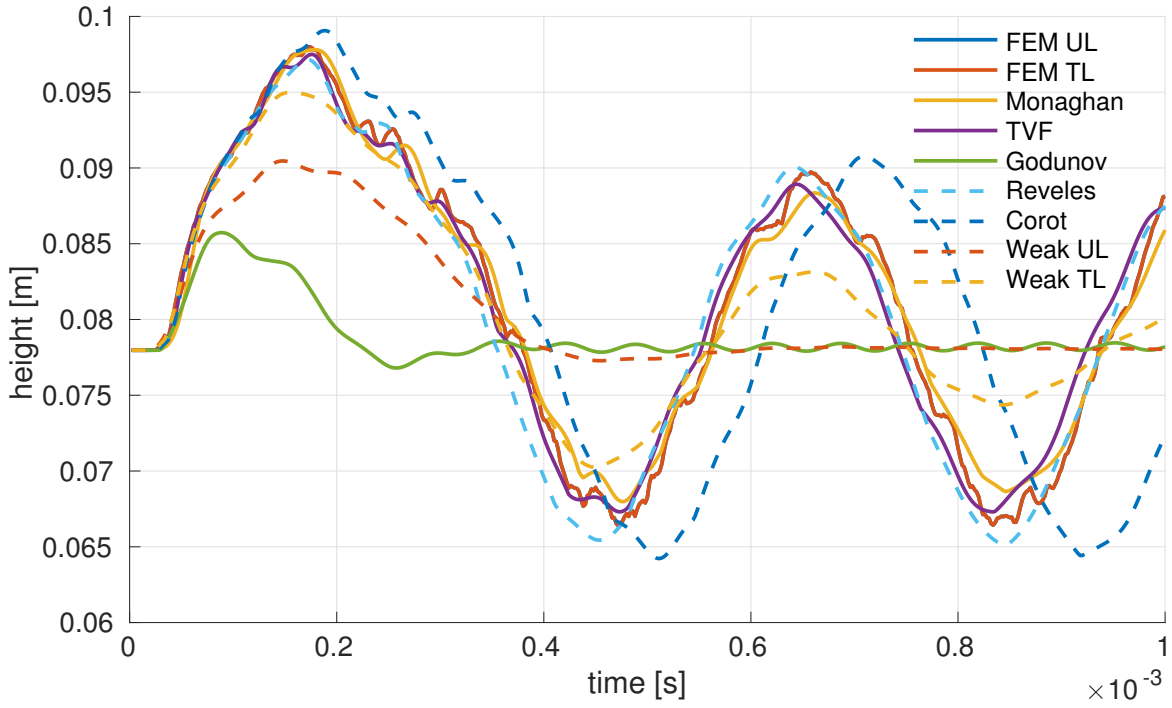

Figure 7.9: Result of the rings benchmark. The vertical extension of the left ring is recorded.

similarity of the co-rotated SPH and the potential based approach, this is in line with the observation that the potential based algorithm does the same up to such a degree that finding a stable timestep becomes impractical.

The figure 7.10, where result frames at maximum vertical extensions are shown, directly reflects the amount of numerical dissipation discussed. The lack in the strength of the impact, which develops over some dozens of time steps, is clearly visible for the Godunov and Weak UL methods.

7.2.2.1 *Exploring Corrections Measures in Algorithm 1*

Algorithm 1 is amongst the algorithms most closely following the FEM reference solutions. However, one of the main criticism with algorithm 1 is that it depends on three different stabilization measures, each featuring one to three tunable parameters, see section 5.1.1. Tuning of these parameters has been shown to significantly alter the physics simulated under some conditions, see [122]. It is thus worthwhile to investigate the behavior of the algorithm if some of these measures were to be deactivated. The rubber ring benchmark described was thus run three times: once disabling both artificial viscosity and artificial stresses, once disabling artificial stresses and, finally, using the full array of correction measures.

The results are exhibited in figure 7.11. The result frames were taken shortly before the rings would reflect off of each other. As can be seen, the result without any stabilization measures fails catastrophically. The introduction of artificial viscosity fixes some of the numerical fracture, but not all of it. Indeed, only the fully corrected simulation is able to complete the simulation without numerical artifacts, revealing that Monaghans algorithm relies on the complete set of correction measures for correct working of the method.

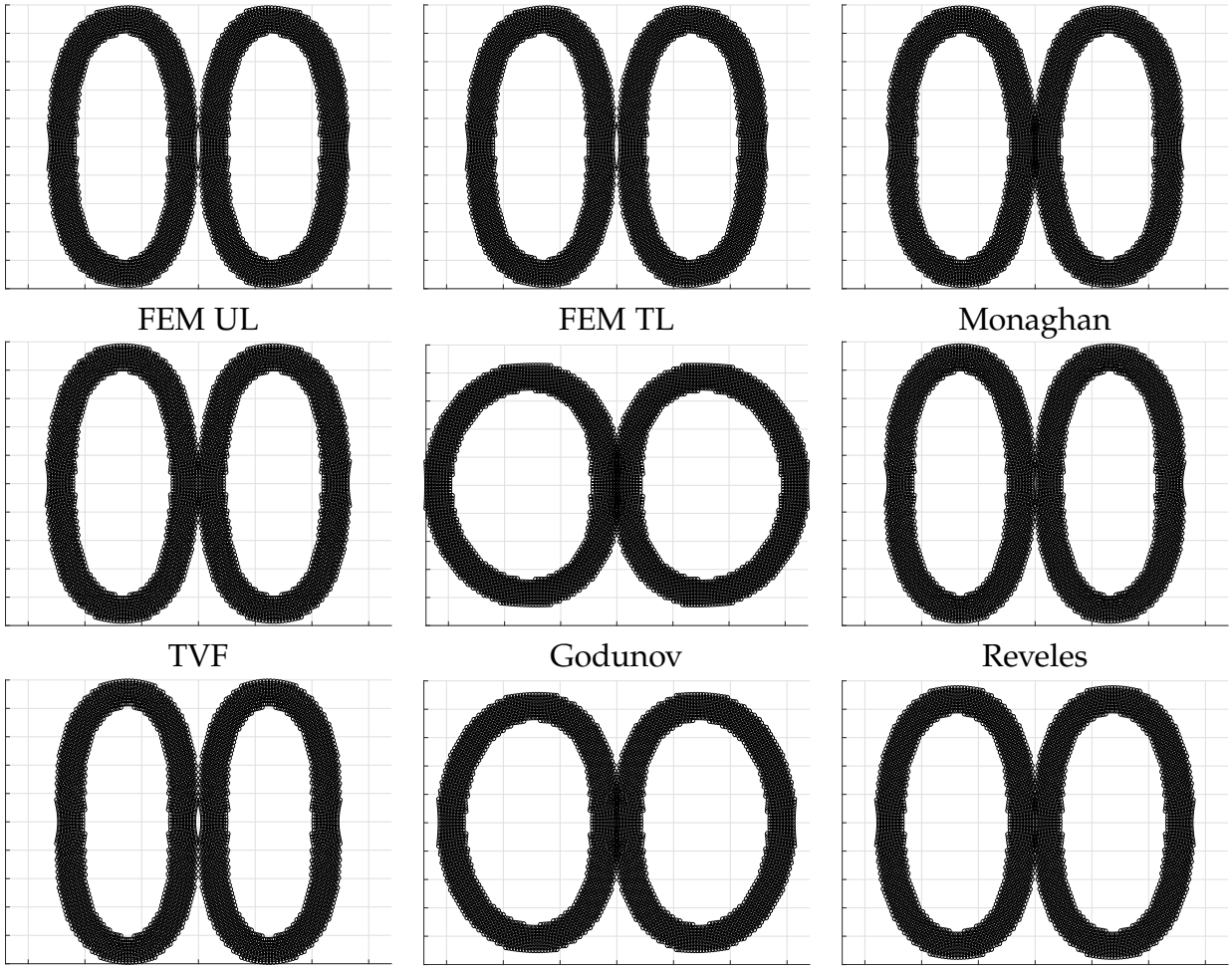

Figure 7.10: Particle distribution at the instant where the left rings reached maximum height. The ring shapes are a direct reflection of the numerical dissipation observed in figure 7.9

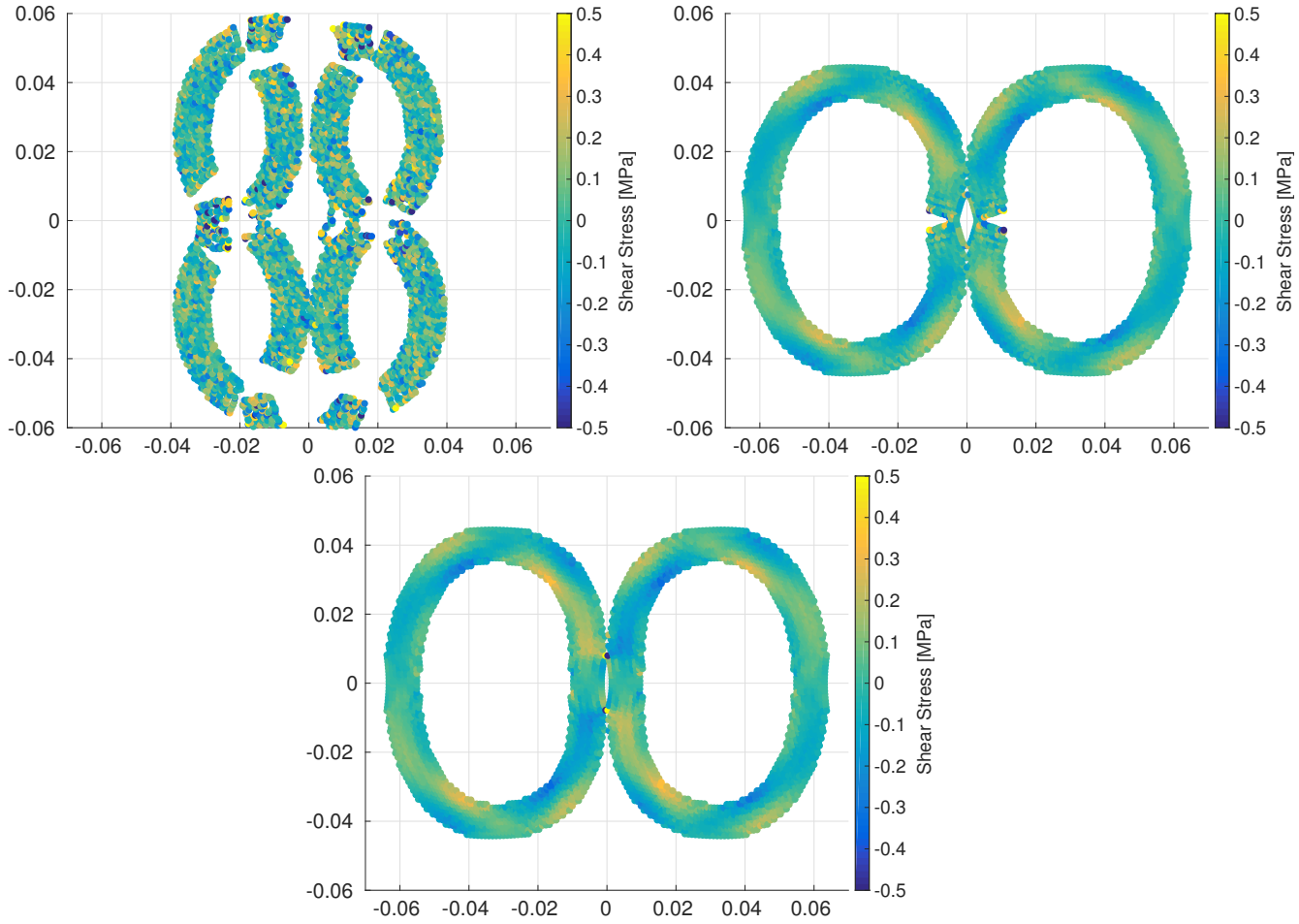

Figure 7.11: Three different versions of Monaghans algorithm. Top left with no correction terms except XSPH, top right using artificial viscosity, bottom using both artificial viscosity and artificial stresses.

7.2.2.2 *Exploring TVF and Artificial Stresses in 3D*

In a first step, a version of the Monaghan algorithm lacking the artificial stress terms was ported to 3D. This is because diagonalizing the stress tensor in a fashion that allows back rotation is not a straight forward task in 3D, at least not if all edge cases are to be considered while still keeping the code efficient. In fact, the eigen decomposition of the stress tensor needs to be computed:

$$\underline{\underline{\sigma}} = \underline{\underline{Q}} \cdot \underline{\underline{\Lambda}} \cdot \underline{\underline{Q}}^{-1} \tag{7.4}$$

where $\underline{\underline{\Lambda}}$ contains the Eigen values, which are equal to the principal stresses. Since $\underline{\underline{\sigma}}$ is symmetric $\underline{\underline{Q}}$ will be real valued, and is more commonly written as $\underline{\underline{R}}$. This is not to be confused with the rotation matrix obtained from the polar decomposition of $\underline{\underline{F}} = \underline{\underline{R}} \cdot \underline{\underline{U}}$. Since $\underline{\underline{R}}$ is a rotation matrix it holds that $\underline{\underline{R}}^{-1} = \underline{\underline{R}}^{T}$. Efficient algorithms for this purpose, both closed and iterative, are described in in the classic text book [89]. [231] describes an efficient implementation in modern `C++` and provides the source code as free software, available at `https://www.geometrictools.com/GTEngine/Include/Mathematics/GteSymmetricEigensolver.h`, retrieved 11.04.2018. Description freely available at `https://www.geometrictools.com/Documentation/RobustEigenSymmetric3x3.pdf`, retrieved 11.04.2018. This is the implementation that was incorporated into the code at hand.

In a first step, the impact of two rubber *spheres* was considered. While this situation does not correspond to a situation representable in two dimensions, since plane strain would imply rubber cylinders, and plane stress rubber rings, it is the most natural in the sense that it could, in principle, be carried out by experiment. For example by having two tennis balls collide. To make the situation more challenging, the wall thickness of the rubber rings was reduced to 0.005m and the velocity increased to 250 m/s, for a relative impact velocity of 500m/s. It turns out the situation is stable, even without stabilizing the algorithm by artificial stresses or the transport velocity. Some frames of the simulation are given in figures 7.12. Figure 7.13 shows the intensity of artificial stress and transport velocity formulation, respectively.

In a next step, the impact of two rubber cylinders was considered. In this case, the uncorrected formulation fails, but both the transport velocity and the artificial stresses remove any numerical fracture, see figures 7.14 and 7.15. This simulation highlights that numerical fracture is just as much of a problem in 3D as in 2D, and both artificial stress and the TVF are effective measures to counter the problem.

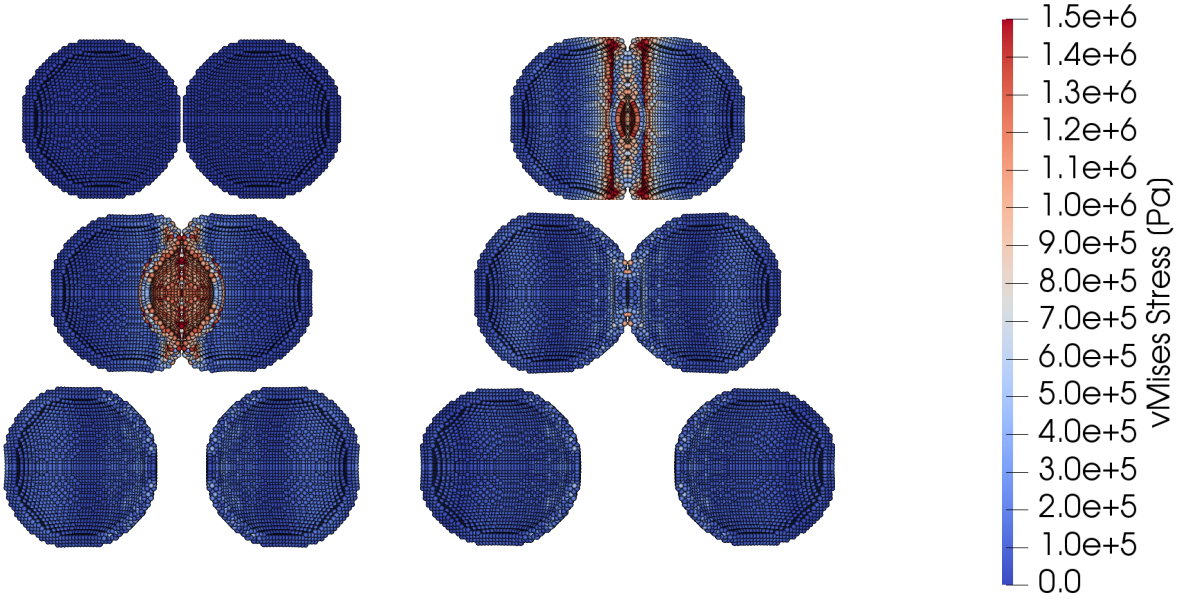

Figure 7.12: Particle distribution and von Mises stress of the sphere impact benchmark. No correction measures are applied. The sphere was cut in half for better view of the dynamics at hand. The impact is very violent as can be seen from the large inflection of the balls. Frames are left to right, top to bottom.

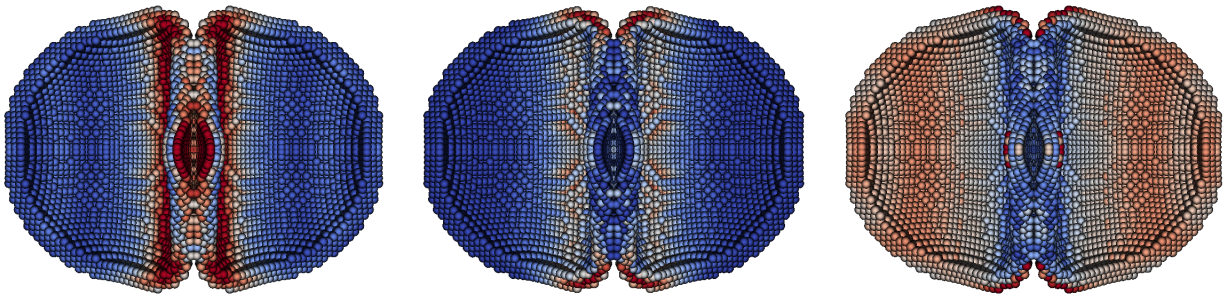

Figure 7.13: The colliding spheres, roughly at maximum height. Left is the uncorrected simulation displaying the von Mises stress. In the middle artificial stresses were employed, color indicates von Mises stress of the artificial stresses imposed. TVF on the right, color is intensity of transport velocity. While the transport velocity is more evenly active on both spheres, the heavily deformed top and bottom of the spheres is corrected the most in both correction schemes.

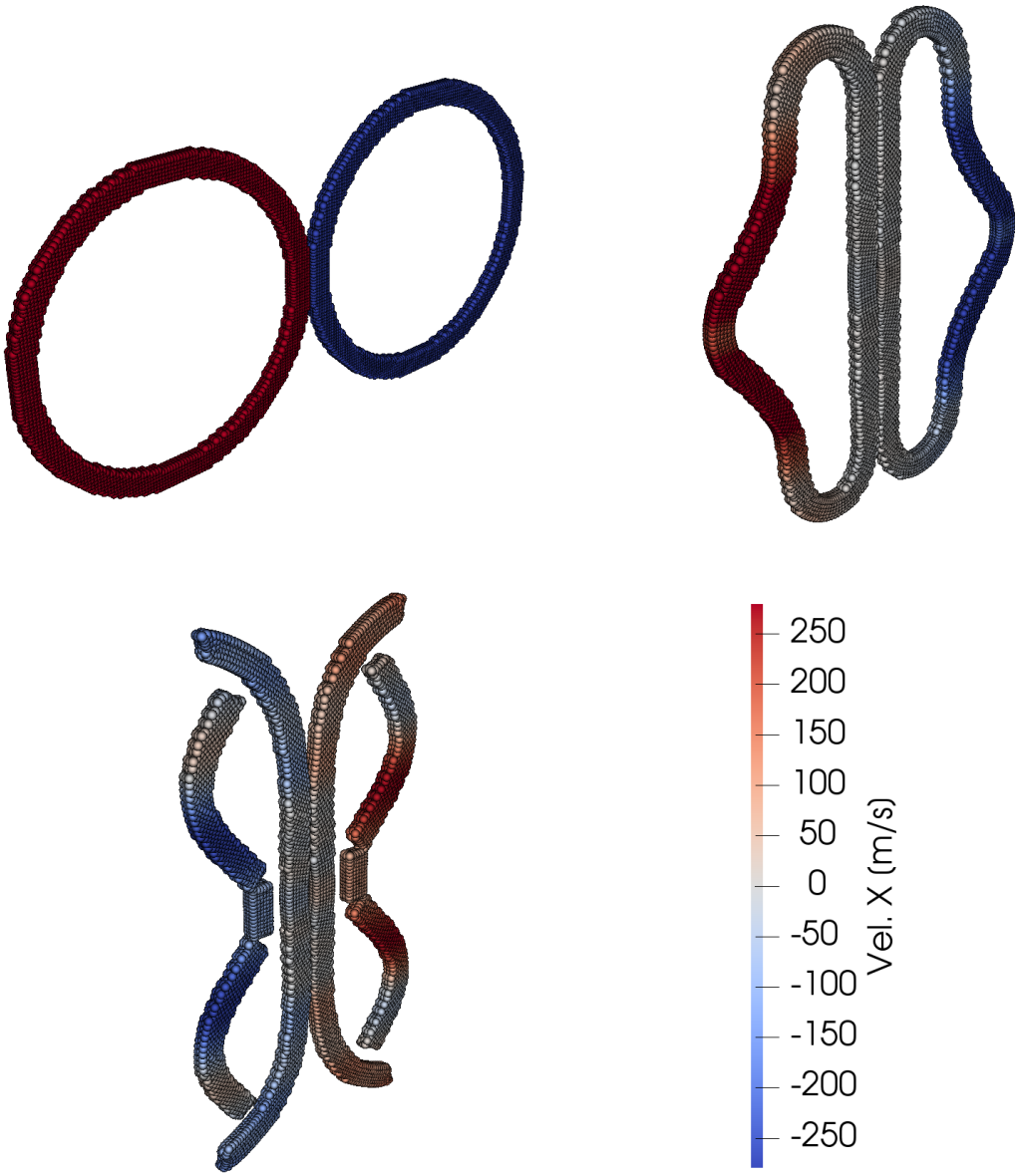

Figure 7.14: Some frames of the colliding cylinders simulation. The camera was slightly tilted to highlight the geometry of the cylinders. The simulation fails and unphysical fracture occurs.

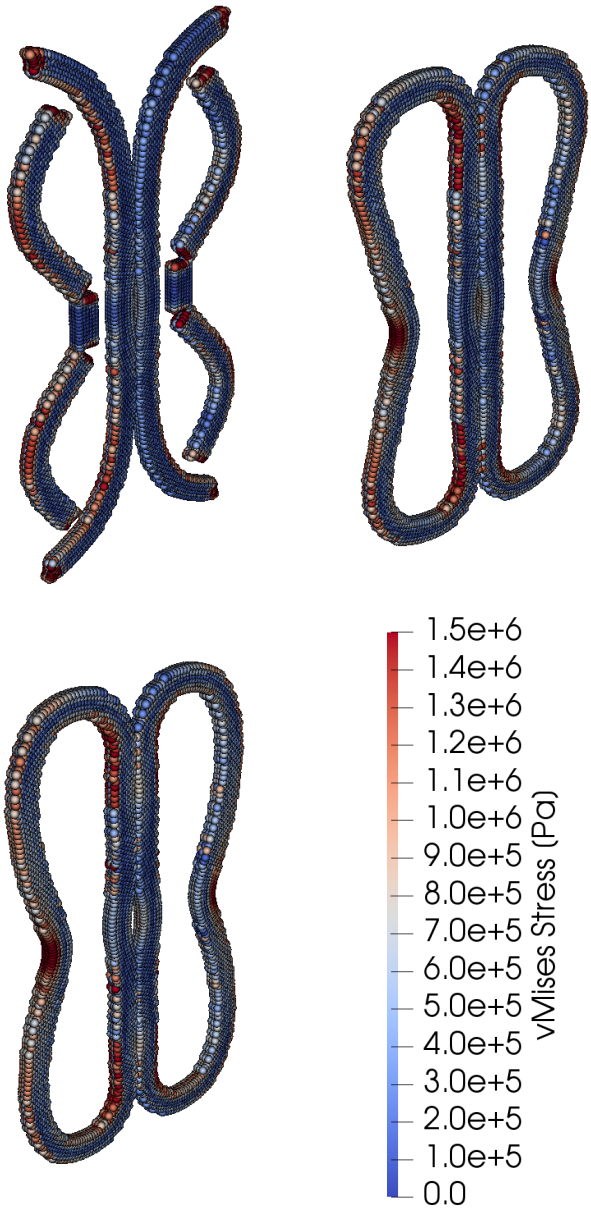

Figure 7.15: The last frame of figure 7.14 again, this time corrected by artificial stresses in the middle, and the Transport Velocity Formulation on the right. Both schemes are devoid of any unphysical fracture.

7.3 PLASTIC IMPACT

The purpose of this benchmark is to assess the adequacy of the investigated algorithms for plastic deformations in a more controlled situation than metal cutting. To this end, a moderate velocity impact of a plastic projectile on a plastic wall is considered. Both the projectile and the wall are made of steel, see 7.3, and the impact velocity is $500m/s$. This way, large plastic strains are encountered but at slower rates, and without material separation, rendering visibility criterion considerations as described in section 5.4.6 a mute point for now. A sketch of the simulation set up is given in figure 7.16.

Since the potential based algorithm was proven incompatible with the contact algorithm implemented, and it is unclear how to incorporate the plasticity algorithm into it, it was excluded from this and any further benchmarks. Additionally, both the total Lagrangian as well as the updated Lagrangian FEM implementation are too simplistic to handle the benchmark at hand, probably due to lack of any hour glassing control, see for example [78]. Consequently, the Lagrangian FEM solver in `LSDYNA` was used as a reference. As discussed in section 7.2.2, a high order time stepper is inapropriate for problems dominated by contact, since the penalty contact algorithm limits the time step more heavily than the CFL condition imposed by a high order time stepping scheme. Hence, the leap frog scheme, which is of second order, was employed in favor of the RK4 scheme used in previous benchmarks.

A number of results was collected during the simulation runs. The final frames at $t = 3 \cdot 10^{-4}$ seconds are exhibited in figure 7.17. The weak form methods as well as Godunov's method seem to exhibit quite a bit of numerical dissipation, showing lower plastic strains than the other methods. The FEM reference is quite smooth in the plastic field, while some meshless schemes, especially Monaghans algorithm and the TVF show pronounced areas of high plastic strain. Additionally, the TVF scheme resolves quite a bit of plastic straining at the top and bottom fixtures.

The horizontal extension, or, alternatively put, the position of the right most point of the wall over time was recorded in figure 7.18. This time, there is a scattering of the methods about the FEM reference. Most of the methods cluster in a band of about $+/-0.005$ meters, which seems fair given the dynamic nature of the problem at hand. A clear outlier is found in the Godunov scheme, which deforms almost one wall thickness less than the reference.

This prompts the closer inspection of final shapes of both projectile and wall. This was performed in figure 7.19 for the projectile and in figure 7.20 for the wall, respectively. Here, especially the differences in shapes of the projectile is striking, while the differences in shapes of the wall appears quite small because of the large aspect ratio of the object at hand. Numerical dissipation is again clearly visible in Godunovs method as well as the weak form schemes. Additionally, the contact surface remains flat in all weak form methods including the FEM, whereas some strong form methods, especially the algorithm by Monaghan and the TVF resolve a rough surface.

Finally, it remains to clarify that, while the FEM was chosen as reference, it can not be stated without doubt that it is the most correct one. Inspection of the mesh, see figure 7.21, reveals that mesh distortion is quite severe, even at the moderate deformations encountered in this

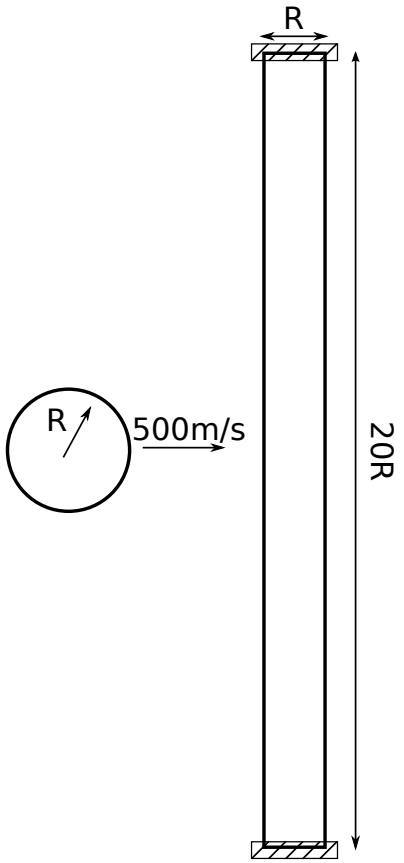

Figure 7.16: Sketch of the plastic impact benchmark, R was chosen as $R = 0.03m$. The simulation is run until $3 \cdot 10^{-4}$ seconds.

Table 7.3: Elastic, Johnson Cook and thermal material data of steel AISI 4340 according to [124].

Parameter	Value	Unit
Youngs modulus E	200e9	Pa
Poissons ratio ν	0.29	-
Density ρ	7830	kg/m^3
A	792e6	Pa
B	510e6	Pa
C	0.014	-
m	1.03	-
n	0.26	-
Reference temperature T_0	300	K
Melting temperature T_m	1793	K
Specific heat capacity c_p	477	$J/(kg\ K)$
Thermal conductivity k	50	$W/(m\ K)$

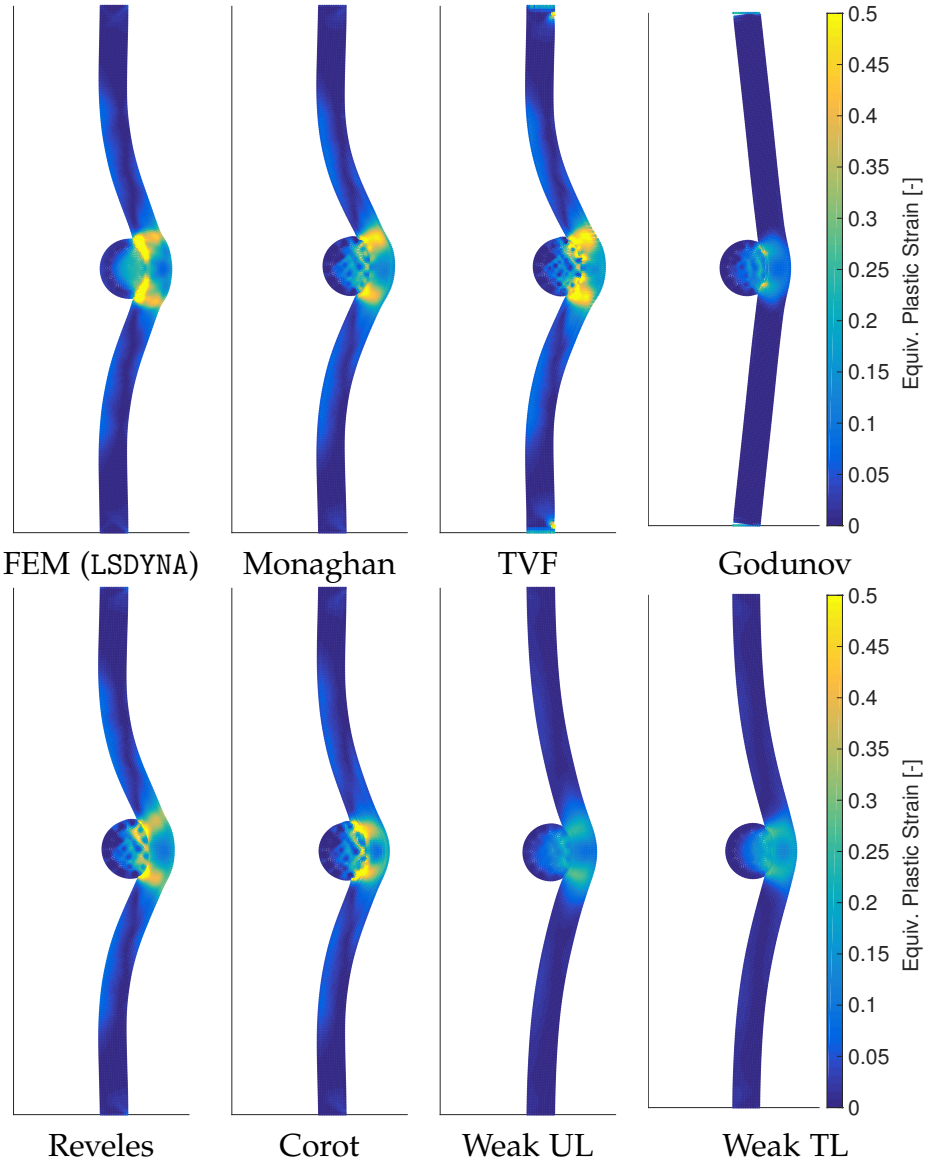

Figure 7.17: Last frames of the plastic impact simulation for all participating methods.

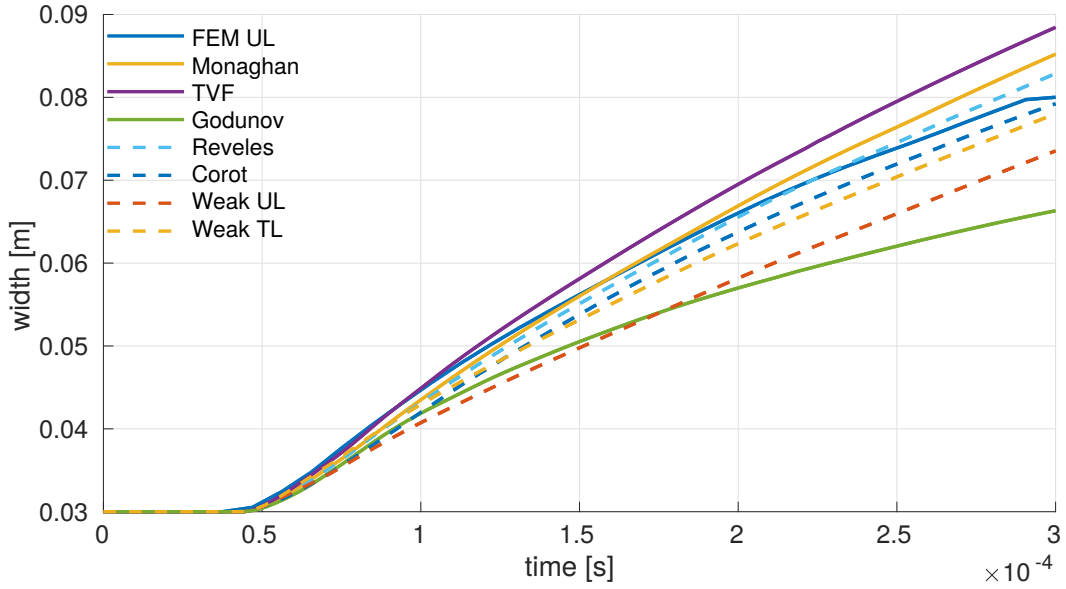

Figure 7.18: The width, or rather position of the right most point in the wall over time.

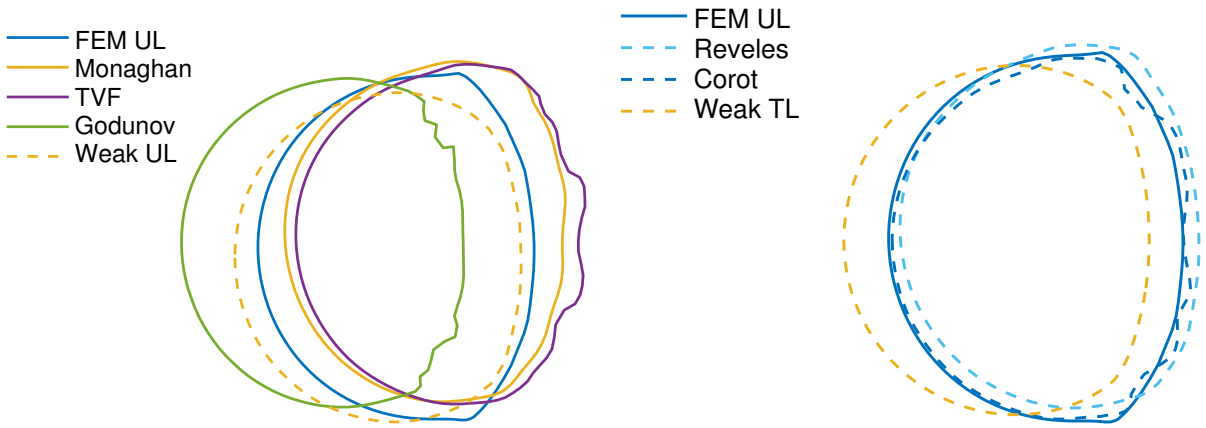

Figure 7.19: Final shapes of the projectile in all participating methods. Updated Lagrangian methods on the left, total Lagrangian methods on the right.

example. The figure exhibtis both mesh penetration in the contact zone as well as some quads with quite bad aspect ratio.

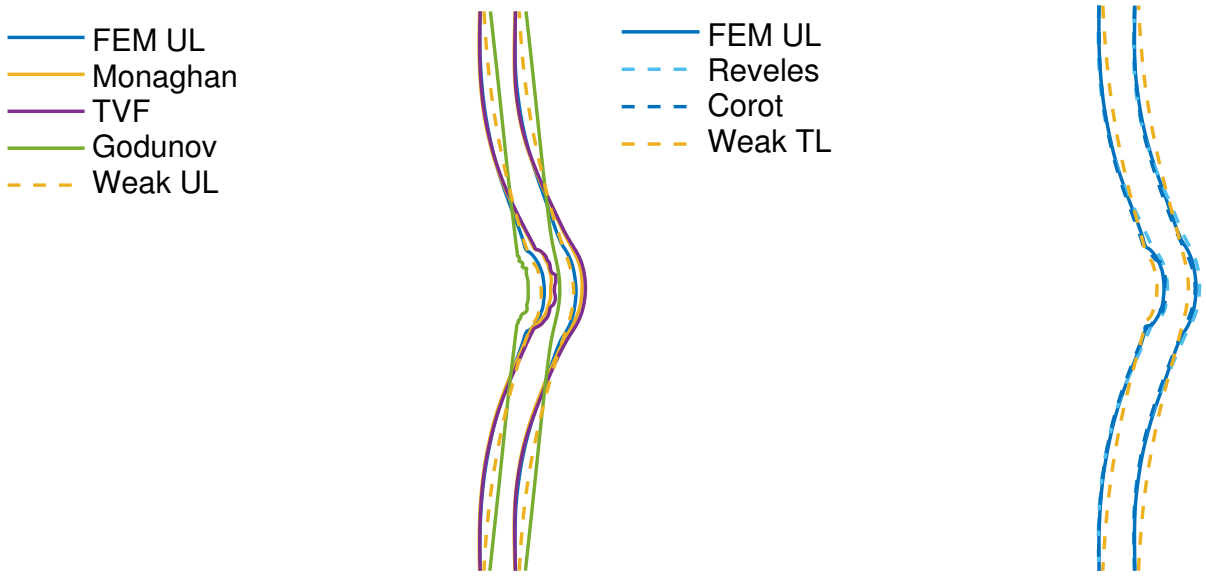

Figure 7.20: Final shapes of the wall in all participating methods. Updated Lagrangian methods on the left, total Lagrangian methods on the right.

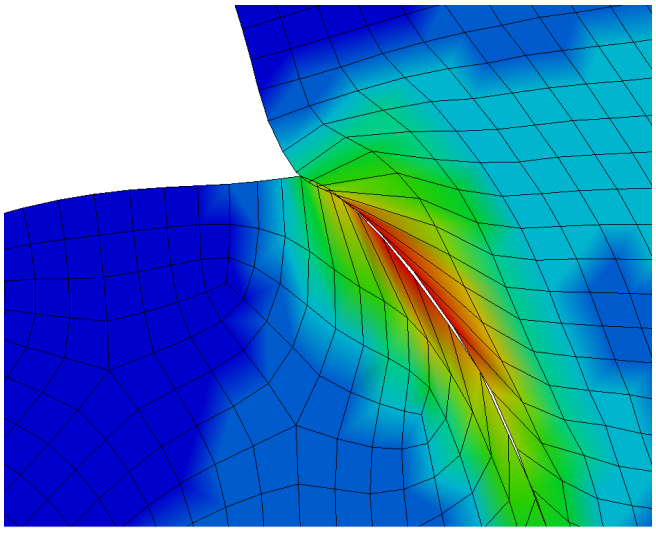

Figure 7.21: A close-up of the mesh in the plastic impact simulation as simulated by LS-DYNA.

7.4 CONCLUSIONS OF PRELIMINARY BENCHMARKS

Before the algorithms at hand are used for their intended purpose with respect to this thesis, some preliminary remarks are made.

- The potential based algorithm, i.e. algorithm 5, is not compatible with the penalty contact algorithm pursued in this work. It is unclear how to incorporate the Johnson-Cook plasticity model into it. It is thus excluded from the rest of this thesis
- High order time integration schemes are only useful in the absence of the penalty contact algorithm, i.e. if there is no contact or contact is handled by particle interactions
- There is no algorithm handling all benchmarks equally well. For example Reveles' algorithm becomes unstable in a simple rigid body mode test, but handles all order situations remarkably well.
- One of the distinguishing features of the algorithms is the amount of numerical dissipation introduced. The weak form algorithms, especially in the updated Lagrangian frame, and most of all, Godunovs algorithm seem to introduce excessive numerical dissipation.

Since no algorithm except for one can be excluded at this stage, and there is no algorithm handling all situations inspected equally well, all algorithms except the potential based one are investigated in orthogonal cutting simulations.

7.5 ORTHOGONAL METAL CUTTING

The orthogonal turning operation is the only metal cutting operation that can be reasonably approximated using 2 dimensional geometry. The two dimensional approximation is still rough in the sense that in the workpiece the plane strain condition is appropriate, whereas in the chip the plane stress condition would be more appropriate. Despite this, the plane strain condition is assumed for the whole work piece, just as in all other two dimensional benchmarks.

The material considered is Ti6Al4V, see 7.5. The basic set up of the benchmark is exhibited in figure 7.23, with parameters chosen according to table 7.4. The workpiece would actually be cylindrical, but can be assumed straight for the short cutting distances considered. The depth of the work piece D is much larger in real life, but was chosen heuristically such that the plastic strains become negligible at the boundary. The friction parameter was chosen in accordance to [225] as 0.35.

The tool is assumed rigid and adiabatic. Thermal conduction is considered in the work piece, but no heat flow from the workpiece into the tool. For more involved thermal modeling, consider section 7.5.3.1. The focus of the simulation was to retrieve the cutting forces. This can be achieved with quite low particle resolution. About 6000 particles were used for each

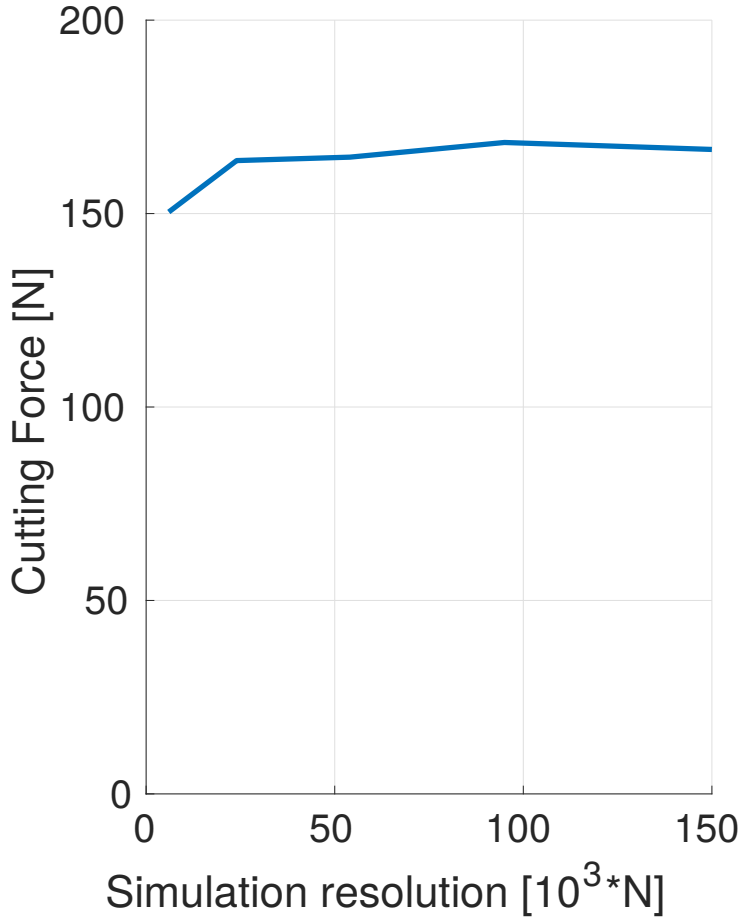

Figure 7.22: Cutting forces are plotted against simulation resolution. Simulation resolution is in thousands of particles. As can be seen, as soon as a certain, quite modest, simulation resolution is reached, the forces do not change anymore.

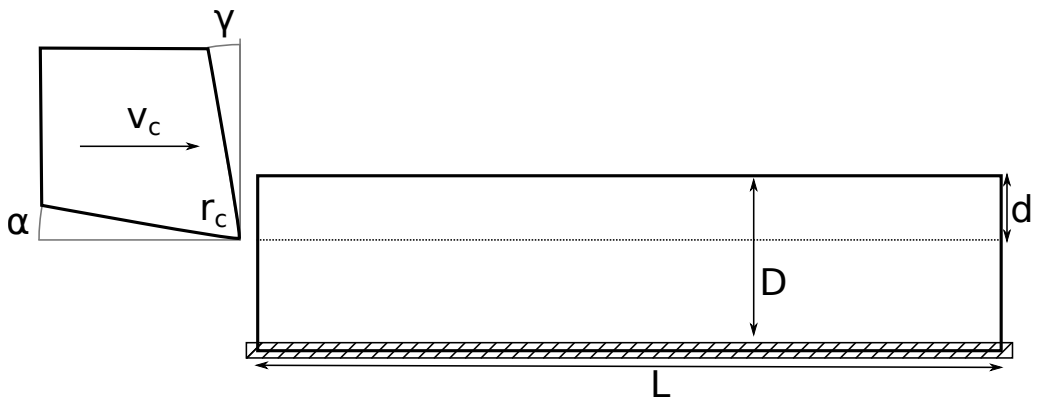

Figure 7.23: Sketch of the orthogonal cutting benchmark. The sketch is presented completely parametrized. For explanation of each parameter and values used see table 7.4.

Table 7.4: Cutting parameters used

Parameter	Explanation	Value	Unit
γ	Rake angle	10	Degrees
α	Clearance angle	8	Degrees
r_c	Cutting edge radius	$10 - 50$	μm
v_c	Cutting velocity	70	m/min
d	Uncut chip thickness	$1 \cdot 10^{-4}$	m
D	Height of workpiece	$3 \cdot 10^{-4}$	m

simulation. Increasing the resolution does not significantly alter the cutting forces, see figure 7.22. For more high resolution simulations consider 7.5.3.2.

Verification of the results is done against the experiments of Wyen [279], where the cutting edge radius was varied at various cutting depths. Cutting depth was fixed for these studies at $d = 0.1$ mm, but the same range of cutting edge radii was investigated.

7.5.1 *Updated Lagrangian Methods*

In this section, updated Lagrangian methods are considered. Updated Lagrangian methods are the more natural choice for metal cutting, or any situation where new surfaces are generated for that matter. This is because they handle topographical chances by design, see 7.5.2 for further elaboration. After mentioning some adaptions that had to be made to the TVF in order to make the algorithm fit for metal cutting, results are presented in depth.

7.5.1.1 *Adapting the TVF for cutting*

Some alterations to the original TVF publication for solid mechanics [282] had to be made in order to make the scheme suitable for metal cutting simulations. It was observed that

Table 7.5: Ti6Al4V

Parameter	Value	Unit
Youngs modulus E	$110 \cdot 10^9$	Pa
Poissons ratio ν	0.35	-
Density ρ	4430	kg/m^3
A	$862 \cdot 10^6$	Pa
B	$331 \cdot 10^6$	Pa
C	0.01	-
m	0.8	-
n	0.34	-
Reference temperature T_0	300	K
Melting temperature T_m	1836	K
Specific heat capacity c_p	562	$J/(kg\ K)$
Thermal conductivity k	6.8	$W/(m\ K)$

the scheme reacts very sensitively to newly generated surfaces, i.e. very high transport velocities occur in areas of the cutting edge. This is expected, the transport velocity depends on the lack of zero-order completeness of the most simple SPH gradient approximator (4.13), hence a violent change in the boundary will lead to an equally violent increase in transport velocities.

This oscillations lead to unstable simulations. It was found it is best to set the transport velocity to zero at the boundary. The boundary can be reliably identified using a simplified version of the scheme presented in [158]. This scheme performs an Eigenvalue analysis of the correction tensor suggested by Randles-Libersky, see [214] and [143], as well as section 4.3.2:

$$\underline{\underline{B}} = \left[\sum_{j=1}^{N} (\underline{x}_i - \underline{x}_j) \otimes \nabla W_h(\underline{x}_i - \underline{x}_j)\ \omega_j \right]^{-1} \tag{7.5}$$

Particles with minimum Eigenvalue $\lambda^{\min}$ larger than some constant $c < 1$ are flagged as boundary particle, and their transport velocity is nulled. The choice of c is not overly important, all simulations carried out were run with $c = 0.875$.

7.5.1.2 *Results*

Even though the focus of the experimental study was the retrieval of the cutting forces, it is still worthwile to consider the chip shape, even at the low simulation resolutions that were used for this study. Unfortunately, the experimental study [279] did not record chip shapes for various cutting conditions. Still, one close up of a chip produced in cutting conditions equivalent to the simulations at hand is given and reproduced here in figure 7.24. As can be seen, a serrated chip is formed with a spacing of 50μ between the segments. No information

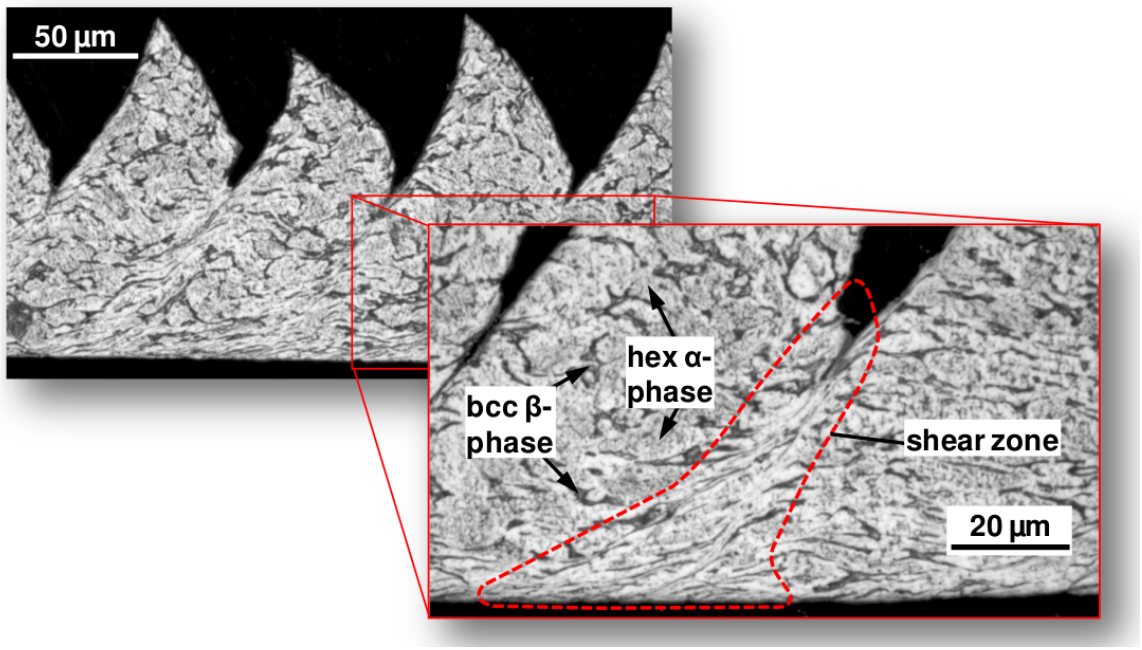

Figure 7.24: Close up of a chip from the experimental study, taken from [279]

with regard to chip curling is given. Other experimental works, for example [280] suggest that chip curling is present. A quantitative assessment can not be done however, since the cutting parameters are different from the ones used here.

Chips from the simulation runs are given in figure 7.25. The Godunov type algorithm resolves a large crack in from of the tool tip. This crack grows up to the bottom of the workpiece, conflicting with the fixed boundary condition and the bottom on the simulation terminates. It is thus concluded that Godunovs algorithm is not appropriate for cutting. The updated Lagrangian weak form scheme shows some escaped particles at the chip root and some noise in the particle distribution on the side of the chip away from the rake face. However, all simulation runs terminate.

The different algorithms all produce different characteristics of the chip. From very little chip curling in Monaghans algorithm, to some more chip curling in the Transport Velocity scheme to a almost turned over chip in the updated Lagrangian algorithm. The curling in the updated Lagrangian algorithm is so pronounced that it crashes back into the work piece if the simulation is continued.

All simulations produce a serrated chip in some form, although in the updated Lagrangian scheme the serration is not or hardly visible in the particle distribution but expressed in the plastic strain field only. Monaghans algorithm shows what can be described as notches in place of the segments, whereas the TVF scheme clearly shows a segmented chip. A quantitative study of the chip shape at these low resolutions is questionable at best but the segments are surprisingly close to 50μ in the TVF scheme and the updated Lagrangian weak form algorithm, while Monaghans algorithm at least resolves the correct order of magnitude in the lowest segment, see figure 7.26. Reproduction of curled and segmented chips is a clear advantage over LSDYNA, which presents a straight flow chip according to [225] in this situation.

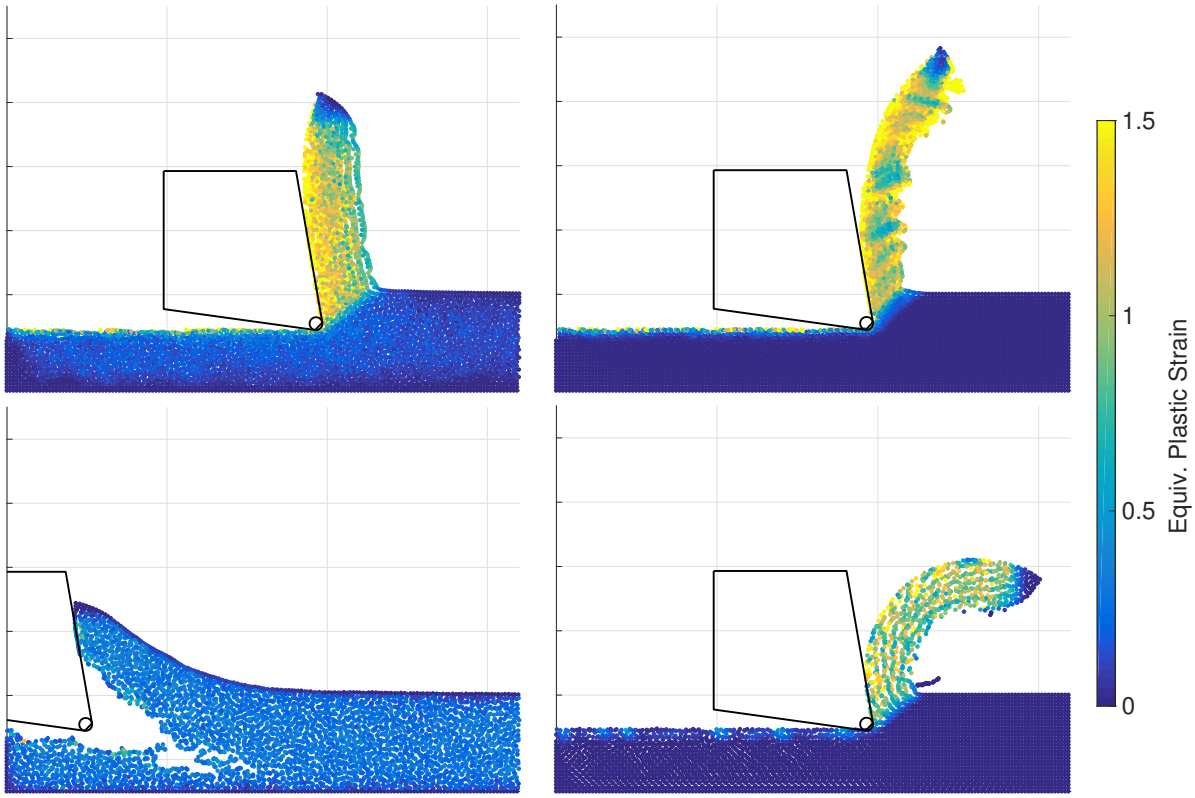

Figure 7.25: Result frames from the simulation run showing the chip shape. Monaghans Algorithm, TVF scheme, Godunov Type SPH and updated Lagrangian weak from, left to right, top to bottom. Color is plastic strain, limited at 120%.

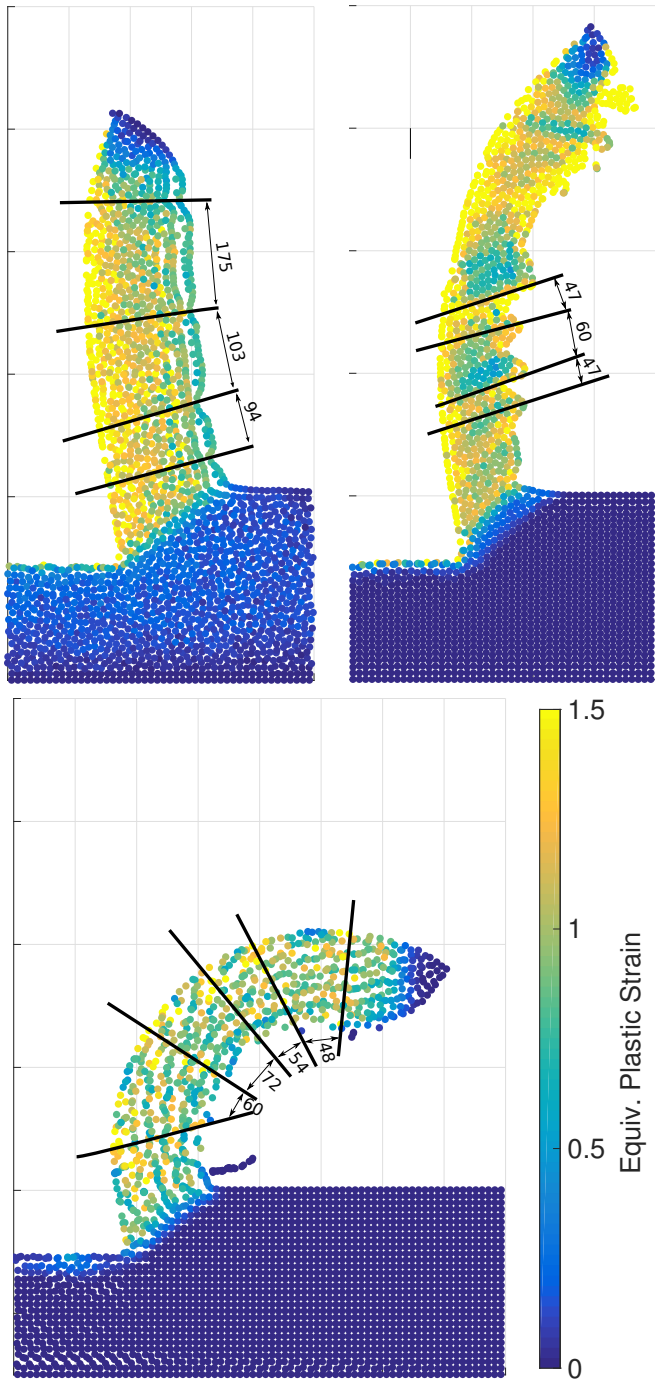

Figure 7.26: Measuring the chip shape. Monaghans algorithm and TVF on top, left and right, respectively, and the updated Lagrangian weak form on the bottom. Numbers are in μm. Distances where measured at the intersection points of the black lines and the back of the chip.

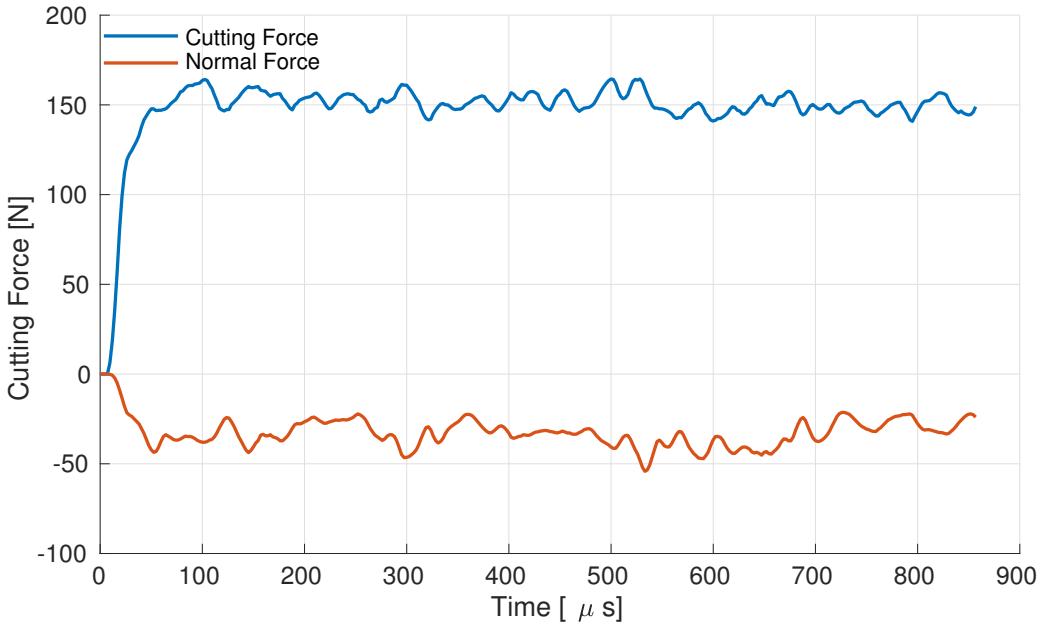

Figure 7.27: Force history of a typical cutting simulation as presented in this text. The noise is due to the low simulation resolution in conjunction with the penalty contact algorithm 5.4.

What can clearly be quantitized are the cutting forces. As discussed, the cutting edge radius was varied from $10\mu m$ to $50\mu m$. Forces reach steady state quite quickly, as can be seen from the exemplary force history given in 7.27. For the comparison with the experiment, forces were averaged in the second half of the simulation run.

Figures 7.28 show cutting- and normal forces. Normal forces are also denoted "thrust-" or "passive force" in other works. The first thing to note is that the reproduction of the normal force seems very bad. This is a general trend in metal cutting simulations and not limited to meshless methods, as can be seen from [119]. In [27] it is demonstrated that both commercial general purpose FEM solvers and special purpose solvers for metal cutting like `AdvantEdge` and `DEFORM` may even resolve normal forces that point into the wrong direction in some cases.

Compared to the experiments by Wyen [279], the cutting forces are predicted with errors ranging from 10-20% in the case of Moanghans algorithm. This is well in line with previously reported results in the literature using the SPH formulations available in `LSDYNA`. The other methods resolve unsatisfactory errors of up to 40%. This is expected for the updated Lagrangian weak form algorithm since it displayed excessive numerical dissipation in all preliminary simulations but quite surprising for the TVF scheme, for which the opposite is true.

7.5.1.3 *Results - Improving the Cutting Force*

It has to be noted that the force can be tuned in almost linear fashion by tuning the friction parameter, at least for the strong form methods. The friction parameter used was

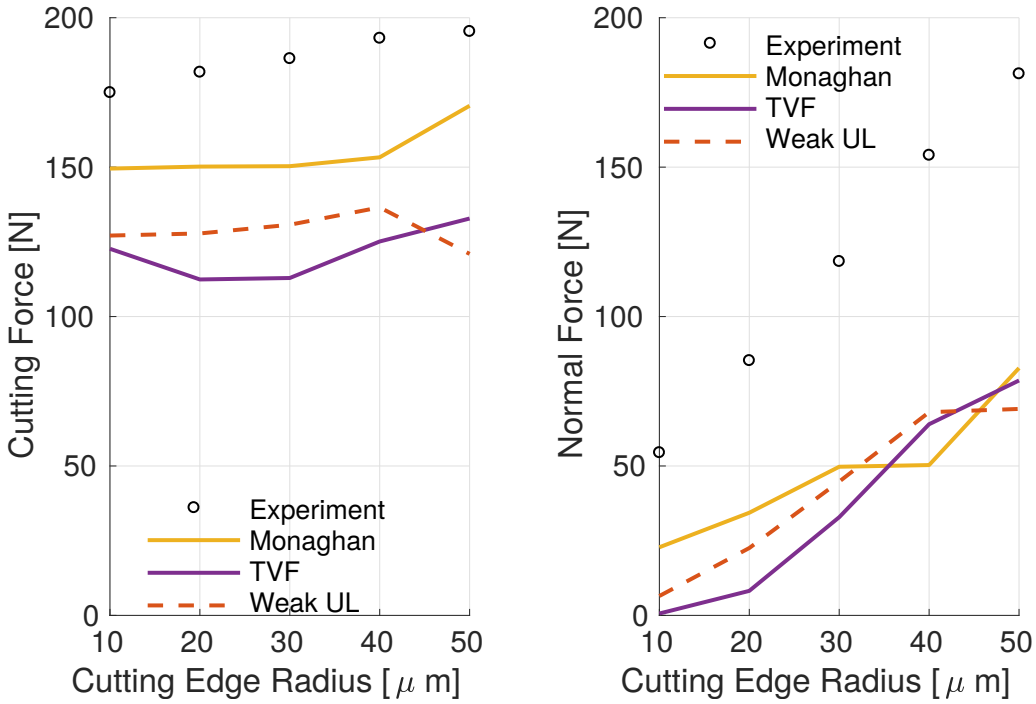

Figure 7.28: Cutting and normal forces for all updated Lagrangian methods. The normal force on the tool points upwards in figure 7.23.

demonstrated with a pin on disk test. Its validity in a metal cutting situation is certainly questionable, justifying refitting the friction parameter to one simulation, then using it under variation of the cutting edge radius. To this end, the friction parameter was allowed to grow from 0.35 up until 0.7, see figure 7.29.

As claimed, the scaling of the process forces is almost linear in the friction parameter for the strong form methods. Slight oscillations occur, but the trend is clear. It is speculated that these oscillations stem from the impact on the contact length due to the friction parameter. The updated Lagrangian weak form algorithm is insensitive to the choice of friction parameter. It is speculated that this is a consequence of the very short contact length of the chip geometry resolved by this algorithm.

For Monaghans algorithm, a optimum friction parameter is observed somewhere between 0.5 and 0.6, which is certainly reasonable. Simulations were re-run with friction parameter of 0.6, yielding significantly more accurate cutting forces, as can be seen in figure 7.31.

For the TVF, the force grows noticeably smaller with increasing friction parameter and the experimental value is not reached, even at 0.7. This raises some concerns, since the frictional force on the interface a material can support is limited. The LSDYNA theory manual [99] suggests to limit the friction force in the following fashion:

$$|\underline{f}^{\text{lim}}| = min(|\underline{f}^{\text{fric}}|, \kappa^{\text{fric}} \cdot A_m \cdot \sigma^0_{yield}/\sqrt{3}) \tag{7.6}$$

Where $|\underline{f}^{\text{fric}}|$ is computed using the procedure outlined in section 5.4.7, κ is a tuning parameter, A_m the area of the master segment in contact and σ^0_{yield} the static yield limit in

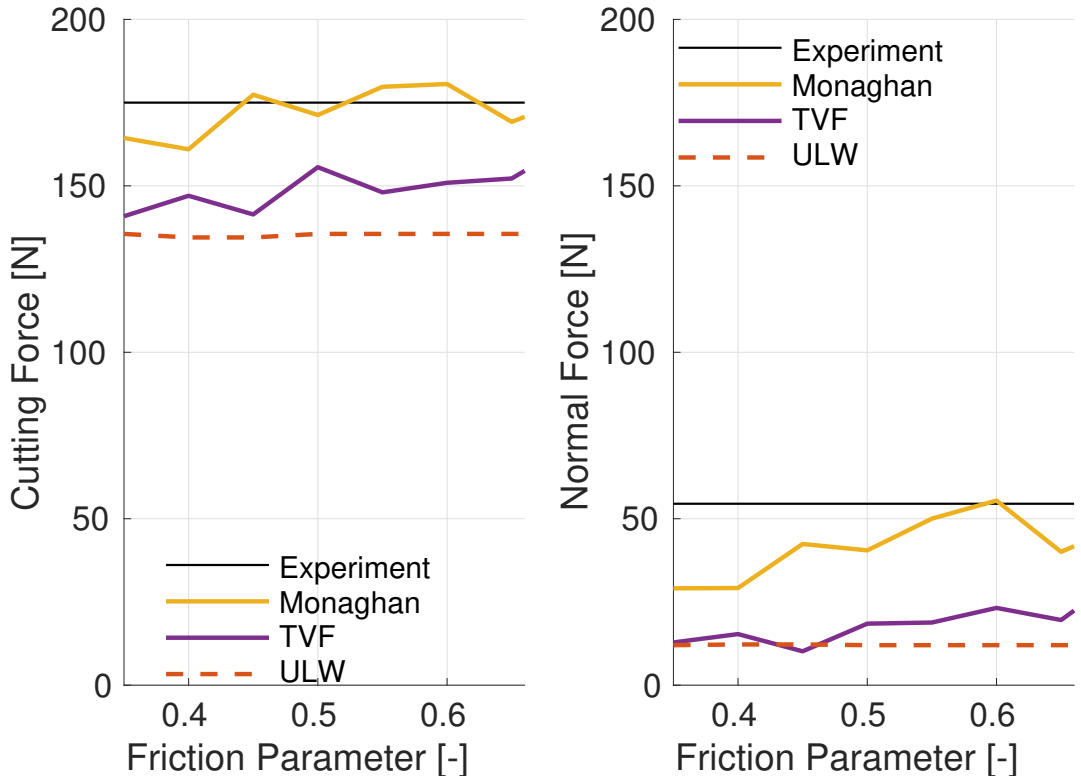

Figure 7.29: Cutting and normal force under variation of the friction parameter for the updated Lagrangian algorithms.

tension, making $\sigma^0_{yield}/\sqrt{3}$ the static yield limit in shear. This is parameter A in the Johnson-Cook model 3.4.3. One implementation detail to consider is that in the implementation at hand, the area of the master segment is not well defined since the tool is parametric. Hence, A_m was changed to A_s, i.e. the area of the slave segment. This should not be a major issue, since the contact algorithms for FEM are usually only valid for similarly fine meshes anyway. Limiting the friction in this way is quite rough, in particular, the *current* yield limit should be used. However, it can still be used to give a first estimate if the material can actually support the frictional forced imposed by the friction / contact algorithm implemented. Note that all of this assumes that no chemical processes happen at the tool tip that would induce brazing of the material to the tool, which would enable virtually arbitrary large friction coefficients.

To check whether it is prudent to further increase the friction parameter in the situation of the TVF algorithm, this limiting factor was imposed under friction parameter 0.7 and $\kappa^{\text{fric}} = 1$, see figure 7.32. As can be seen, limiting the friction parameter limits the cutting force, suggesting further increasing the friction force is unphysical.

To assess whether using σ^0_{yield} instead of σ_{yield} in (7.6), the history of the yield limit in two particles in the cutting zone is investigated in figure 7.33. As can be seen, the yield limit is affected quite severely during deformation. In particular, it is not even clear if κ^{fric} should be chosen smaller or greater than 1 in (7.6) to address this behavior, since thermal softening may be strong enough to dominate the hardening.

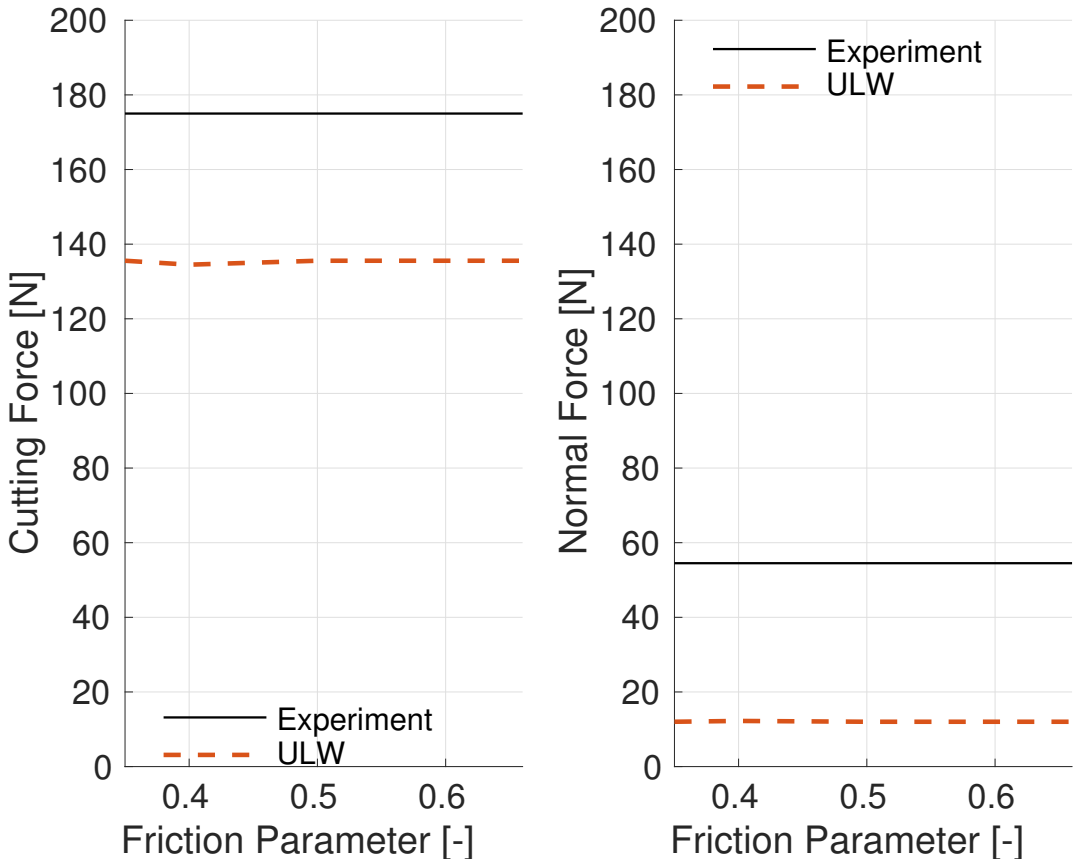

Figure 7.30: Cutting and normal force under variation of the friction parameter for the updated Lagrangian weak form algorithm.

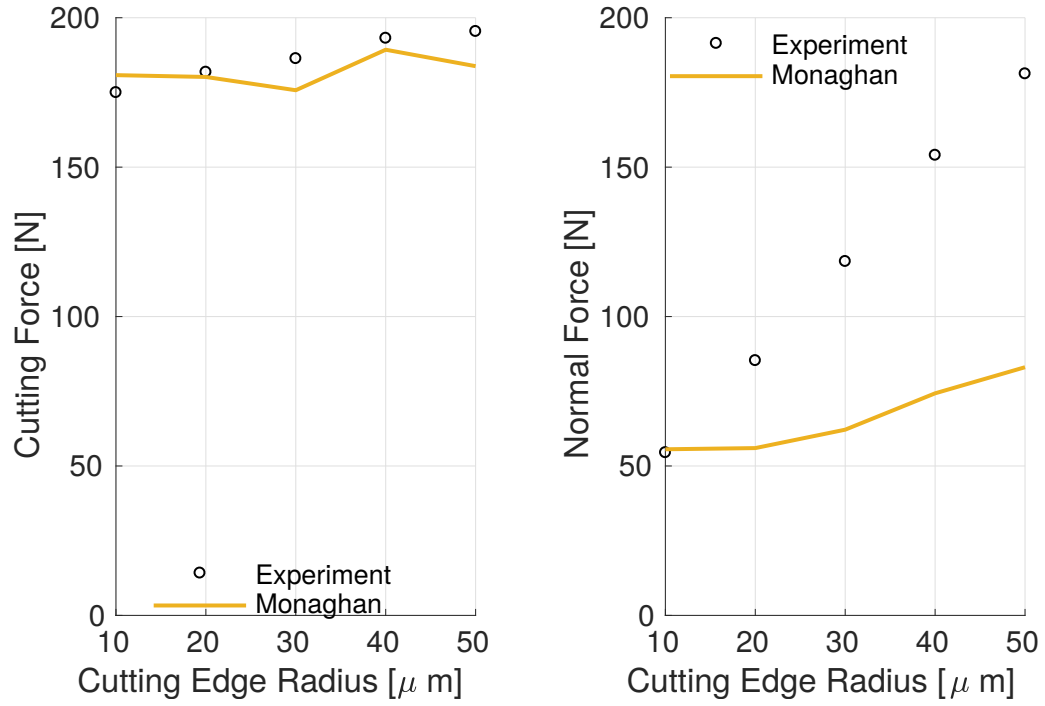

Figure 7.31: Cutting and normal force under variation of the cutting edge radius for Monaghans algorithm using the re-fit friction parameter.

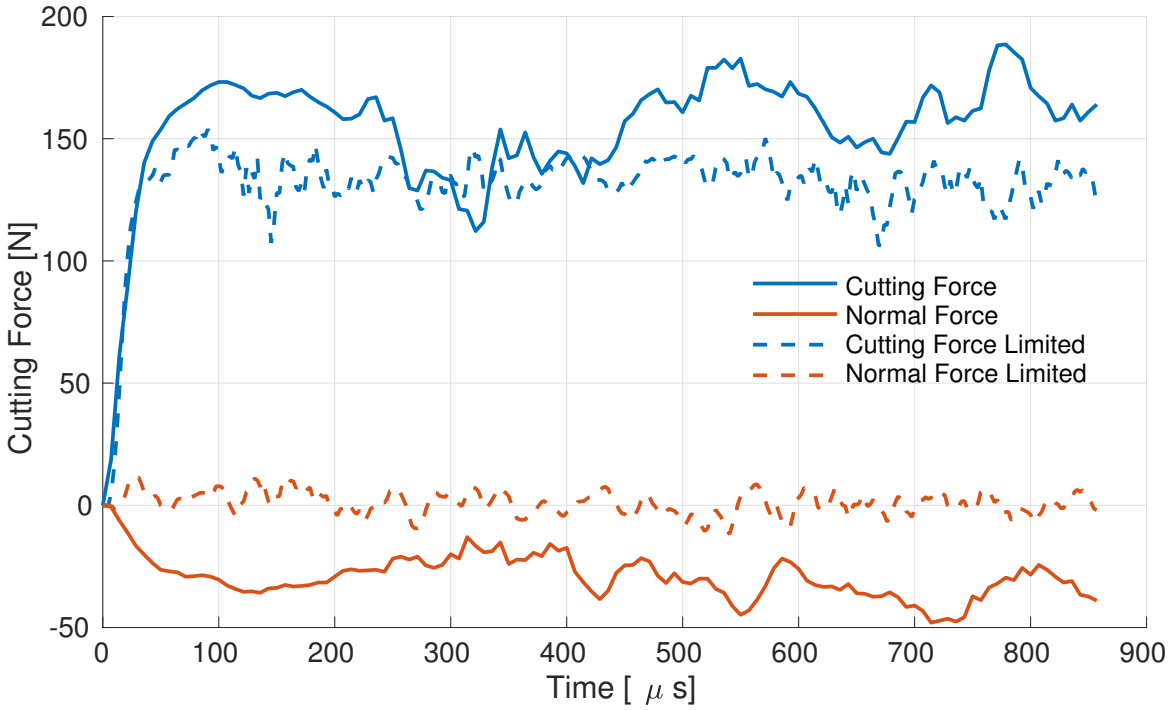

Figure 7.32: Force history of the TVF algorithm with high friction coefficient, both limited by (7.6) and unlimited.

Thus, the simulation was run again, but this time using:

$$|\underline{f}^{\text{lim}}| = min(|\underline{f}^{\text{fric}}|, \kappa \cdot A_m \cdot \sigma_{yield}/\sqrt{3}) \tag{7.7}$$

The results are given in figure 7.34. It is observed that using the limitation by the current yield limit allows for slightly higher cutting forces. The mean increases from 131 Newton, using (7.6), to 141 Newton using (7.7). This is still shy from the observed 163 Newton when not limiting the friction force. It is thus concluded that fitting the friction force is not a suitable means to correct the cutting forces in the case of the TVF.

Another way to improve the prediction of cutting forces is by giving up conservation of linear momentum but enabling the scaling of the smoothing length h by a simple scheme:

$$\dot{h} = h \cdot \nabla \underline{v} \tag{7.8}$$

This scheme can be shown to be equivalent to the scheme called "Density - Continuity" in section 4.4, it is restated here in more compact notation for convenience. As can be seen in figure 7.35, results are improved for both Monaghans algorithm and the TVF scheme. The updated Lagrangian weak form scheme is absent from the graph since it showed to be extremely sensitive in the choice of h, rendering it unstable using (7.8).

While the global and local errors decrease significantly, the general trend that forces increase with enlarged cutting edge radius is not recovered anymore. In [156] it is noted that the chip shape is highly sensitive to the smoothing length chosen. While adapting the smoothing length did not change the chip characteristic, e.g. from a serrated to a flow chip as observed in [156], the spacing between the segments was indeed affected, especially in the TVF scheme, see figure 7.36.

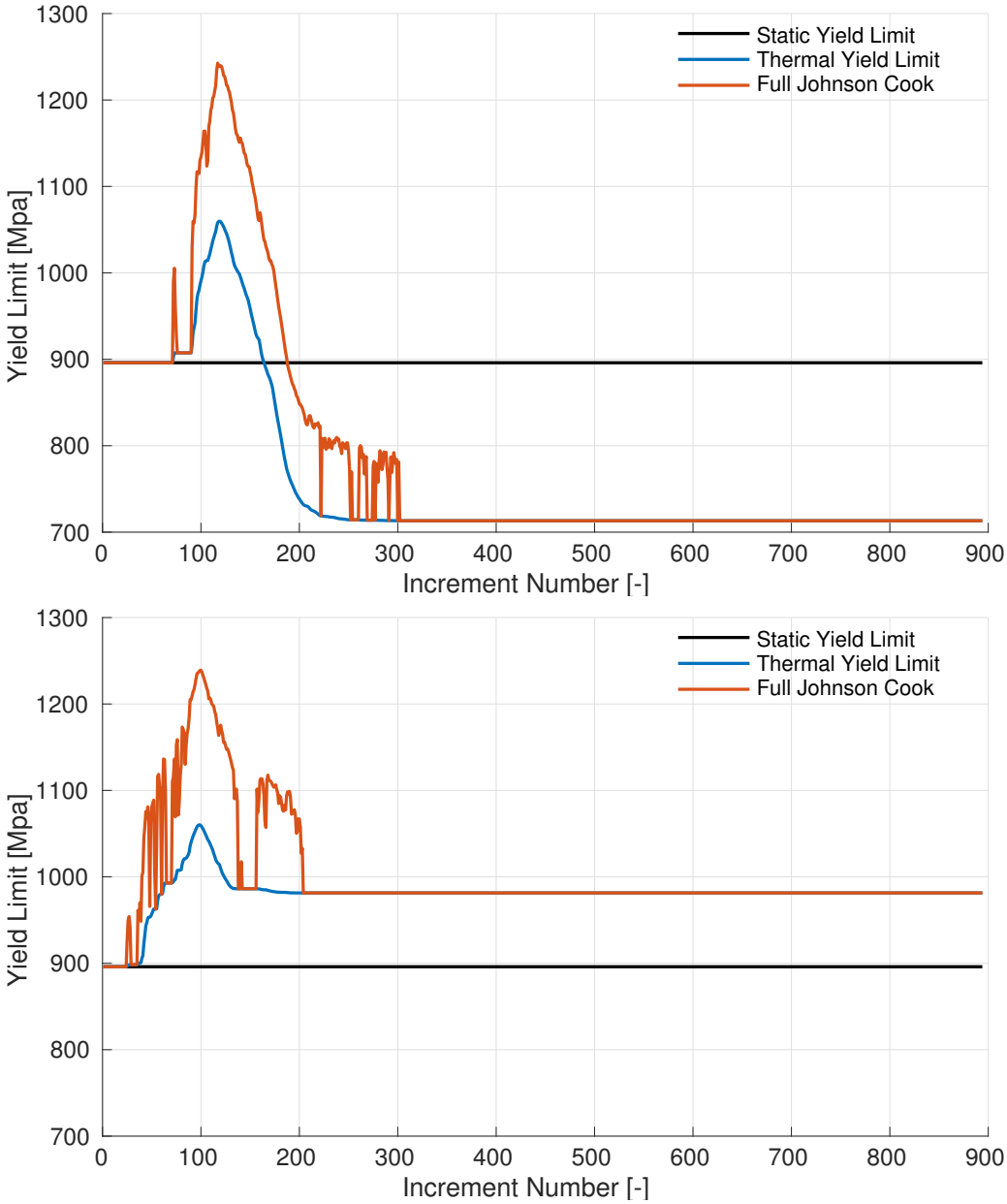

Figure 7.33: Development of the yield limit for two select particles in the contact zone. "Thermal Yield Limit" refers to the Johnson Cook model without strain rate dependent term. Simulations were run with parameters due to [162], [265].

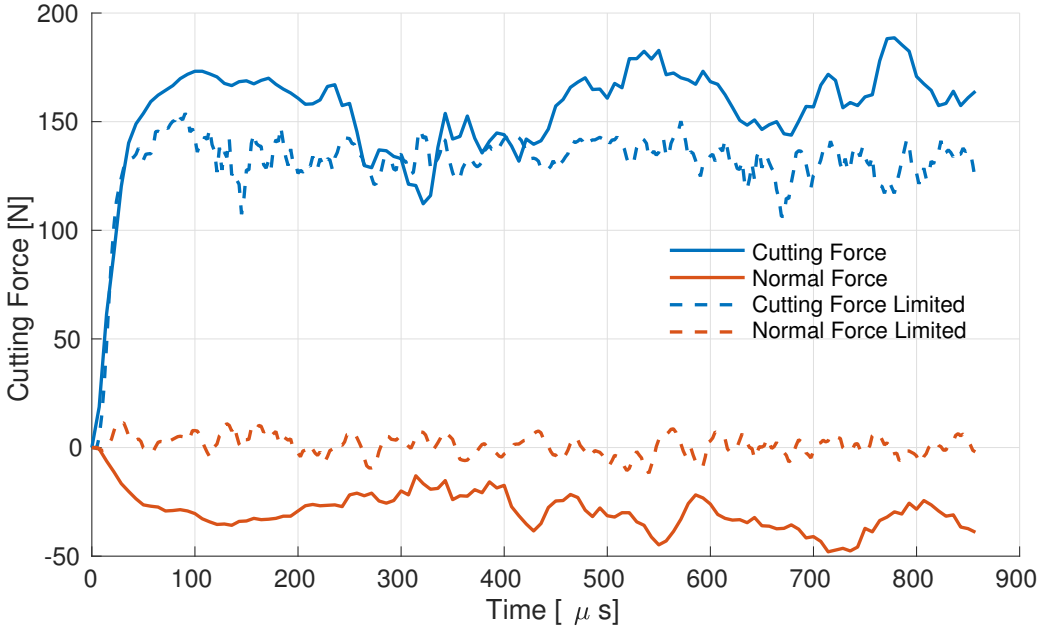

Figure 7.34: Force history of the TVF algorithm with high friction coefficient, both limited by (7.7) and unlimited.

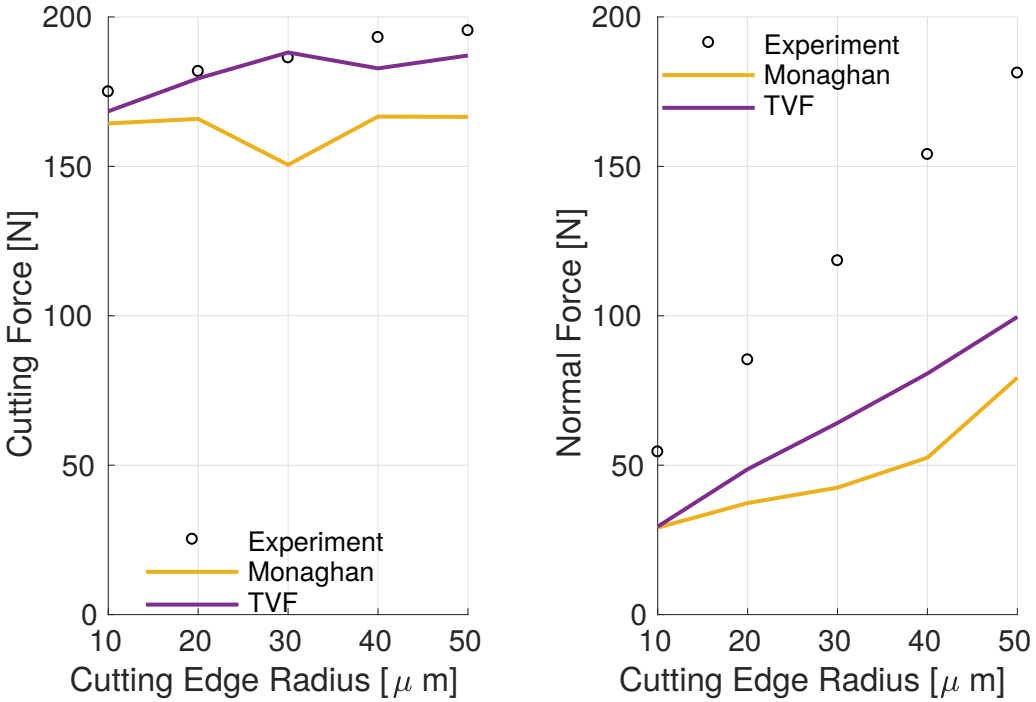

Figure 7.35: Cutting and normal forces for Monaghans algorithm and the TVF, smoothing length is adapted using (7.8). The normal force on the tool points upwards in figure 7.23.

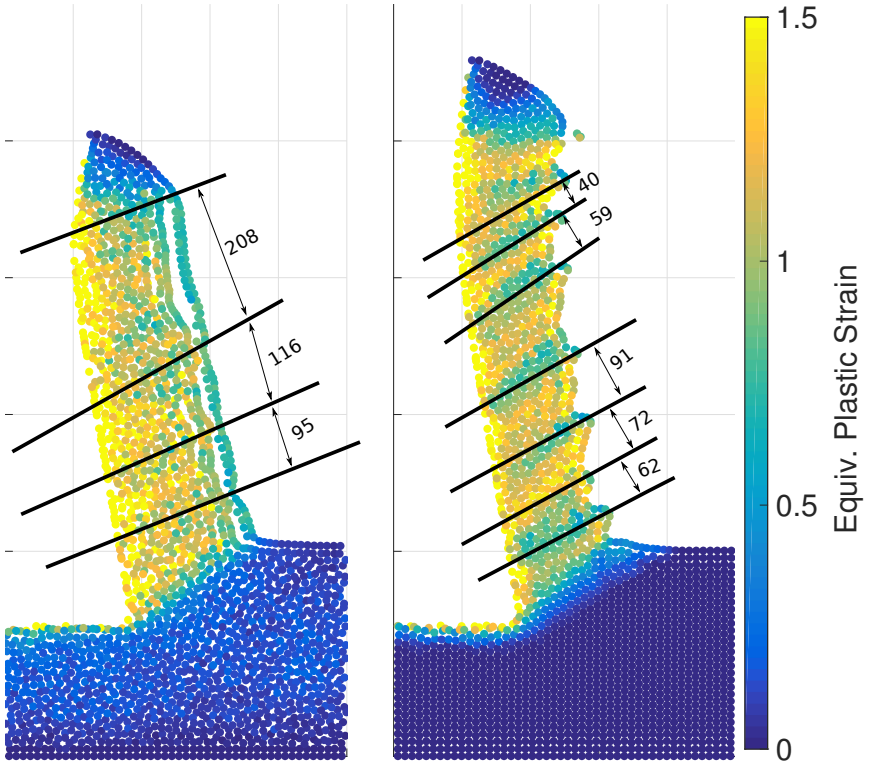

Figure 7.36: Measuring the chip shape, all simulations run with adapting smoothing length. Monaghans algorithm on the left and the TVF on the right, respectively. Numbers are in μm. Distances were measured at the intersection points of the black lines and the back of the chip.

7.5.1.4 *Results - Smooth Contact*

In section 5.4.5 a smooth contact algorithm was described. The idea of the algorithm is quickly re-summarized: repulsive particles are seeded on the tool surface, featuring a special kernel inspired by the Lennard-Jones potential. Instead of immediately adding a very large force as soon as a particle penetrates the tool, particles start to be pushed with gradually increasing intensity away from the blade as it comes closer. The algorithm can hence be considered "smooth".

In this section, this smooth contact approach is introduced to Monaghans algorithm. Since this algorithm starts acting on the particles before the tool reaches their mass center, the problem illustrated in figure 5.19, i.e. spurious interaction around the tool tip even in an updated Lagrangian simulation is expected to be alleviated. This can also be seen in example result frame 7.37 and especially in the closeups exhibited in figure 7.38.

The algorithm leaves quite a few parameter up to the user. First, there is the contact stiffness κ. A good contact algorithm is not very sensitive to the choice of κ, at least as long as κ has the right order of magnitude. In sentinel simulations it was determined that $\kappa = 1 \cdot 10^{-3}$ leads to a stable simulation for the conditions outlined in the introduction of this section 7.5.

To ensure that the choice of κ does not influence the resulting cutting forces some trial simulations were performed. Three runs with $\kappa = c \cdot 1 \cdot 10^{-3}$ and $c = 0.5$, $c = 1.0$, $c = 1.5$ were run. The force history for the cutting force is shown in figure 7.39. As can be seen, the result is largely insensitive to the choice of κ. Even though κ was tripled, the forces barely change 8% in the mean over the steady state.

The other parameters left for tuning are how densely the tool surface should be sampled and what smoothing length is to be chosen for the tool particles. To this end a parameter study was conducted. The necessity of these parameters reveals a direct concern with the algorithm at hand; since the algorithm is smooth it does not represent the tool geometry exactly anymore. Hence, the advantage of the algorithm directly creates a new predicament. Thus, a parameter study is conducted with regard to these variables.

Both the sampling density and the smoothing length are chosen in relation to the smoothing length and sampling density of the workpiece particles, which is left constant for this parameter study. The results are summarized in figure 7.40 for cutting and normal forces. As can be seen, the force is highly dependent on these two factors. The sampling density $\Delta\bar{x}$ and smoothing length $\bar{h}$ of the repulsive particles was varied from from 0.5 to 1.5 times the sampling density / smoothing length of the workpiece particles in increments of 0.25. Cutting and normal force decrease drastically when particles are seeded more sparsely or the smoothing length is increased.

Furthermore, the choice of both factors, but especially the smoothing length of the repulsive particles has a dramatic effect on the chip shape, as is demonstrated in figure 7.41. While the chip shapes produced by large smoothing lengths of the repulsive particles are clearly nonsensical, heavy chip curling can be achieved by setting $\bar{h} = h$, see figure 7.42. Still, since it is not known what radius the chip curling takes and the forces are under estimated at

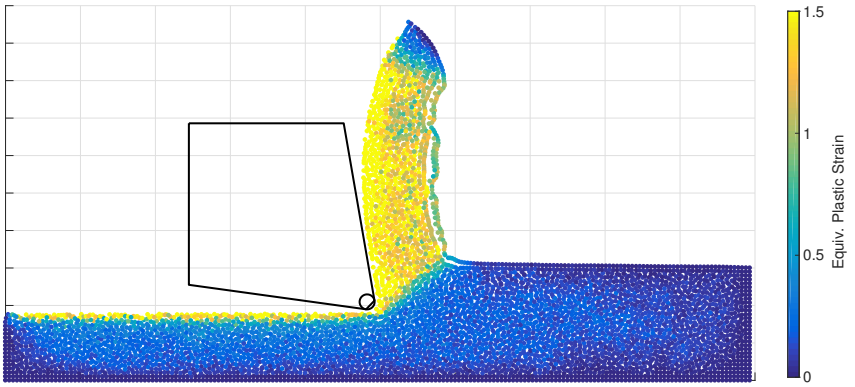

Figure 7.37: Metal cutting simulation employing the smooth contact algorithm discussed. This simulation is equivalent to the top left frame of figure 7.25 except for smooth contact. The chip is slightly more curled and less compressed. A small gap between the tool can be seen if observed closely. A close up is given in figure 7.38.

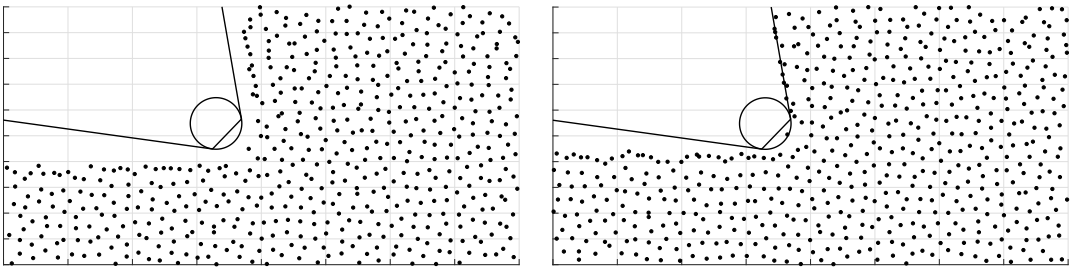

Figure 7.38: Close up of Monaghans algorithm in cutting, using smooth contact on the left and penalty contact on the right. The gap between the workpiece and the tool caused by the smooth contact is clearly visible on the left.

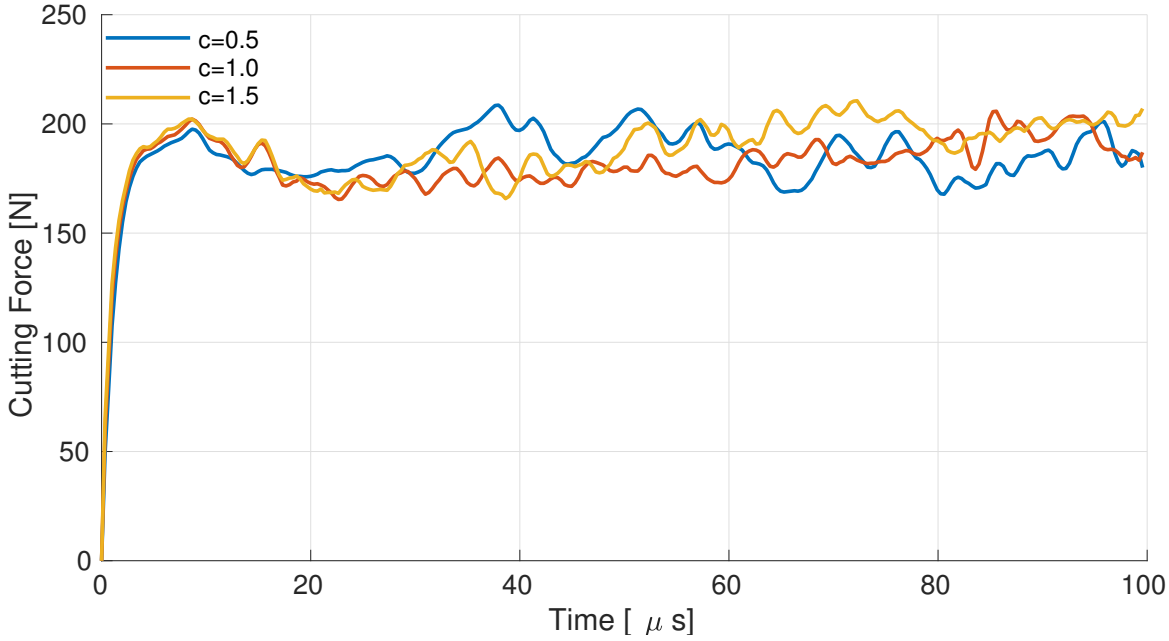

Figure 7.39: Force history of the cutting forces reported by the smooth contact algorithm for three values of $\kappa = c \cdot 1 \cdot 10^{-3}$.

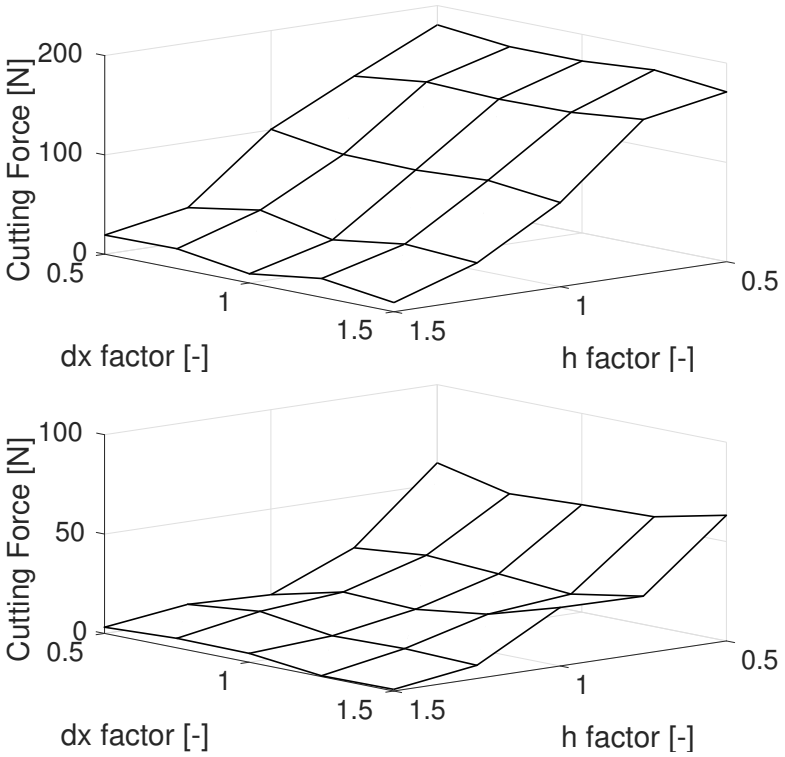

Figure 7.40: Plot of cutting (top) and normal force (bottom) against varying sampling density and smoothing length of the repulsive particles on the tool.

this setting the following recommendation to configure the smooth contact algorithm can be made: Super sample the tool with regard to the workpiece and set $\bar{h}$ to a small value. Values lower than, $\bar{h} = 0.5 \cdot h$ say, are not recommended in order to not invalidate the smooth nature of the contact algorithm.

It remains to determine if the contact algorithm with recommended configuration can reproduce the experimental study over the chip shape as presented in the previous sections. The results to this end are presented in figure 7.43. As can be seen, the results are very comparable to the Monaghans algorithm with refit friction coefficient 7.31 or the TVF with adapting smoothing length 7.35. This renders the smooth contact algorithm in question a viable alternative to the more commonly used penalty contact algorithm.

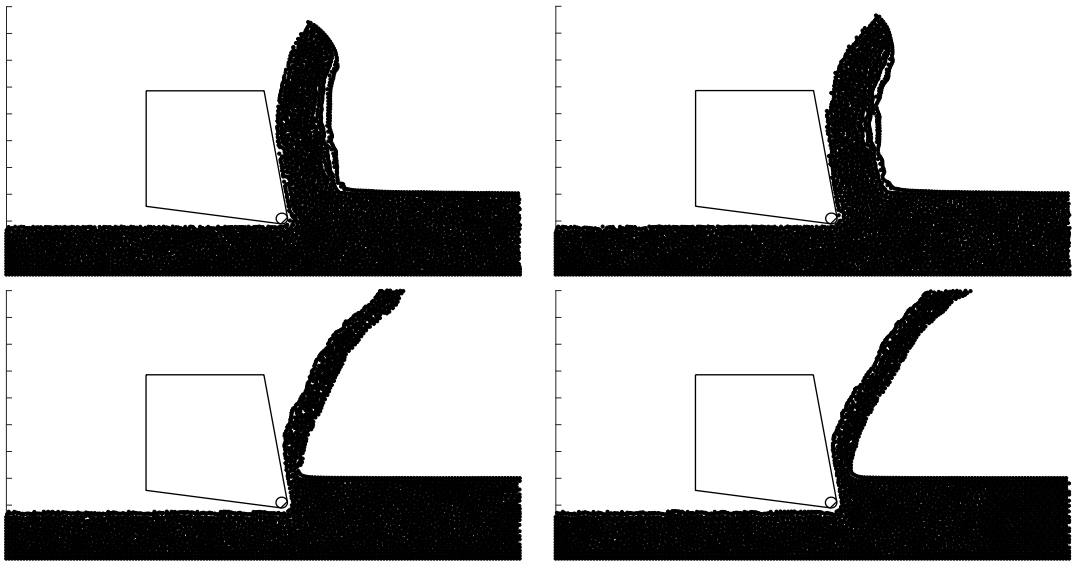

Figure 7.41: Chip shapes for different configurations of the smooth contact algorithm. Top left: $\bar{h} = 0.5 \cdot h$, $\Delta\bar{x} = 0.5 \cdot \Delta x$, top right: $\bar{h} = 0.5 \cdot h$, $\Delta\bar{x} = 1.5 \cdot \Delta x$, bottom left $\bar{h} = 1.5 \cdot h$, $\Delta\bar{x} = 0.5 \cdot \Delta x$, bottom right $\bar{h} = 1.5 \cdot h$, $\Delta\bar{x} = 1.5 \cdot \Delta x$.

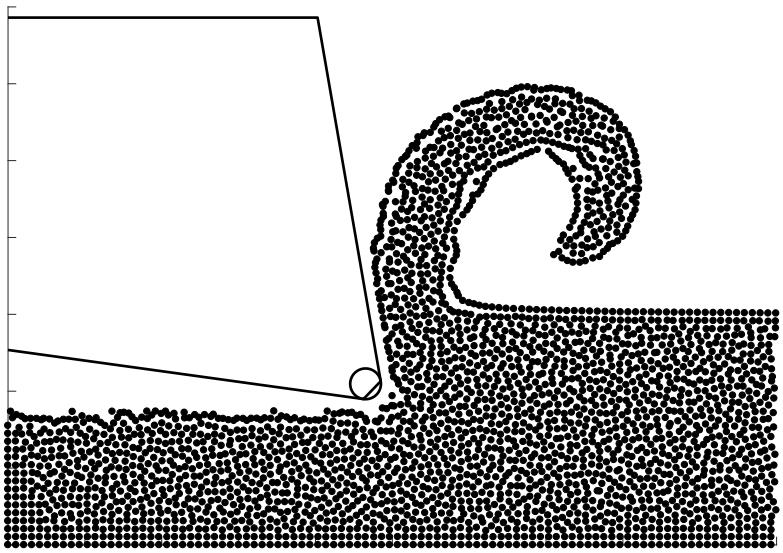

Figure 7.42: Choosing smoothing length and sampling density equal to the configuration of the work piece causes heavy chip curling. Notice the gap between tool and workpiece again to appreciate the repulsive nature of the smooth contact algorithm at hand.

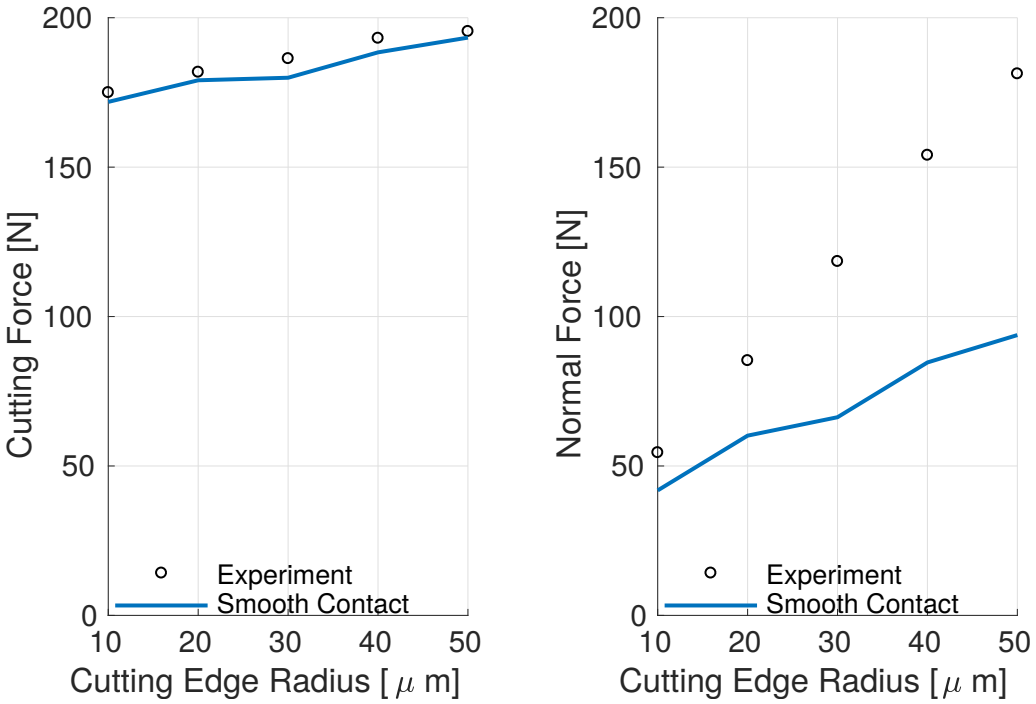

Figure 7.43: Cutting and normal force under variation of the cutting edge radius for Monaghans algorithm with smooth contact. The normal force on the tool points upwards in figure 7.23.

7.5.2 *Total Lagrangian Methods for Metal Cutting Simulations - Proof of Concept*

While some of the previous examples featured quite large deformations, none of the required the generation of new surfaces. This is a problem both for updated Lagrangian and total Lagrangian methods, since generation of surfaces i.e. topological changes, violates one of the basic assumptions of continuum mechanics. However, total Lagrangian methods are affected disparately more than updated Lagrangian ones. This was already illustrated by the simulation described in 5.20 and is illustrated again conceptually in figure 7.44. Hence, an additional benchmark is proposed to ensure the correct working of the visibility test, see 5.4.6. Thereafter, some proof of concept metal cutting simulations are exhibited.

7.5.2.1 *Preliminary Tests - Surgical Cutting Model*

In this section, the tensile test from section 7.2.1 is repeated, but this time a a perfectly sharp blade cuts the specimen while it is stretched. By "perfectly sharp" the following conditions are implied:

- The blade does not impose contact forces on the specimen, nor vice versa
- The blade is infinitely thin

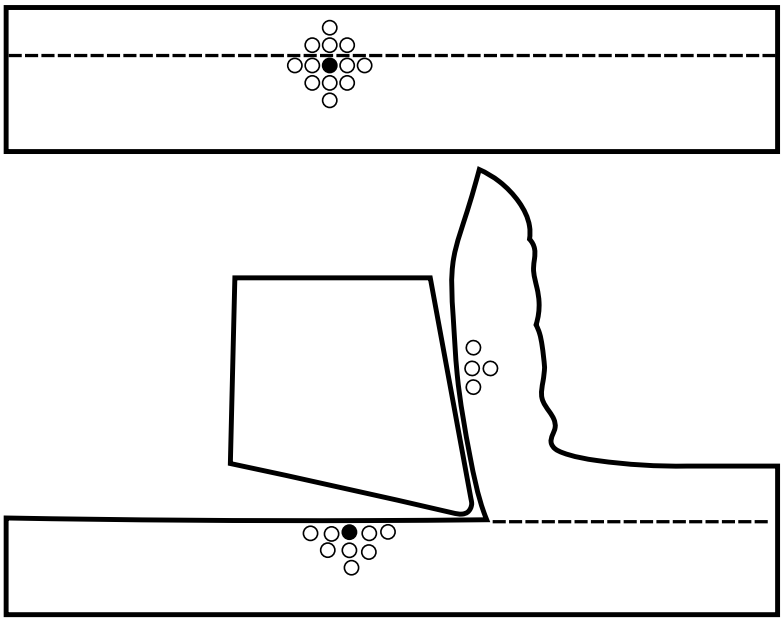

Figure 7.44: The particle indicated in black has neighbors below the material separation point as well as on top of it. Some of the neighbors end up in the chip, some and the particle itself remain in the workpiece. This is especially problematic in total Lagrangian situations, since the particles would keep interacting if no visibility criterion is used.

These conditions are coined *surgical* in this thesis since these conditions are a suitable model for a surgical tool like a scalpel in some situations, e.g. in some virtual surgery applications [121]. Also, the material is assumed to be plastic, using material parameters of steel, see table 7.3. A schematic view of the simulation setup is given in figure 7.45. The cut is performed by simply dissolving visibility across the cut.

While a situation like 5.20, where a chisel is driven through a slab of material, could have been tested, this time accounting for visibility, the situation as described here was preferred since it isolates the issue of topological changes / generation of new surfaces more clearly than the situation in 5.20. It is stressed that this benchmark, due to the cutting model, does not represent a physical situation and is merely intended as a stability challenge. That is, it is impossible to tell which algorithm resolves the more favorable plastic strain field, say, but conclusions with regard to stability can still be made.

The results for all total Lagrangian algorithms still considered are given in figures 7.46 and 7.47 for two instants in time. As can be seen, the co-rotated SPH algorithm fails at quite an early stage of the simulation. It is speculated that this is because the rotation matrix is approximated by means shape matching procedure (5.36), is heavily deficient at the boundaries. This might be fixed by using first order complete kernels when estimating the local transformation matrix. This however, would defeat the purpose of the algorithms, which aims to avoid requiring first order complete kernels to establish correct resolution of rigid body modes. The co-rotated SPH was thus excluded from further discussion.

This preliminary simulation also revealed an important implementation detail: If corrected kernels, in this case the Randles-Libersky correction and the RKPM are used, care has to be

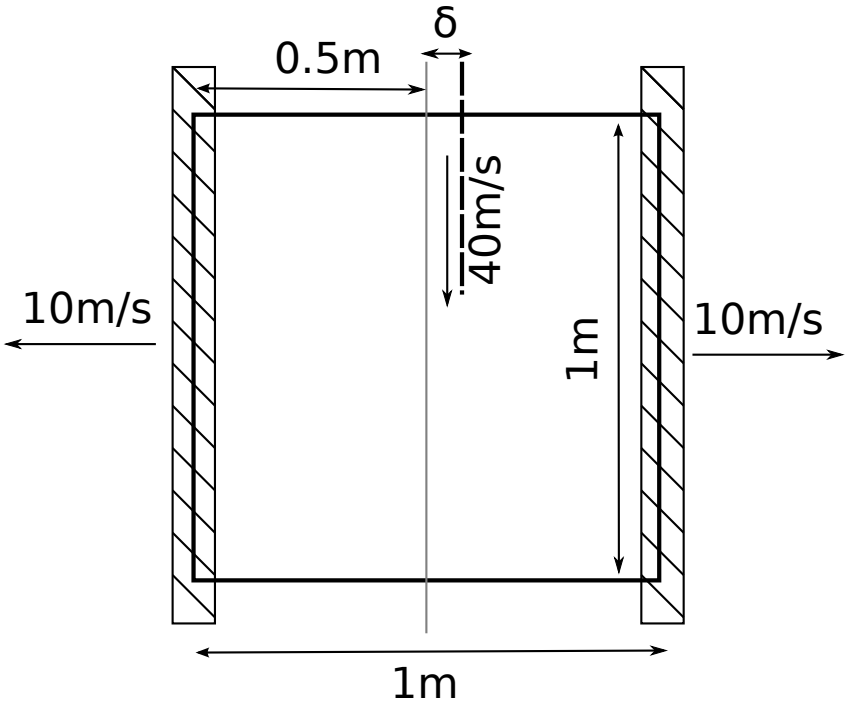

Figure 7.45: Sketch of the simulation setup for the tensile test including a surgical cut. The specimen is cut slightly off center by a distance of $\delta = 0.75\Delta x$ where Δx is the particle spacing. This is in order to avoid cutting through a particle or gauss point center. δ is not to scale. The cut is performed with $40 m/s$.

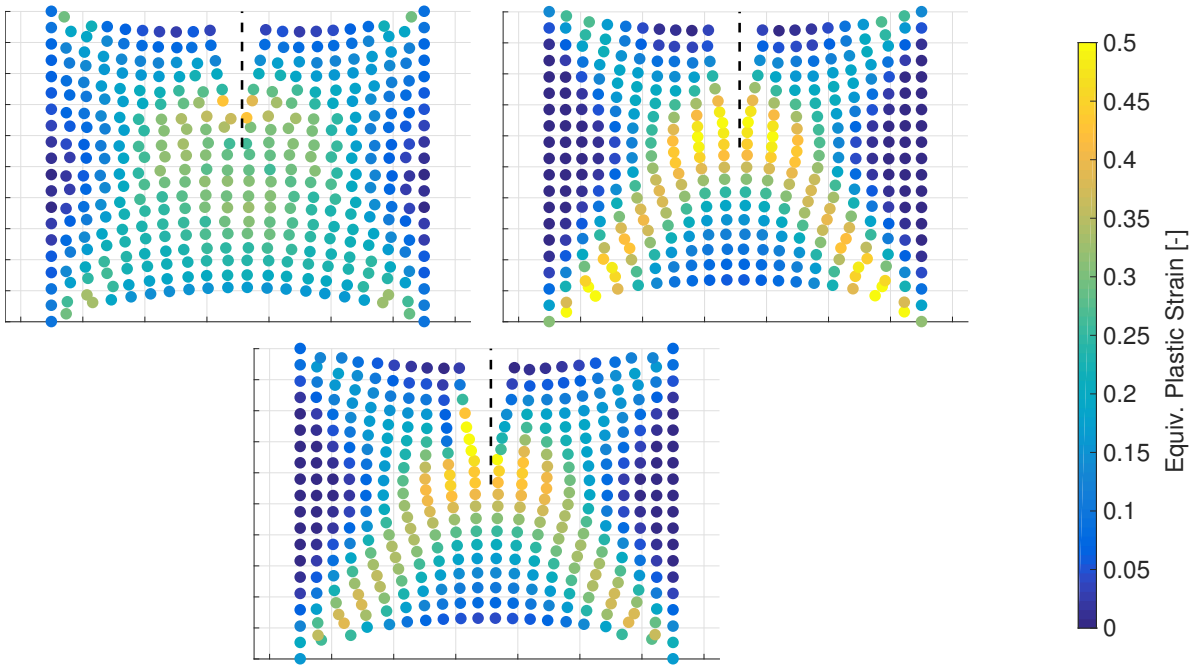

Figure 7.46: Frame of the tensile test simulation after about 0.010 seconds. The cut is indicated as black, dashed line. The co-rotated SPH shows some oscillation around the tip of the cut and fails right after.

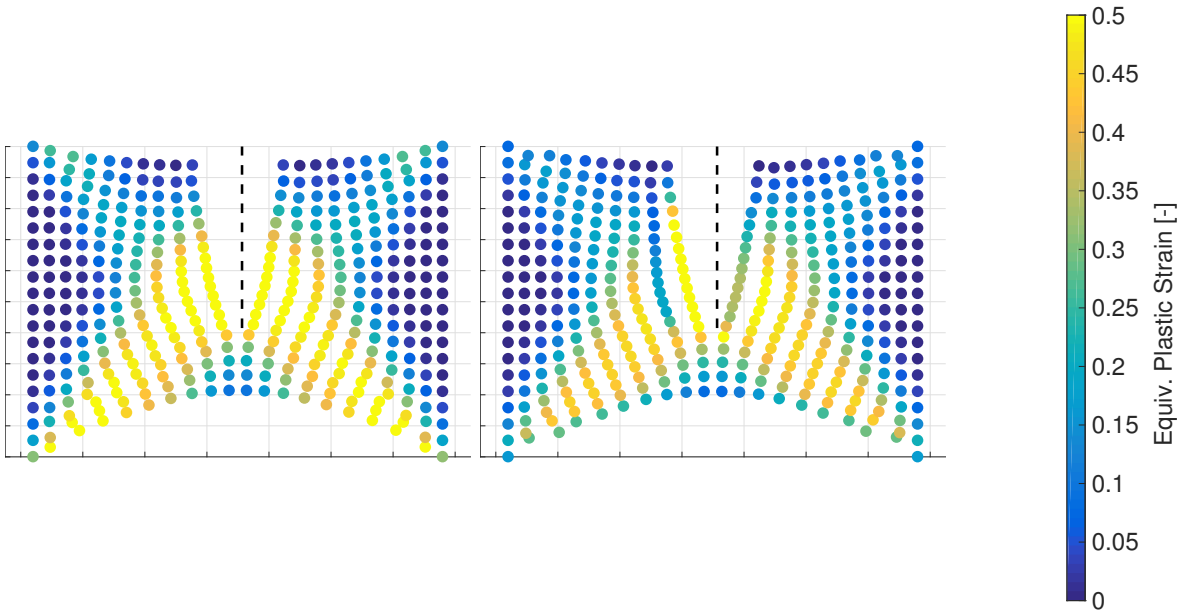

Figure 7.47: Frame of the tensile test simulation after about 0.016 seconds. The cut is indicated as black, dashed line. Reveles algorithm and the total Lagrangian weak form algorithm remain stable.

taken if the neighborhood is adapted by the visibility criterion. The correction matrices (4.48) or (4.43), respectively, which are normally computed only once in the beginning in a total Lagrangian algorithm have to be recomputed every time this happens. Failure to do so lead to immediate divergence of the simulation, both in Reveles algorithm as well as in the total Lagrangian weak form.

7.5.2.2 *Results*

After the co-rotational SPH proved unsuitable in presence of surface generation two total Lagrangian algorithms are left: Reveles algorithm and the total Lagrangian weak form. To assess their suitability in metal cutting, the algorithms were subjected to a number of situations of rising complexity / closeness to the experimental situation. Trial simulations are usually conducted at, at least in the case of titanium cutting, unrealistic cutting speeds of 500 m/min, and no thermal conduction. This is because the CFL condition (7.1) is dominated by the wave speed, and not the maximum speed in the domain, allowing for simulation times roughly inversely proportional to the cutting speed.

Results of these trial runs are given in figures 7.48 and 7.49 according to the scheme given in table 7.6. Studying these figures reveals quite a concerning situation, while conditions “A” present a reasonable chip, more realistic situations “B” and “C” show quite some artifacts. Finally, situation “D”, which represents the experimental conditions, deteriorates completely: Either by exhibiting unacceptably high chip compression ratio in case of Reveles algorithm or heavy delamination at the tip of the chip in case of the total Lagrangian weak form.

It is speculated that this is due to yet another visibility effect not covered in the current implementation. Since a serrated chip is expected in the situation at hand, particles may keep

Table 7.6: Test scheme for total Lagrangian algorithms in orthogonal cutting. "adiabatic" still includes heat generation due to plastic work (3.122), but no thermal conduction, while "thermal" includes conduction as well (3.121).

-	500m/min	700m/min
adiabatic	A	B
thermal	C	D

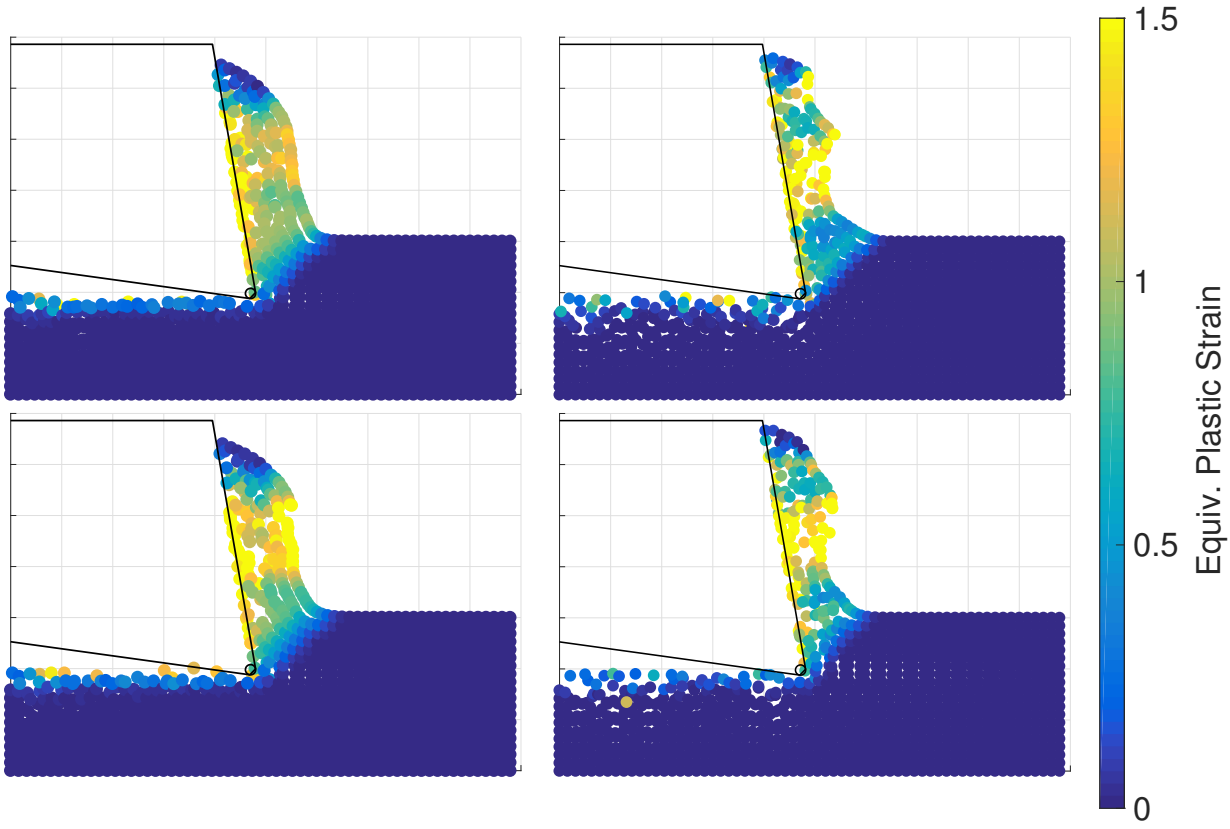

Figure 7.48: Trial simulations for Reveles algorithm according to scheme 7.6.

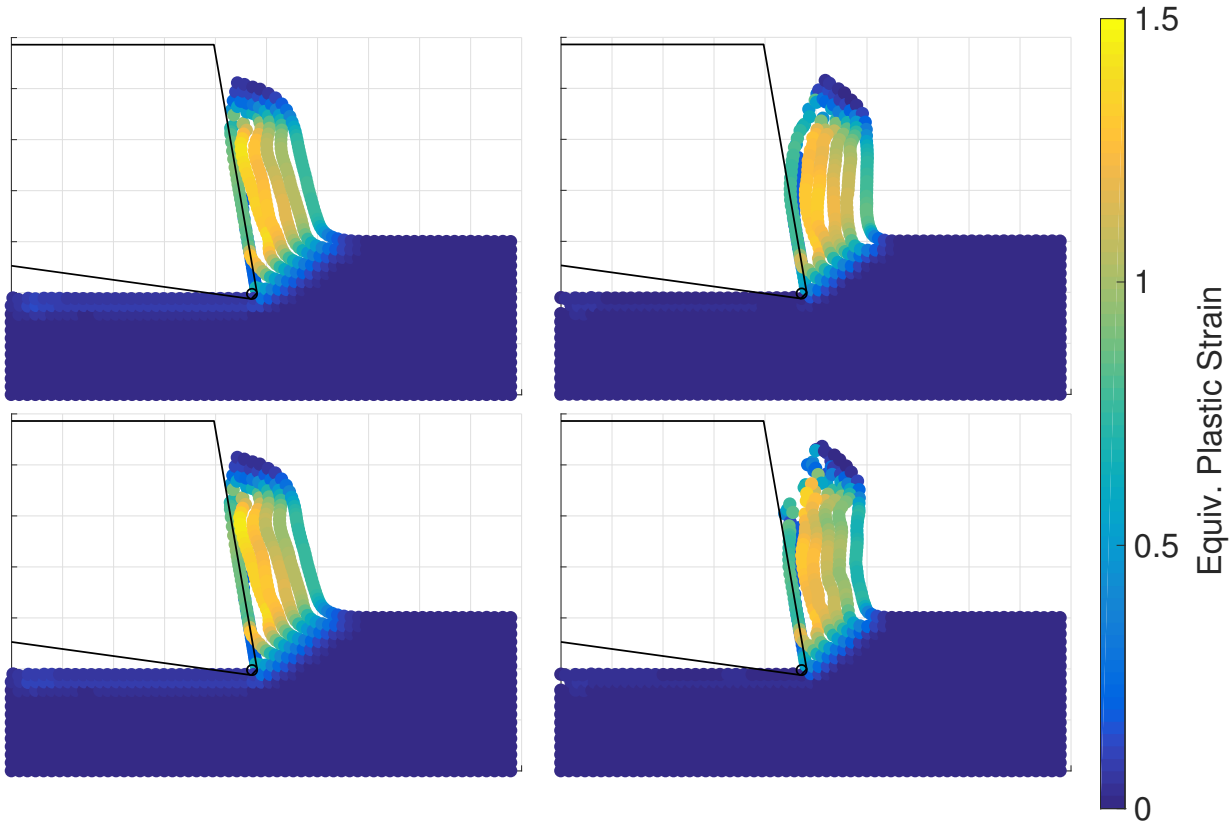

Figure 7.49: Trial simulations for the total Lagrangian weak form algorithm according to scheme 7.6.

interacting over the material boundary. This is illustrated in figure 7.50. It is concluded that the algorithm would resolve a segmented chip, but the lack of a proper visibility criterion introduced spurious forces, pulling the segments together again. This effect may cause the deteriorating chip shapes observed.

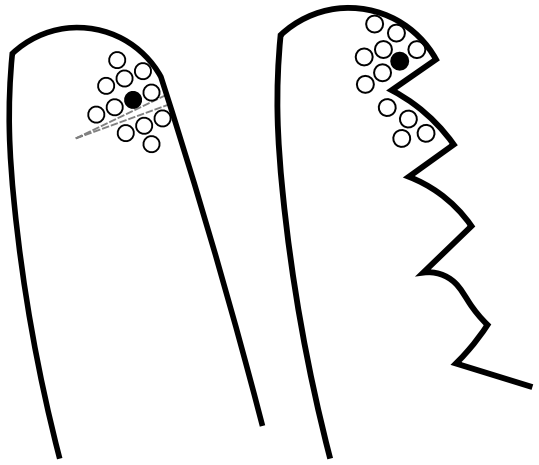

Figure 7.50: The mechanism that may be responsible for preventing serrated chips in total Lagrangian simulations illustrated. A gap would need to open do determine the segments. This is prevented by the situation on the right hand side; the particle in black keeps interacting with the particles one segment further down.

This effect can't be addressed with a simple scheme as described in 5.4.6, since an explicit, or in fact, any representation of the surface is available in a smooth method like the SPH. Instead, efficient surface tracking methods for particle methods by means of Particle Level Set (PLS) methods would need to be used, see for example [73], [72] or [106].

7.5.3 *GPU Specific Applications*

All applications so far could potentially be run on the GPU with the effect that the time to solution would have been one to two orders of magnitude faster. While convenient, there is no additional insight to be gained if the same simulation is run on the GPU instead of the CPU.

This section is concerned with the question of what additional utility there is to short production cycles beside convenience. Two applications are presented: First, rapid evaluation of cutting forces is exploited to automatically optimize a metal cutting process with regard to machining cost / tool wear. Then, the opposite route is taken, i.e. allowing simulations to run for hours again, but with very high resolution, allowing for a study with regard to the chip shape. To the best of the authors knowledge, these simulations constitute the most high resolution meshless orthogonal metal cutting simulations to date.

7.5.3.1 *Finding Profitable Cutting Conditions Automatically*

As stated, for example in the chapter 1, the interest in metal cutting, and a lot of other simulations, is two fold. On one hand, experiments may be explained and physical insight is to be gained. On the other hand, once a reasonable level of confidence is established in the simulation method at hand, it may replace experiments, allowing for quick variation of process parameters and the optimization of the underlying process.

A metal cutting process may be optimized in many ways, for example a process could be made as accurate as possible by searching for parameters causing the least amount of spring back. However, the most important optimization criterion for a company relying on metal cutting is probably the cost. In [137] it is suggested to model the cost as function:

$$\mathcal{G} = \frac{c_1}{\dot{V}_{MRR}} + c_2 \cdot \dot{\mathcal{W}}_{\text{Usui}} \tag{7.9}$$

where $\dot{V}_{MRR}$ is the Material Removal Rate, $\dot{\mathcal{W}}_{\text{Usui}}$ is the Usui wear rate [266] and c_1 and c_2 are constants. $\dot{V}$ is to be understood as the volumetric velocity, not as an acceleration as in $\underline{\dot{v}}$. The Usui wear rate was chosen in accordance with [137]. The same cost function could be employed with other, potentially more involved wear models/rates. The reasoning behind this approach is clear: A metal cutting process becomes more profitable the more material is removed per unit time, and more costly the higher the tool wear. The constants c_1 and c_2 can be used to tune the function to either more "aggressive" strategies favoring higher $\dot{V}_{MRR}$, or more conservative ones avoiding tool changes.

Table 7.7: Relevant thermal properties of Tungsten Carbide.

Parameter	Value	Unit
Density ϱ	14'700	kg/m^3
Specific heat capacity c_p	39.8	$J/(kgK)$
Thermal conductivity k	110	$W/(mK)$

The Usui wear rate is defined as:

$$\dot{\mathcal{W}}_{\text{Usui}} = \Psi e^{\xi/T} pv \tag{7.10}$$

where ξ and K are material parameters, or rather material paring parameters since they need to be determined for each workpiece and tool pair. Parameters are given in [197] as $\xi = 2500\text{K}$ and $\Psi = 7.8 \times 10^{-9}\text{Pa}^{-1}$ for a Tungsten Carbide tool. No further information on the tool this values are reportedly varied for is given in [197]. p is the pressure on the tool, T the temperature and v is the magnitude of sliding velocity. Since the temperature T is in the exponent of (7.10) it is especially vital that the thermal modelling is as accurate as possible. To this end, the full range of thermal effects presented in this thesis has been incorporated:

- Heat generation due to plastic work
- Heat generation due to frictional work
- Heat conduction through the workpiece
- Thermal contact between workpiece and tool

Additional thermal boundary conditions have been imposed, see figure 7.51, where the cutting parameters are defined as well. The thermal properties of the tool, i.e. the ones of carbide, are given in table 7.7.

The material removal rate can be estimated by $v_c \cdot d$, i.e. cutting speed times cutting depth. This is an estimate because the material separation point is generally higher than $D - d$, refer to figure 7.51 for explanation of these quantities. These are probably the most important variables with regard to $\dot{\mathcal{W}}_{\text{Usui}}$ as well since they dominate the process forces. Still, an argument can be made that the tool wear is also influenced by a wide array of other parameters, especially the tool geometry.

It is thus not clear which and how many parameters should be modified to optimize $\mathcal{G}$ (7.9). By limiting the optimization time to roughly one day, about 100 simulations can be run. Thus, one can either reasonably exhaust few parameters, or scratch the surface of the parameter space spanned by many different factors. While [137] chose the latter option by employing 12 parameters, in this work only d and v_c are modified. Next up lower and upper limits have to be identified for each parameter. This was done in accordance to the recommendations in [249] and with the experiments in [279]. Accordingly, cutting depth d was varied from 0.1mm to 0.3mm and the cutting speed was allowed to vary from 70m/min

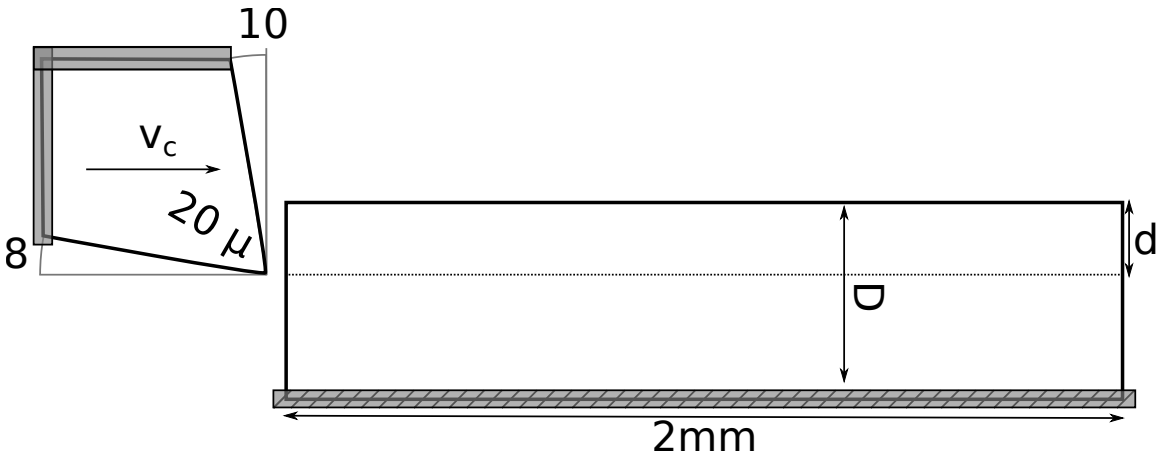

Figure 7.51: Sketch of the orthogonal cutting benchmark for the optimization procedure including cutting parameters. The cutting speed v_c and cutting depth d are varied. Additional boundary conditions where the temperature is fixed at room temperature have been indicated in gray.

to 200m/min. To avoid clustering in the sampling of the parameter space, lattice hypercube sampling was employed, see [201].

It is still not entirely clear how to even apply (7.9) since the Usui wear rate (7.10) is both local on the tool and instant in time. The average current wear was used for the optimization. That this is a valid choice is justified in figure 7.52, where it is shown that the average accumulated wear increases linearly after a very short simulation time.

Finally, 100 simulations with randomized d, v_c where run on both the Tesla P100 and the GTX660. Results can be seen in figure 7.53. As can be seen, the resulting cost function is quite noisy. This is because the simualtion resolution is quite low, and the cost function is dictated by the behaviour around the cutting edge where the wear is most severe.

In conclusion, a study analogous to the one presented in [137] has been conducted, but instead of on a cluster, on a single graphics card. Either a special purpose one in the case of the Tesla P100 or even on commodity hardware in the case of the GTX 660.

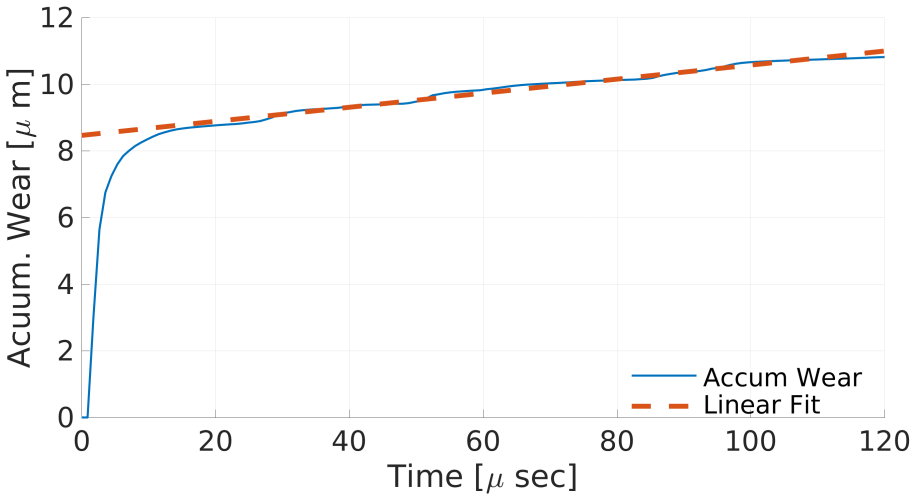

Figure 7.52: The average accumulated wear according to $\dot{\mathcal{W}}_{\text{Usui}}$ is recorded. As can be seen, the increase in wear becomes linear after a short amount of time, indicating that the wear rate is in steady state. This is further highlighted by the accuracy of the linear fit for the wear after 20 μs.

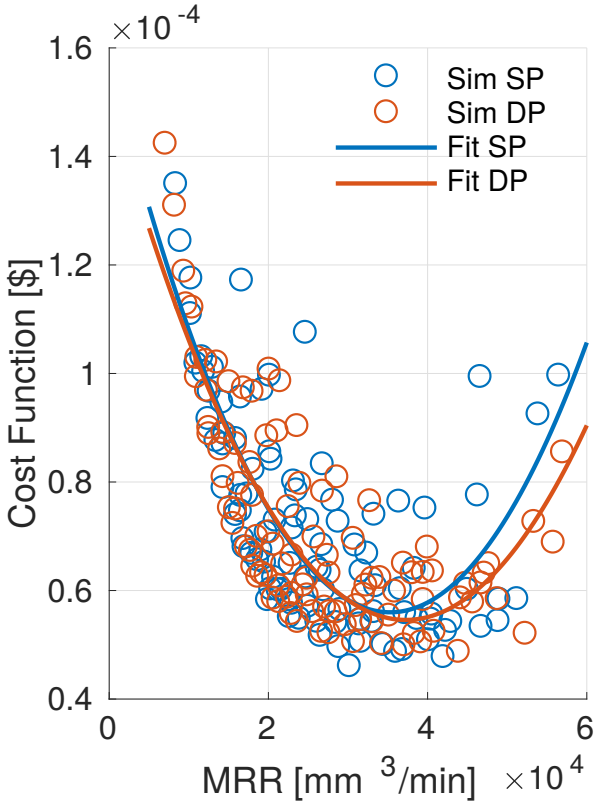

Figure 7.53: Optimization of a cutting process. MRR is plotted against cost function (7.9). The unit of the cost function is, in principle, currency. This has to be taken with a grain of salt, since parameters c_1, c_2 would have to be adapted to material and tool cost.

7.5.3.2 *High Resolution Orthogonal Metal Cutting - Studying the Chip Shape*

The previous section explored what can be achieved when exploiting the very short evaluation cycles offered by low resolution GPU accelerated simulations. This section explores the other alternative: instead of short simulation times, longer simulation times at very high resolution are explored. Timings in section 6.6.4 reveal that such simulations are hardly possible on the CPU. As a focus, the chip shape is studied again, see 7.5.1.2.

Studying the chip shape and chip forming mechanism serves both an academic purpose since there is quite a bit of controversy about the mechanism responsible actually forming the chip. In [44] for example it is claimed that plastic flow around the tool tip separates the material into chip and machined surface while [10] suggests that stable, ductile cracks build up in front of the cutting edge. On the other hand, the chip shape is quite important for metal cutting in practice; it is crucial with regard to tool design, design of the working area in the machine and uninterrupted execution of the machining operation at hand, see [59].

The focus of this chapter is the question whether increasing the particle number yields realistic results regarding the chip shape. To this end, the particle number in y direction is increased in steps of 30 particles, while keeping the sampling along the x axis proportional. This yields simulations with about 6′000, 24′000, 54′000, 95′000 and 150′000 particles, respectively. Other than that, the set-up is kept the same as presented in figure 7.23 with the same material data as in section 7.5. Using higher particle numbers simulations start to fail if the same contact stiffness, constant in the CFL condition (7.1) and correction constants are used as in the low resolution 6000 particle simulation. The simulations could be stabilized again by adapting these parameters. To keep all simulations comparable however, this was not done.

All visualizations in this chapter that show the chip shape were done using `SPLASH` [206]. `SPLASH` does not visualize the particles as point masses but actually shows the smoothed kernels added up on a pixel grid. Low resolution simulations thus appear with blurred boundary, which is not a visualization artifact but the reality of SPH. Also, since the 6′000 particle results are equal to results already presented in section 7.5.1, they are omitted from the results in this section, and only the four higher resolution simulations are shown.

In a first attempt, the simulations were run including thermal softening, i.e. generation of heat by plastic work, but without thermal conduction. Figure 7.54 shows Monaghans algorithm with artificial stresses disabled. Only very few segments are resolved that appear almost disconnected. Heavy oscillatory artifacts are observed at the boundary of the workpiece. This reveals the necessity of a corrected SPH scheme. The uncorrected version was not explored further in this section.

Chip shapes for both Monaghans algorithm, this time including artificial stress terms and the TVF are available in figures 7.55 and 7.56. As can be seen, only Monaghans algorithm converges to a stable, serrated chip. The segments of the TVF become smaller and smaller with increasing resolution, and become virtually non-existent at 150′000 particles, at least at the bottom of the chip. The chip curling increases with resolution in both methods but is more striking in the TVF. The clear serrated structure of the chip in Monaghans algorithm

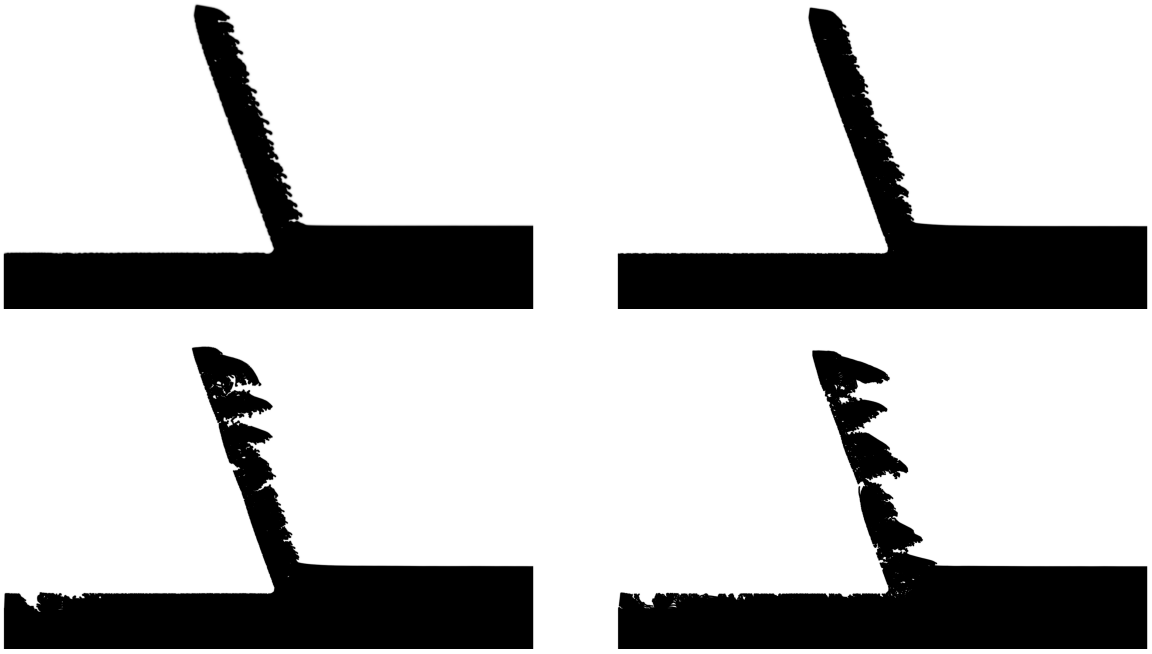

Figure 7.54: Chip shapes for 24'000 53'000, 95'000, 150'000 particles, top to bottom, left to right. No correction terms are used. Thermal conduction (3.121) is deactivated.

asks for a quantitative assessment. This was done for the highest resolution in figure 7.57. As can be seen, the errors are very reasonable.

In a next step, thermal conduction (3.121) was activated. It can be seen from figure 7.59 that Monaghans algorithm resolves a flow chip in this case. The TVF seems to resolve some serration, at least in the lower resolution simulations. Zooming in it is revealed that just like in the adiabatic case the size of the segments does not converge but gets smaller and smaller with increasing particle number, see figure 7.58. Again, the chip curling increases with particle resolution and is more pronounced in the TVF.

The reason for the lack of serration is evident from figure 7.61: In the case without thermal conduction the heat introduced stays in place, causing high temperature ridges where the material can flow, which in turn causes the serrated chip shape. In the case including thermal conduction, the heat diffuses throughout the material, preventing high temperature areas from building up. In figure 7.62 this is further substantiated. Here, the thermal conductivity k was doubled and halved, respectively. In the case with double thermal conduction the trend from the right hand side of figure 7.61 continues, now eliminating the area of elevated temperature at the tool tip completely. In the case where the thermal conductivity was reduced, faint shear bands are visibly, but blurred out, leading wider and less pronounced segments.

Since thermal softening is the only softening effect currently modeled, and thermal conduction prevents thermal softening from taking effect in a fashion that would cause a clearly serrated chip, the question remains what causes the chip to develop its saw tooth shape in reality? The answer may be found by studying the stress-strain curves of Ti6Al4V, see [62]. It is evident from these measurements that Ti6Al4V does not only exhibit a thermal softening,

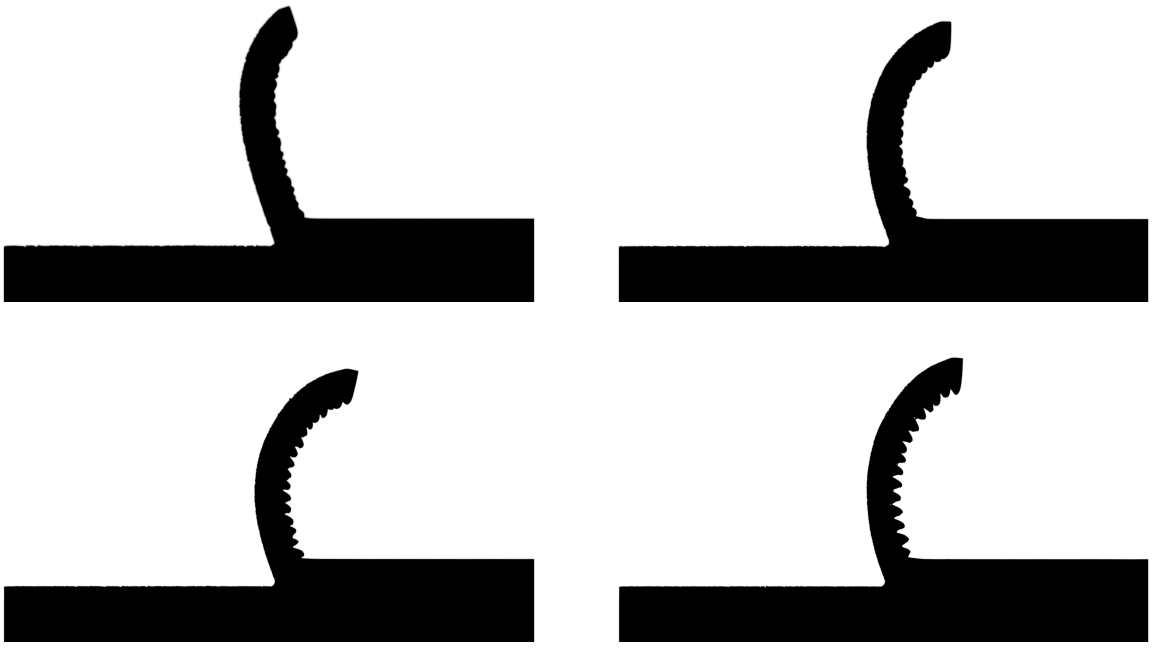

Figure 7.55: Chip shapes for 24'000 53'000, 95'000, 150'000 particles, top to bottom, left to right. Artificial stresses are used. Thermal conduction (3.121) is deactivated.

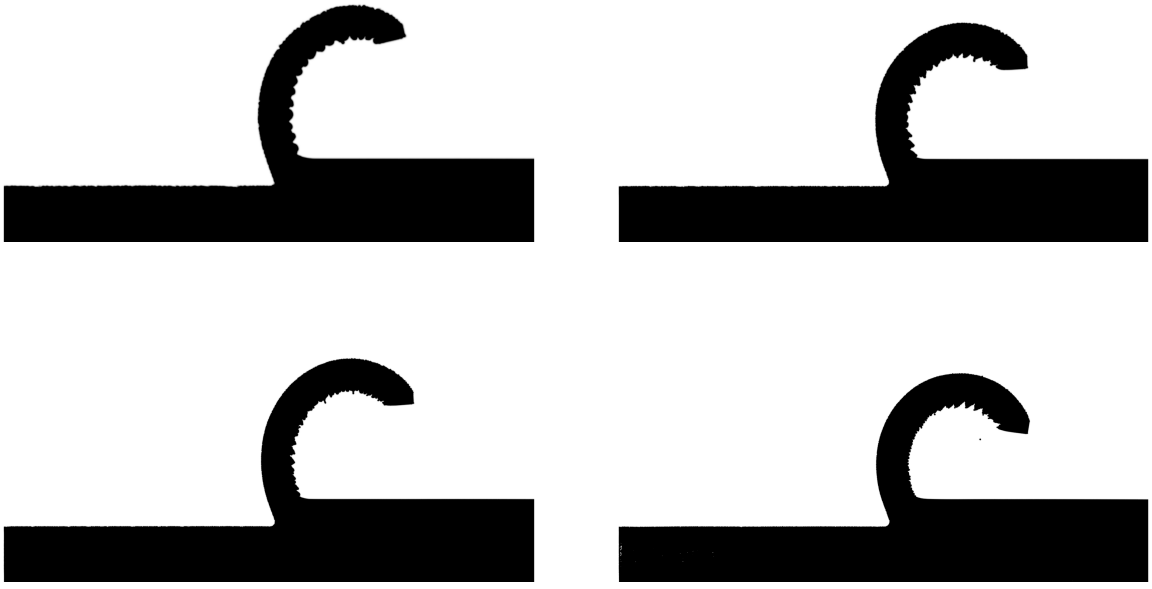

Figure 7.56: Chip shapes for 24'000 53'000, 95'000, 150'000 particles, top to bottom, left to right. The TVF is used. Thermal conduction (3.121) is deactivated.

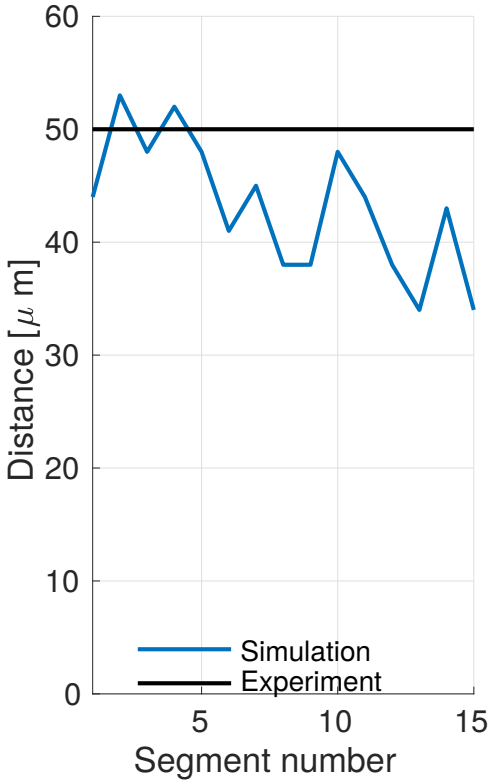

Figure 7.57: Measured distance between segments from bottom to top using Monaghans algorithm including artificial stresses. Thermal conduction is off. Experimental value indicated in black from [279].

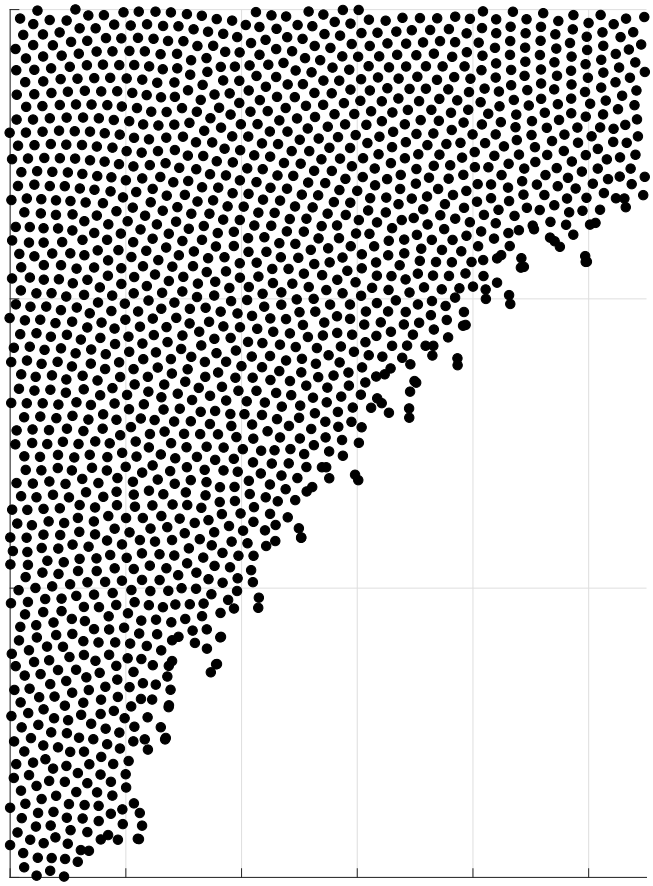

Figure 7.58: Close up of the chip edge in the 95'000 particles case. Many very small segments of only a few times the particle spacing are revealed.

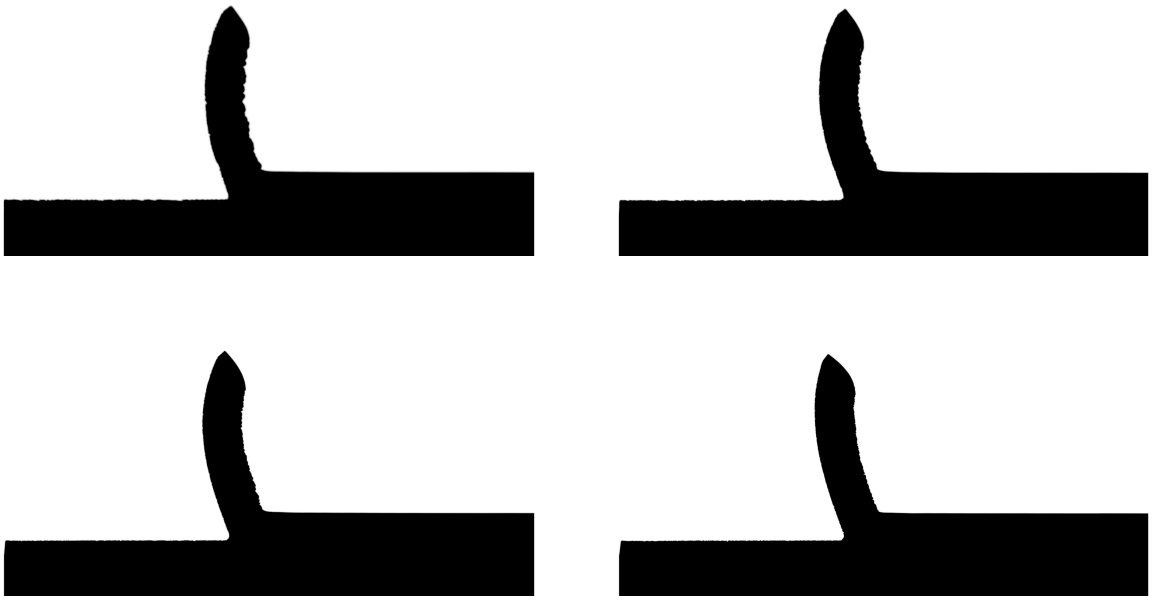

Figure 7.59: Chip shapes for 24'000 53'000, 95'000, 150'000 particles, top to bottom, left to right. Artificial stresses are used. Thermal conduction (3.121) is active.

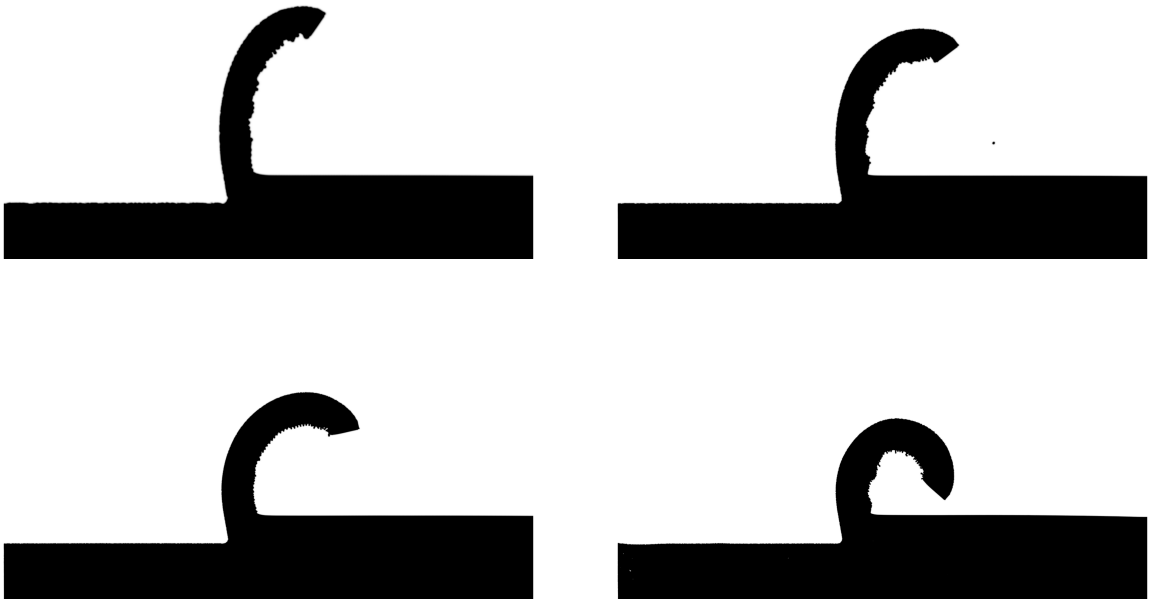

Figure 7.60: Chip shapes for 24'000 53'000, 95'000, 150'000 particles, top to bottom, left to right. TVF is used. Thermal conduction (3.121) is active.

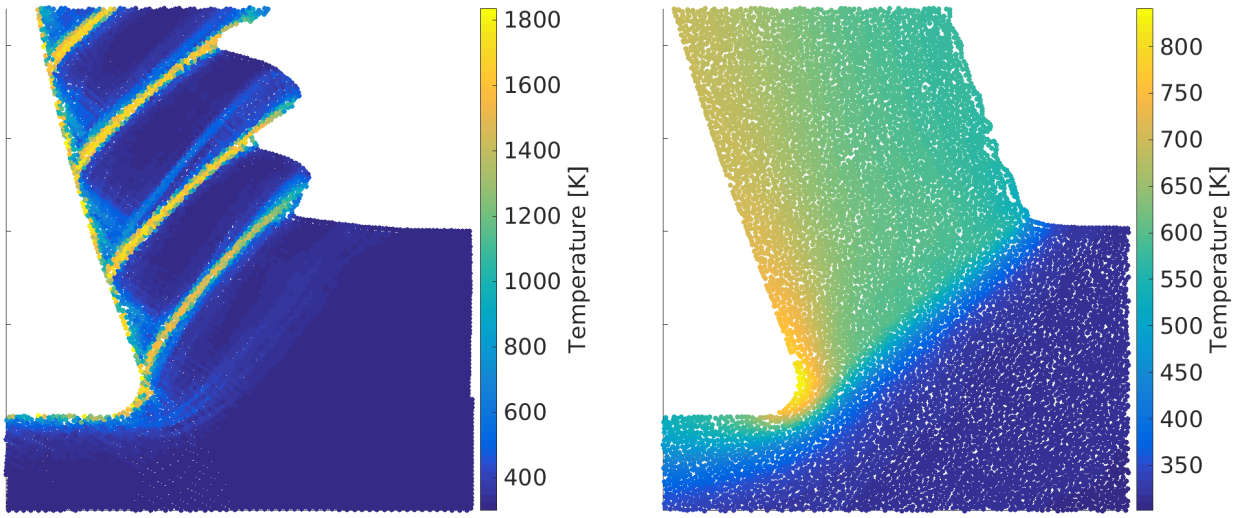

Figure 7.61: Close up of the chip root at highest simulated resolution of 150'000 particles. Left shows the simulation without thermal conduction (3.121), right including thermal conduction. Note the different color scale.

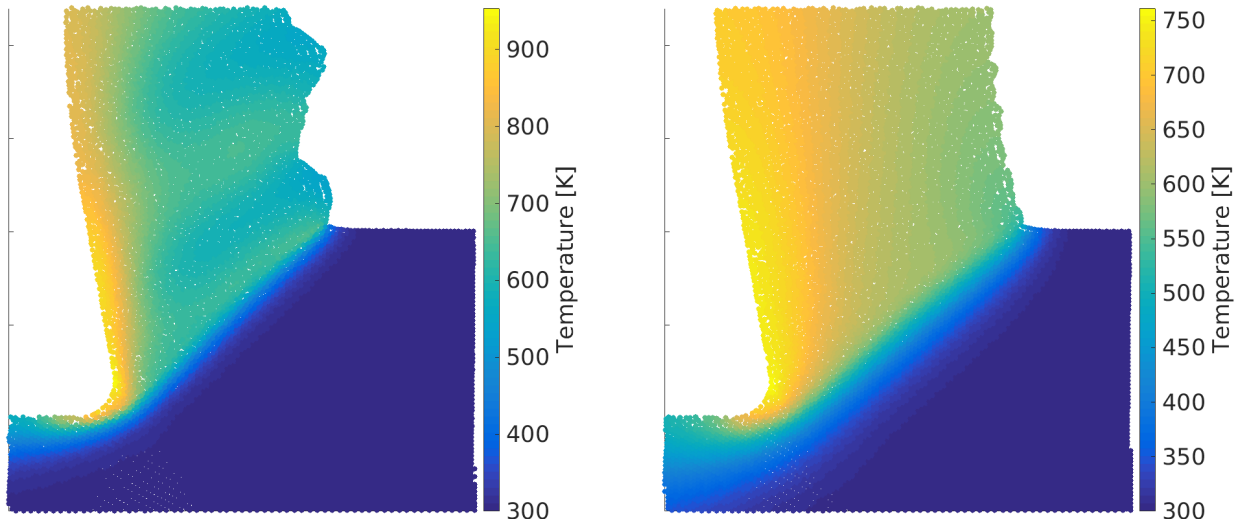

Figure 7.62: Close up of the chip root at highest simulated resolution of 150'000 particles. Both pictures show the simulation with thermal conduction (3.121), however, for the left picture k given in table 7.5 was halved and for the right one doubled.

Table 7.8: Parameters for the modified Johnson-Cook model (7.11)

a	b	c	d
1.6	0.4	6	1

but also a strain softening effect. This motivated the authors of [32] to modify the standard Johnson-Cook model (3.101) to include strain softening, or, alternativel speaking, they built in a damage model by decrease of the yield stress into the Johnson-Cook model. They came up with:

$$\begin{aligned}\sigma_Y(\bar{\varepsilon}_{pl}, \dot{\bar{\varepsilon}}_{pl}, T) = &\left(A + B(\bar{\varepsilon}_{pl})^n \frac{1}{\exp(\bar{\varepsilon}^a)}\right) \\ &\cdot \left(1 + C \ln\left(\frac{\dot{\bar{\varepsilon}}_{pl}}{\dot{\bar{\varepsilon}}^0_{pl}}\right)\right) \cdot \left(1 - \left(\frac{T - T_0}{T_m - T_0}\right)^m\right) \\ &\cdot \left(D + (1 - D) \tanh\left(\frac{1}{(\bar{\varepsilon}_{pl} + S)^c}\right)\right)\end{aligned} \tag{7.11}$$

with

$$D = 1 - \left(\frac{T}{T^m}\right)^d \qquad S = \left(\frac{T}{T^m}\right)^b \tag{7.12}$$

with additional material constants a, b, c, d. Note that these are different from the uppercase constants present in the original Johnson Cook model and they are all without unit. Values for these parameters for Ti6Al4V can be found in [64]. They are reproduced in table 7.8.

The analysis was re-run with this modified Johnson-Cook model and active thermal conduction. The results are exhibited in figure 7.63 and 7.64. Strain softening recovers the saw-tooth

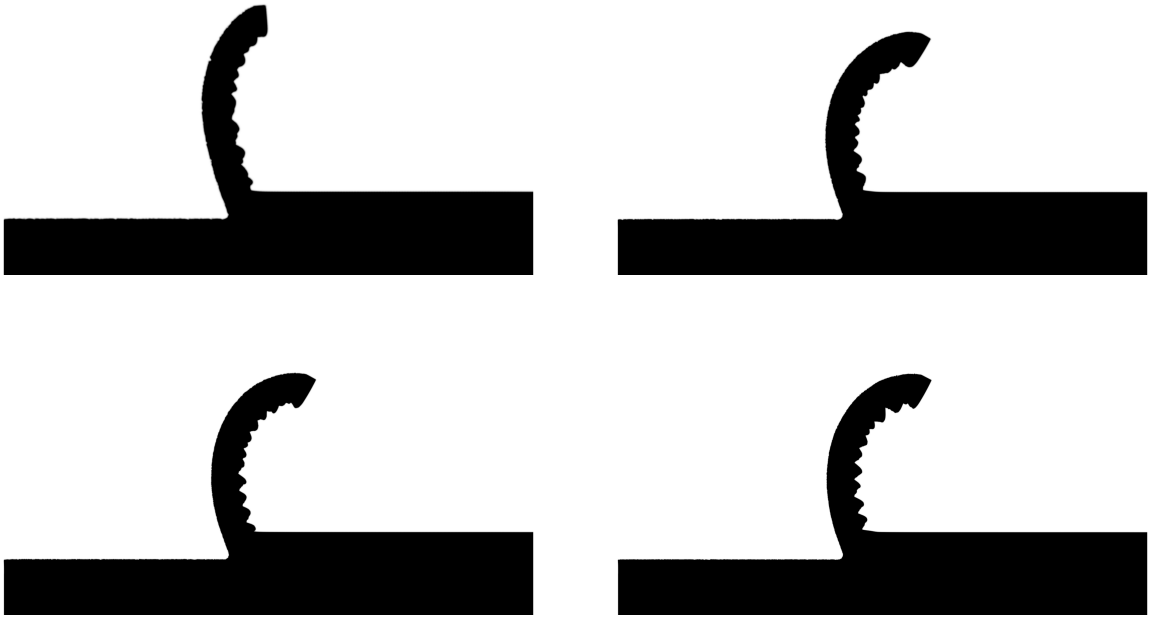

Figure 7.63: Chip shapes for 24'000 53'000, 95'000, 150'000 particles, top to bottom, left to right. Artificial stresses are used. Thermal conduction (3.121) is active. The modified Johnson-Cook model (7.11) is used.

shape of the chip, albeit the chip appears more compressed and curled, allowing for less segments to appear. Quantitative results for the highest resolution are available in figures 7.65 7.66 for Monaghans algorithm and the TVF, respectively. In the case of Monaghans algorithm, the values are close to the target of 50 μm, except for one outlier. In the case of the TVF the segments do not converge to a stable size but become more and more distant towards the bottom of the chip.

Even though the TVF fails to retrieve the chip shape quantitatively, both Monaghans algorithm and the TVF produce simulation results of extremely high quality at high resolutions in the sense that the plastic strain field is very smooth. This is unusual for meshless simulations which are easily afflicted by oscillatory modes. Compare figures 7.25 and 7.67, 7.68 to this end.

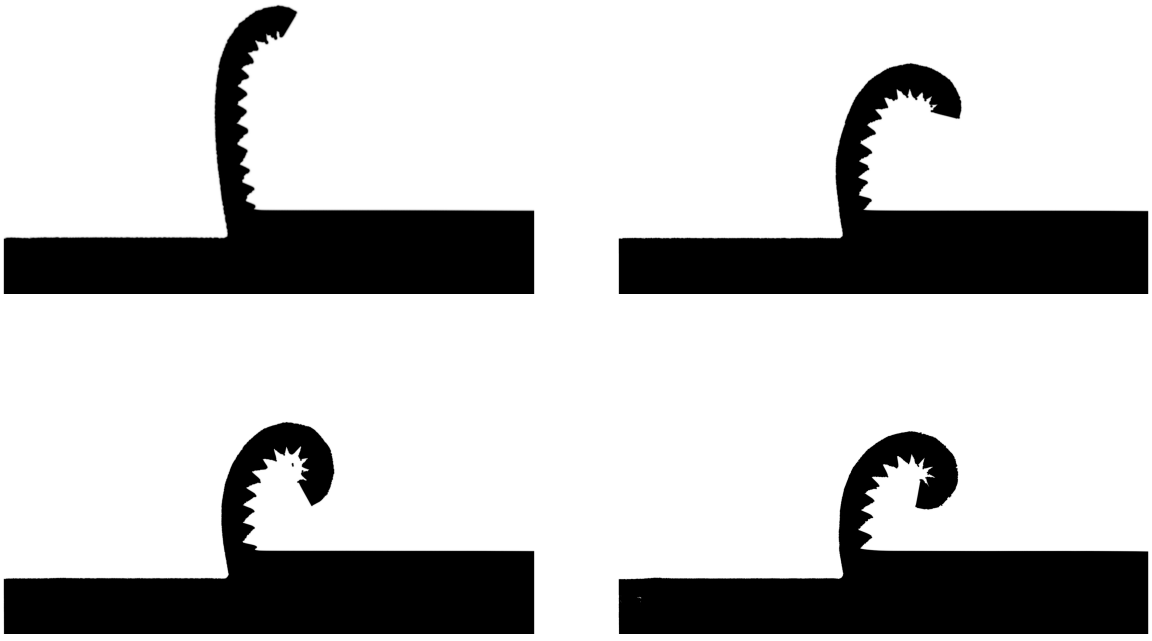

Figure 7.64: Chip shapes for 24'000 53'000, 95'000, 150'000 particles, top to bottom, left to right. Transport Velocity Formulation is used (TVF). Thermal conduction (3.121) is active. The modified Johnson-Cook model (7.11) is used.

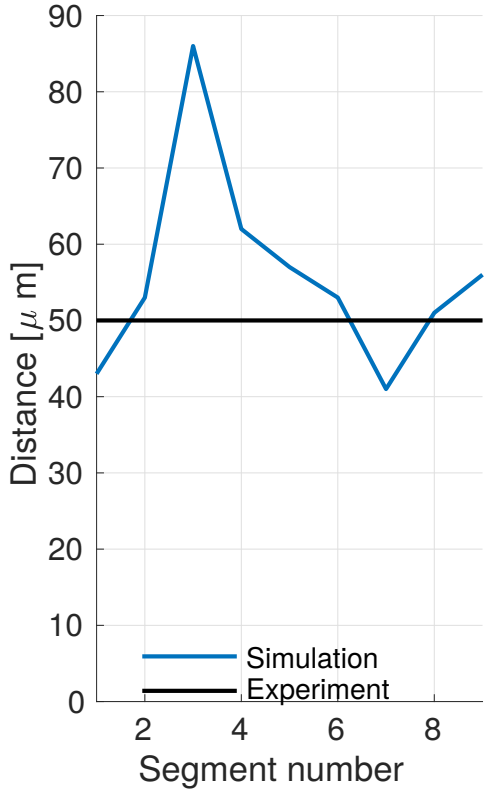

Figure 7.65: Measured distance between segments from bottom to top using Monaghans algorithm including artificial stresses. Thermal conduction (3.121) is on and the modified Johnson-Cook model (7.11) is used. Experimental value indicated in black from [279].

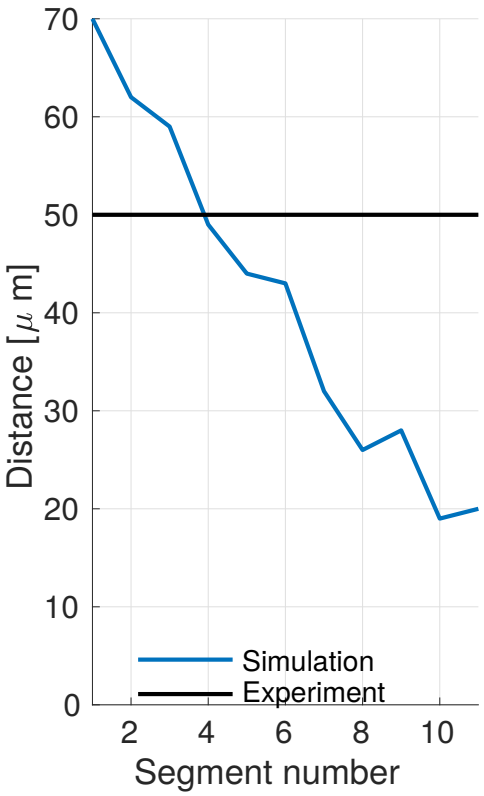

Figure 7.66: Measured distance between segments from bottom to top using the TVF algorithm. Thermal conduction (3.121) is on and the modified Johnson-Cook model (7.11) is used. Experimental value indicated in black from [279].

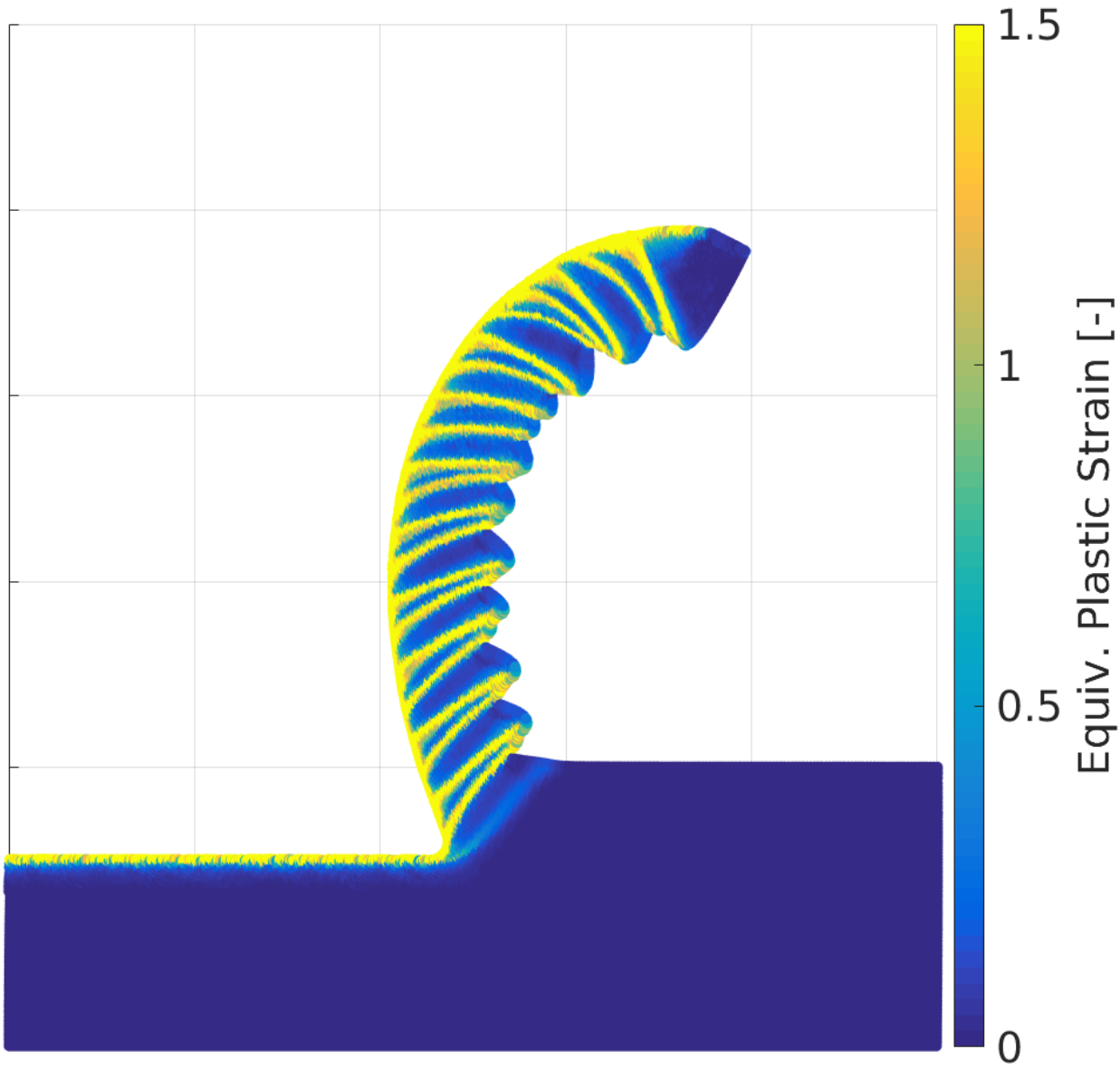

Figure 7.67: Final frame of 150'000 particle simulation for Monaghans algorithm.

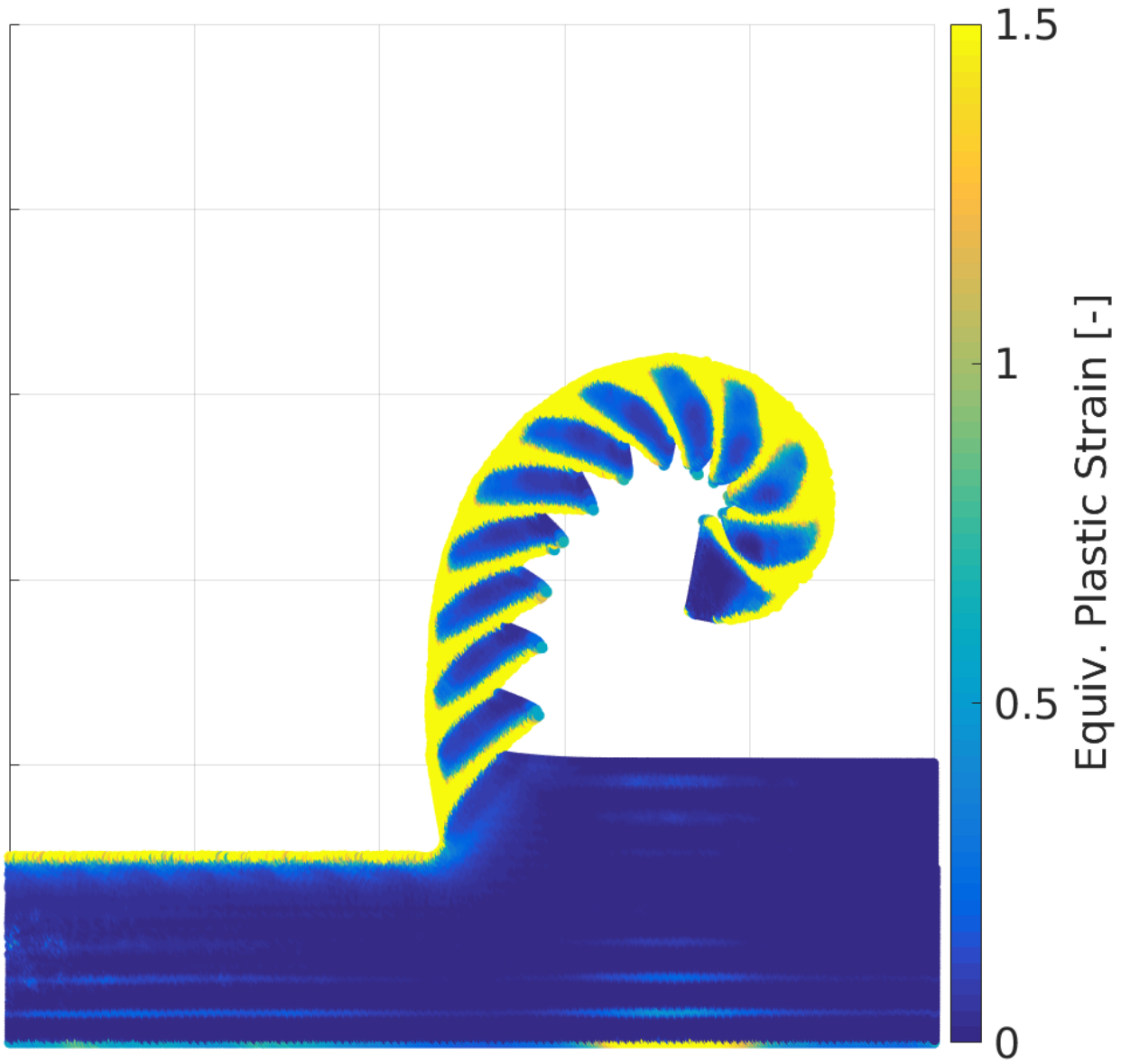

Figure 7.68: Final frame of 150'000 particle simulation for the TVF.

7.6 METAL CUTTING IN 3D ON THE GPU

In this section, all the knowledge generated from the previous results will be combined into a single simulation, which is to be run on the GPU exclusively. For this simulation, single grain cutting was chosen as application. Single grain cutting tests and simulations are important to understand grinding, since grinding can be seen as the action of superimposed single grains acting on the material to be processed. This is especially crucial in Engineered Grinding Tools (EGT)s, where the tool consists of a small number of diamond grains arranged in a defined pattern, see figure 7.69. This enables modeling these patterns towards a certain goal like chip shape [11], tool wear [136] or surface roughness [202].

While clearly interesting from a engineering standpoint, single grain cutting exhibits some unique challenges:

- The grains are not symmetric along any axis, necessitating the model to be in 3D.
- The grains feature a large negative rake angle. The already very severe deformations encountered during metal cutting are increased further
- The width of the cutting grove is one order of magnitude larger than the depth of cut. Hence, a very large number of particles is necessary to model the cutting geometry at scale, or a rescaling of the geometry is needed.

Analysis of single grain cutting using simulation is not new, however, due to the complications mentioned complete models are rare. Two dimensional models using FE are given in [196], which is technically not a single grain application, but cutting is done on the micro-scale with a negative rake angle. A three dimensional model restricted to the ploughing phase is available in [190].

There are two works which are very relevant with regard to this chapter: [3] simulates a single grain cutting experiment at full scale, using the updated Lagrangian FE solver in `Marc3D`. Simulations where run on a cluster using dozens of cores, running for about two days. [227] and precursory work [226], see also thesis [225], simulates a similar situation but at scaled geometry, using the SPH solver in `LSDYNA`.

This section uses the scaled geometry approach as in the publications just mentioned. Using this scaled geometry it is demonstrated how the GPU can be exploited to produce conceptual simulations using a few up to hundreds of grains.

Additionally, it is explored how the features new to three dimensional metal cutting, i.e. the linear complete kernel $\overset{\triangle}{W}$ from section 4.3.2, artificial stresses from section 4.7.2 and the TVF from section 4.7.5 influence the simulation results. Furthermore, thermal contact between the tool and the workpiece will be included, which was present in the FEM simulations in [3] but not the meshless simulations in [227], or any other meshless metal cutting simulation.

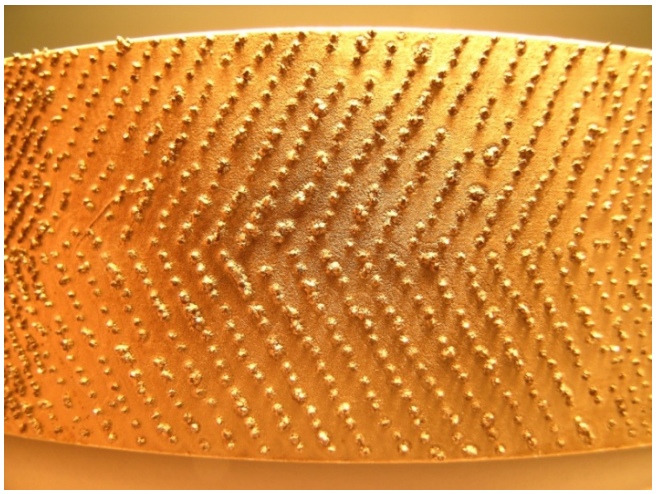

Figure 7.69: Photograph of an engineered grinding tool, taken from [273], also reproduced in [225]

7.6.1 *Simulation Model*

The basic geometry of the model is given in figure 7.70. The scaled and the unscaled geometry differ in the ratio of d and B. Typical values are 1 : 2 for the scaled geometry and 1 : 10 for the unscaled geometry. Experimental cutting speeds are usually very low for single grain cutting experiments, ranging from 0.18 to 0.8 m/s. This is because the feed motor is used instead of the spindle of the machine tool. However, since the simulations in this section are proof-of-concept and are not compared against experiments, a with regard to real grinding very modest speed of 30 m/s is chosen.

7.6.2 *Simulations using 1-5 Grains*

Currently, numerical cost does not allow for the simulation of multiple grains using the unscaled geometry, since a huge number of particles would need to be introduced to reasonably sample the depth of the cuts at the workpiece dimensions required. Hence, the diamond geometry was scaled down ten times, while depth of cuts between 15 μ and 30 μ typical for single grain cutting tests was kept.

Three kinds of simulations are performed in this section

1. Effects of different solver features, using one grain
2. Effects of the aspects of the thermal model, using one grain
3. Interactions of multiple grains, using three to five grains.

The workpiece was assumed to be Ti6Al4V with material parameters according to table 7.5. Friction was set to $\mu = 0.35$. All simulations were run at high resolutions, making them the most high resolution meshless 3D metal cutting simulations to date. The procedure

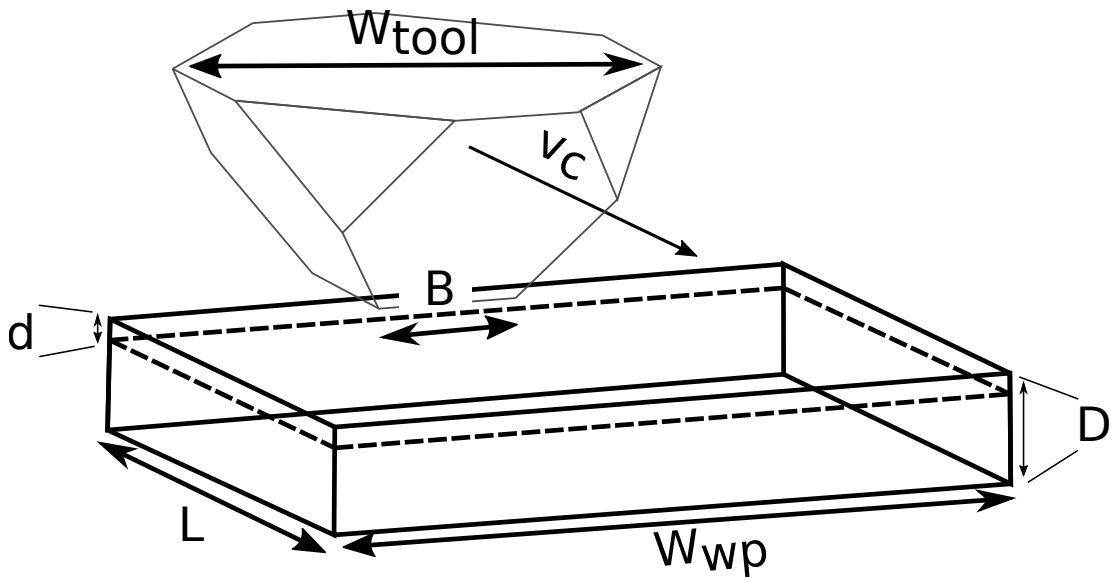

Figure 7.70: Sketch of single grain cutting geometry with diamond edge length B, depth of cut d and cutting speed v_c. D is chosen heuristically such that no plastic straining occurs at the bottom of the workpiece, unless stated otherwise $D \geq 3 \cdot d$.

for chosing the total particle number is completely determined by how many particles are seeded along the height n_{seed} of the depth of cut d. Sampling in the other directions is then kept consistent, i.e.

$$\begin{aligned} d_z &= d/n_{\text{seed}} \\ n_z &= \lfloor D/l \rfloor \\ n_x &= \lfloor L/d_z \rfloor \\ n_y &= \lfloor W_{wp}/dz \rfloor \\ n_{\text{total}} &= n_x \cdot n_y \cdot n_z \end{aligned} \tag{7.13}$$

The diamond grains used are not idealized grains in the sense that they are conforming to a specific geometric shape, but are obtained from optical measurements. Two diamonds are used. They are taken from [227], denoted "Diamond 1" and [3], denoted "Diamond 2". The diamonds are depicted in figure 7.71. The reason for using two different diamonds is because they are expected to produce different chip characteristics. Diamond 1 features a main and secondary cutting edge, while Diamond 2 meets the material with a single cutting edge and a large negative rake angle, favoring chip bending and even curling.

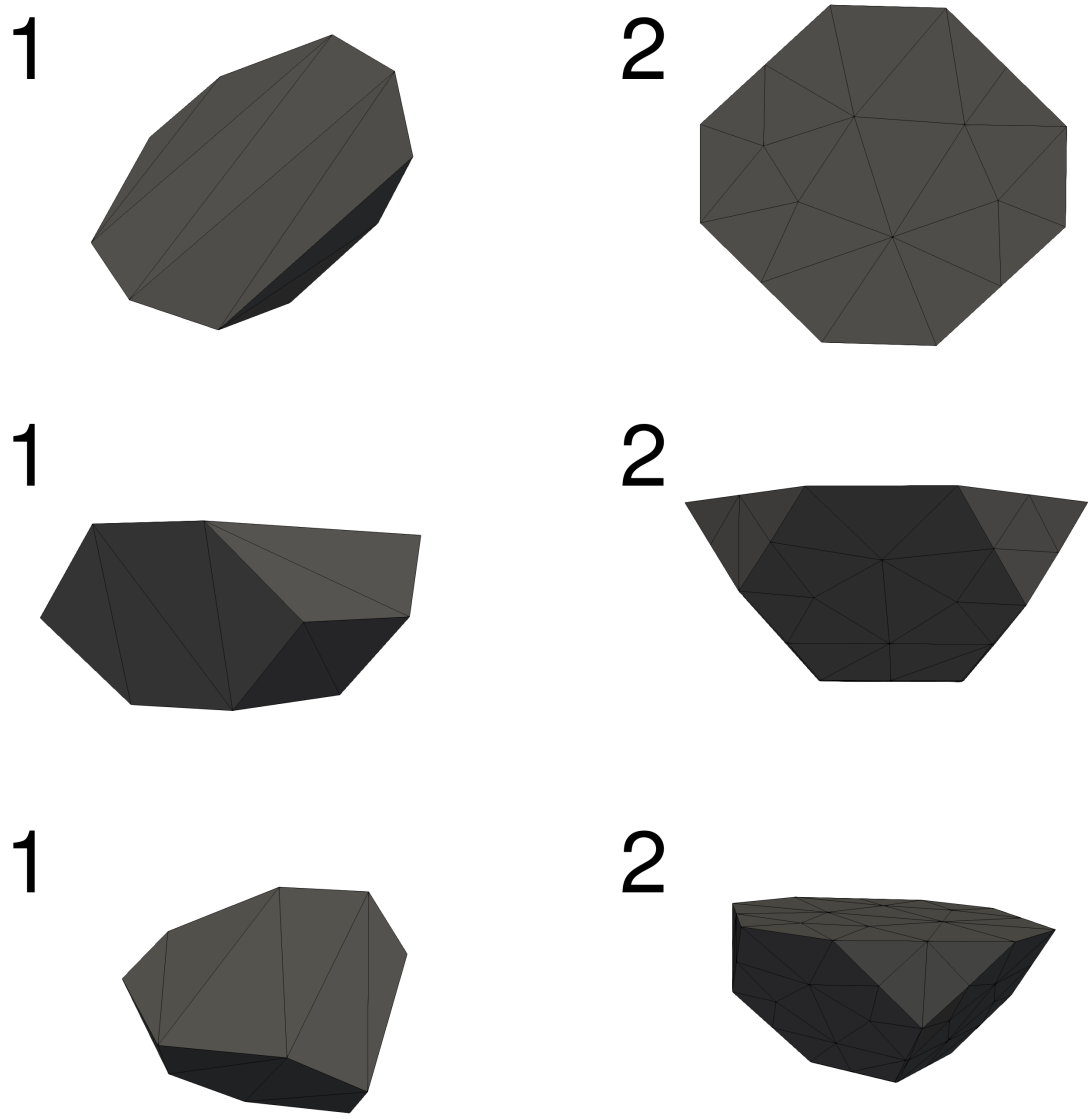

Figure 7.71: The two diamonds used, left and right, in top view, side view and isometric view, top to bottom. Cutting direction is to the right in the top and middle row.

7.6.2.1 *Testing Solver Features*

In this section, the following configurations are tested:

Label	CSPM	Art. Stress	TVF
A	✗	✗	✗
B	✗	✓	✗
C	✓	✗	✗
D	✓	✓	✗
E	✓	✗	✓

All configurations are run with XSPH from section 4.7.3 and artificial viscosity from section 4.7.1 turned on. In a first step, Diamond 1 was used. W_{wp} and L were set to $2 \cdot W_{\text{tool}}$, see figure 7.70, and the number of particles along the depth of cut n_{seed} of 15 μ was set to 18, resulting in about 2 million particles in total. This makes the simulations at hand the most high resolution meshless cutting simulations to date. All thermal features were turned off, i.e. the simulations are completely adiabatic. In a first step, result frames of simulations A and B are compared, see figure 7.72. As can be seen from the figures, the difference in the results obtained is negligible. While in a lot of two dimensional situations the presence of artificial stresses was necessary to obtain a meaningful simulation result, see sections 7.2.2 and 7.5.3.2, this was not the case in 3D, as evident from the three dimensional part of section 7.2.2, and does not seem to be the case here.

Simulation B is then compared to the other simulation in figure 7.73. The introduction of CSPM, i.e. simulations C, D, E, seems to have the largest impact on the result. The chip buldges away from the cutting edge and is bent around the diamond. The introduction of TVF does not seem to change much, except that the chip compression ratio is slightly decreased, resulting in a slightly enlarged chip.

Result frames in general do not reveal the whole story, however. Figure 7.74 explores the effect of the correctors on the cutting forces. The figure permits a few observations: Again, whether or not CSPM is used prompts the biggest difference in the response. However, the effect of CSPM on the forces is arguably even more pornounced than on the chip shape, increasing the forces with roughly a factor of two. Other interesting points include that the feed forces are virtually non-existant, even though the diamond in use is heavily tilted. The normal forces are even higher than the cutting forces, which is typical for grinding. The cutting force increase is in light of the fact that cutting and especially normal forces tend to be underestimated by particle and other methods, see [225], [9], a very intruiging point.

Having established that using CSPM has a large effect on chips and forces it was deemed interesting to try to maximize the impact with regard to the chip shape. In the publication [3], which uses Diamond 2, a picture of the experimental set up is shown, exhibiting a curled chip. This picture is reproduced in figure 7.76. Since in section 7.1 it was shown how employing CSPM can resolve rigid body modes, it is interesting to see if this could be recreated in the simulation at hand by replacing Diamond 1 with Diamond 2. At first, these attempts proofed futile at various cutting depths and speeds. Only after reverting

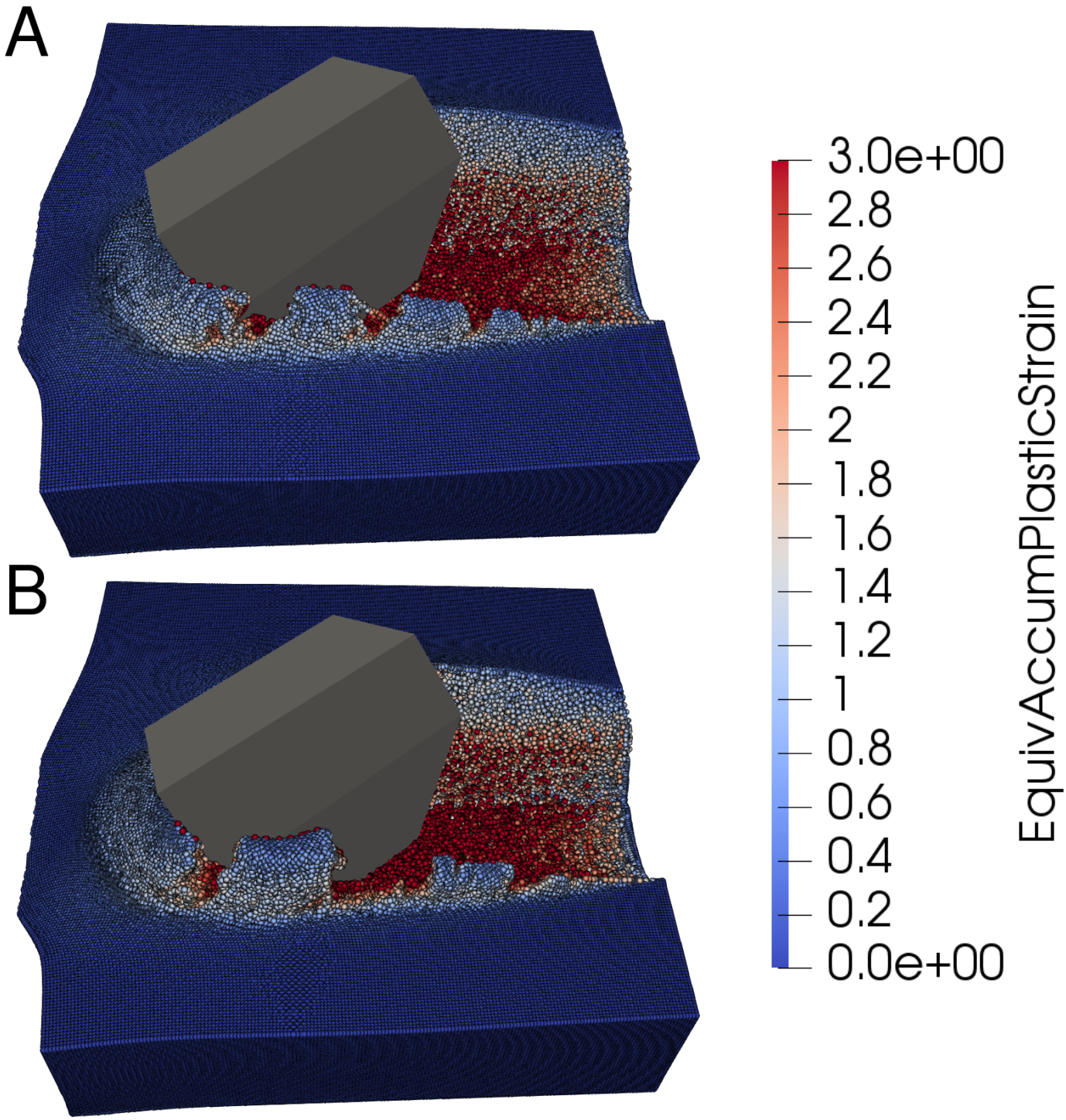

Figure 7.72: No corrective measures (top) vs. artificial stresses (bottom). The results can hardly be told apart, except for the slightly different break up of the chip in the side burr.

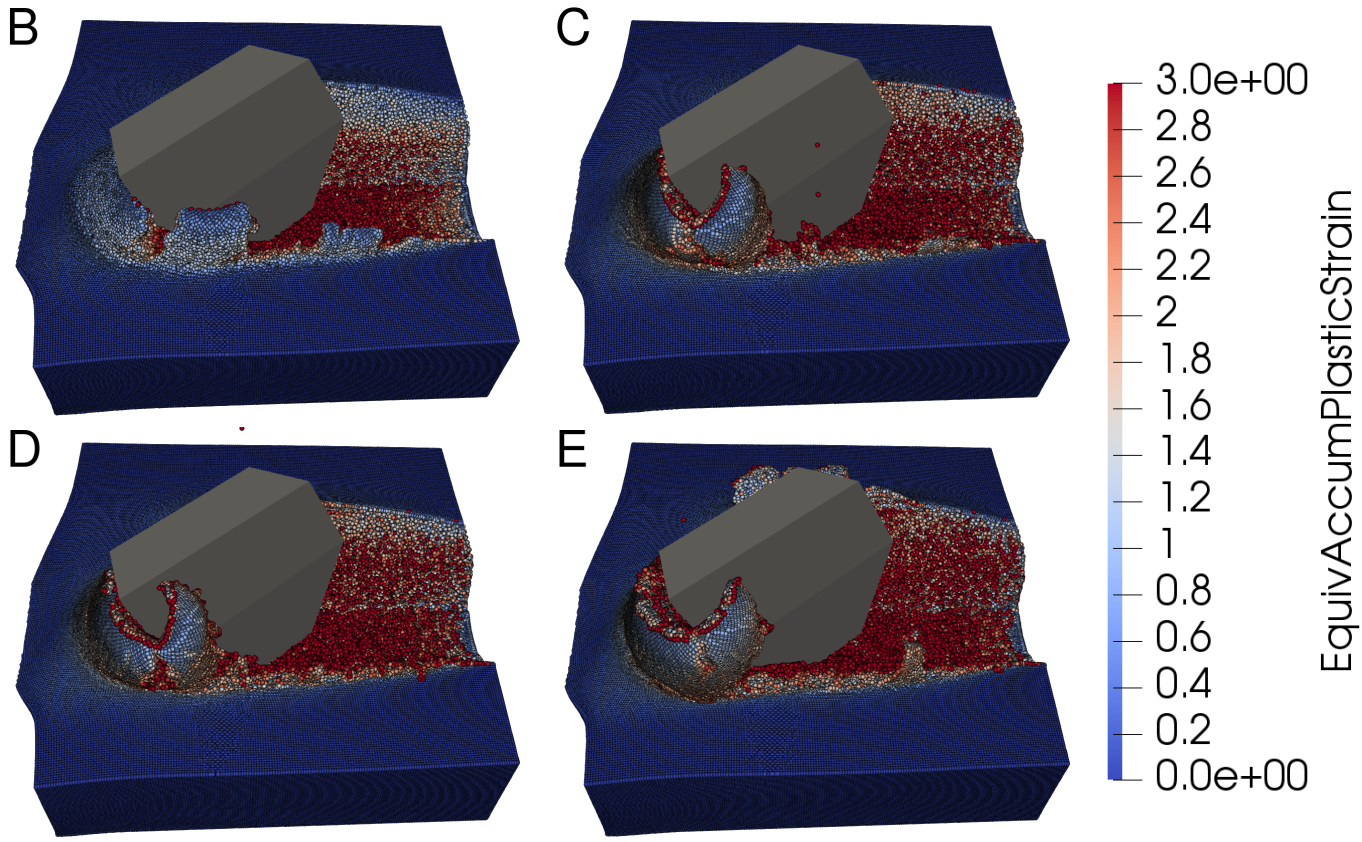

Figure 7.73: Simulations B, C, D, E. See beginning of section for specifications.

the diamond to the real scale and using the experimental cutting depth of 30 μm, a curled chip can be observed. To present this effect, simulation set ups A and D are re-run, using Diamond 2, at the real scale, i.e. not downscaled by factor of 10 as in the other solutions. Hence, n_{seed} had to be reduced to 12 particles to arrive at a comparable particle resolution of about 1.9 M particles. Some result frames are given in figure 7.75. Chip curling is now clearly visible in the CSPM case. In fact, the chip curling is so intense that the chip is forced against the workpiece. Particles in the chip and the uncut workpiece start to interact in sliding motion. Spurious strains and stresses develop, as shown in figure 7.78. This is the same effect as described in section 5.4 and figure 5.7. While the interactions between chip and workpiece are spurious, the chip curling is indeed very high in the experiment. Figure 7.77 shows a close up to better compare the results to the experimental situation 7.76, demonstrating remarkable similarity.

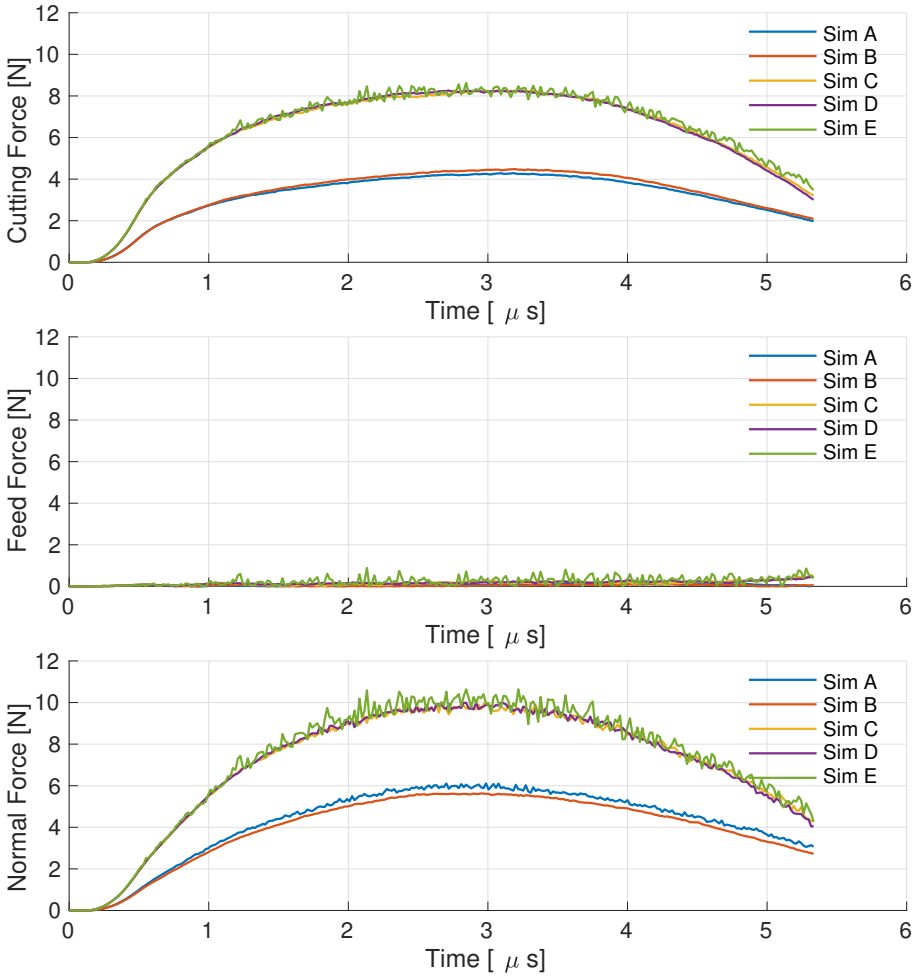

Figure 7.74: Cutting, feed and normal forces (top to bottom) of simulations A-E. Absolute values are plotted, from the tool perspective the normal forces point upward in figure 7.70.

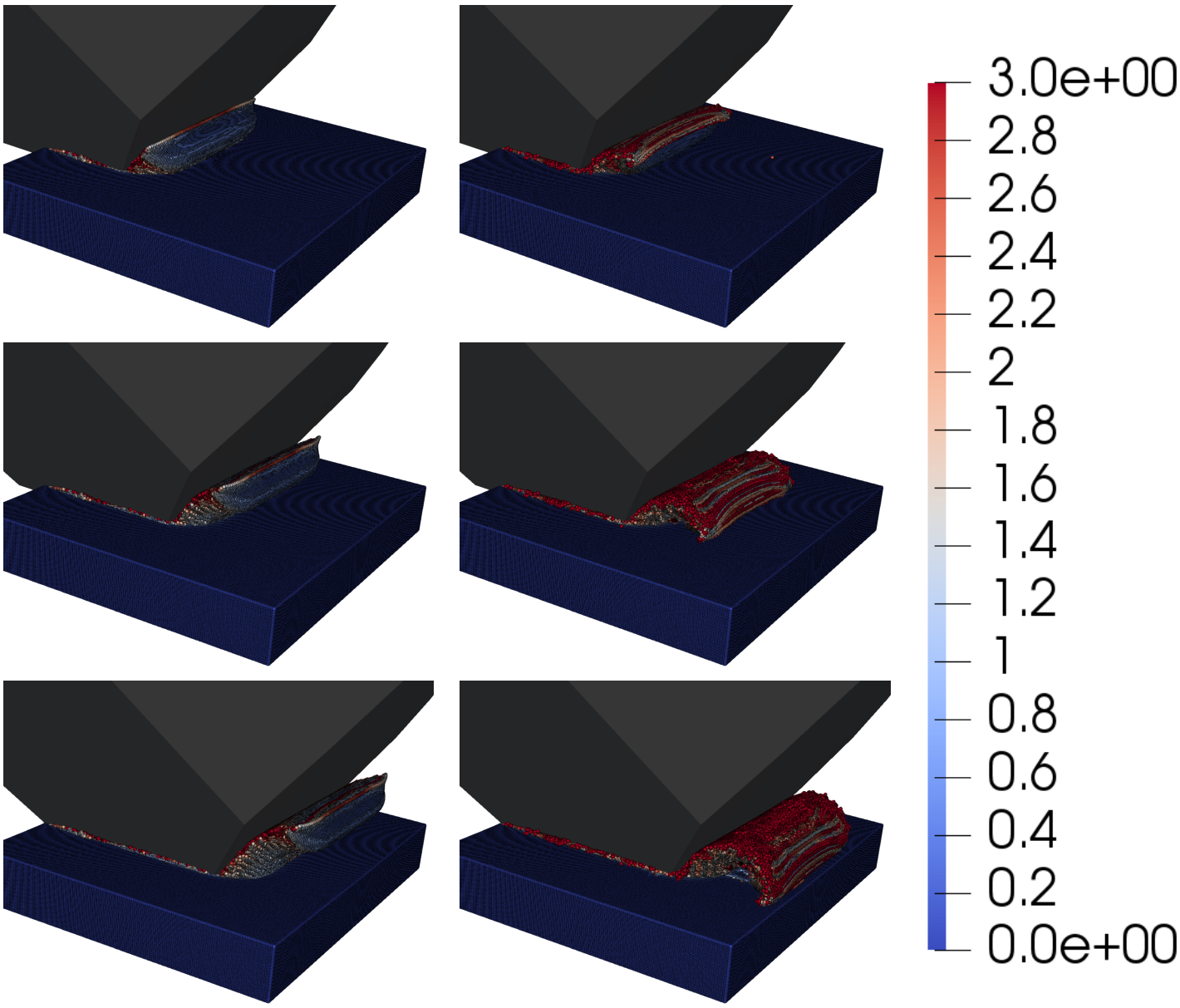

Figure 7.75: Simulation set ups A and D are re-run with Diamond 2. SPH (A) to the left, CSPM including artificial stresses (D) to the right, simulation progress top to bottom. Chip curling can now be observed using CSPM.

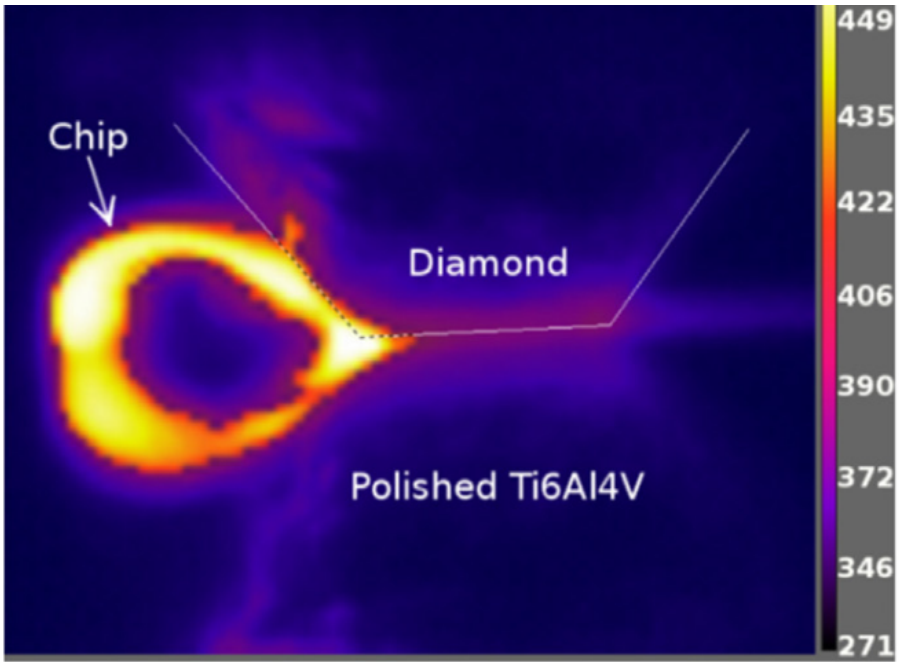

Figure 7.76: Picture of the single grain cutting experiment, taken from [3]. A curled chip is observed

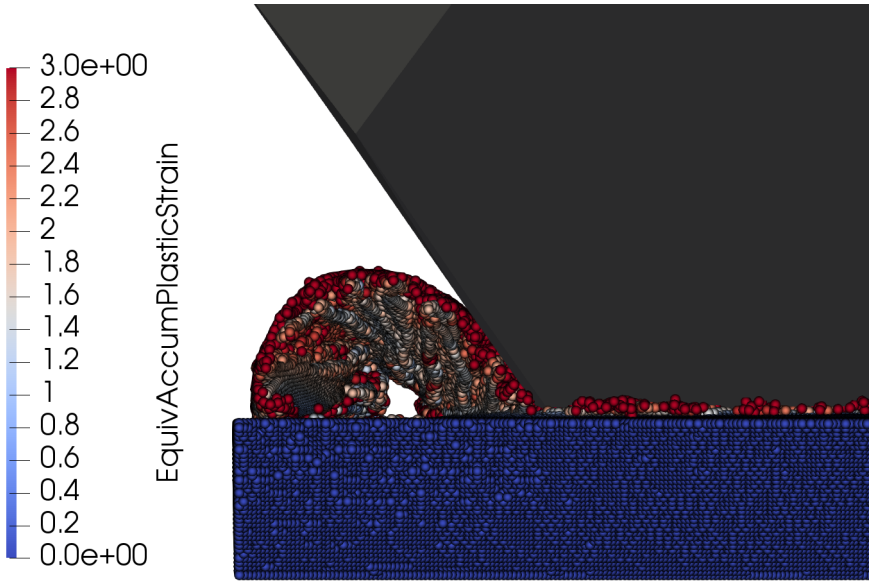

Figure 7.77: Simulation set up D, using Diamond 2. A close up allows for better comparison with experiment 7.76.

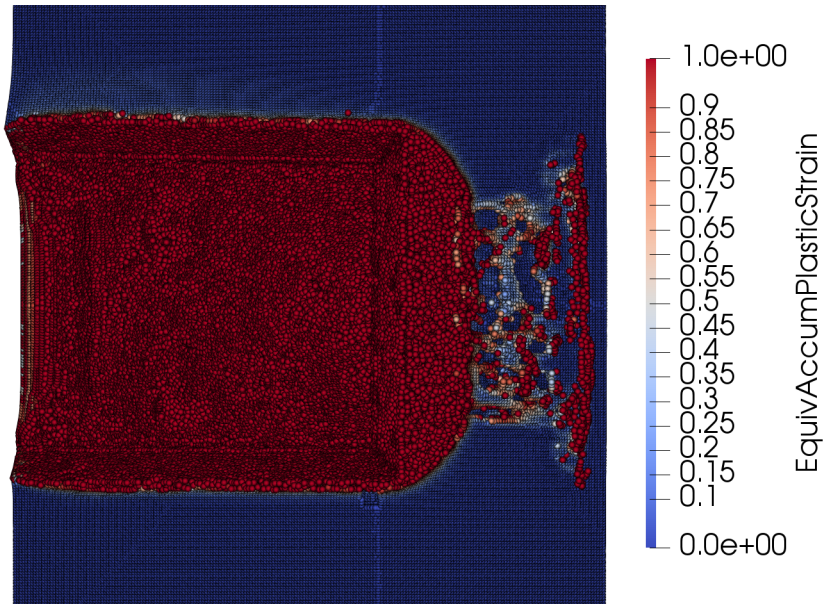

Figure 7.78: Simulation set up D, using Diamond 2: The chip is clipped from the visualization and a top view is shown. The chip curling is so intense that the chip slides along the workpiece.

7.6.2.2 *Testing Features of the Thermal Model*

In this section, the following configurations are investigated:

Label	Ad. Heating	T. Cond.	T. Cond. Tool	Heat Fric.
A	✗	✗	✗	✗
B	✓	✗	✗	✗
C	✓	✓	✗	✗
D	✓	✓	✓	✗
E	✓	✓	✓	✓

the geometry was kept identical to the previous section, i.e. W_{wp} and L were set to $2 \cdot W_{\text{tool}}$, see figure 7.70. The simulation resolution was kept a little more moderate by setting n_{seed} to 12, resulting in rougly 500'000 particles. Diamond 1 is used again, with thermal properties given in table 7.9. Thermal boundary conditions can be seen in figure 7.79. XSPH, artificial viscosity and artificial stresses were turned on.

Table 7.9: Relevant thermal properties of diamond.

Parameter	Value	Unit
Density ϱ	3'530	kg/m^3
Specific heat capacity c_p	630	$J/(kgK)$
Thermal conductivity k	2200	$W/(mK)$

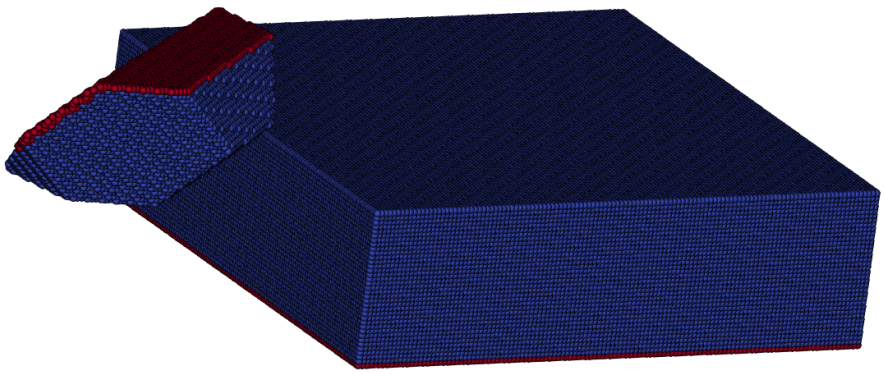

Figure 7.79: Thermal boundary conditions; particles colored in red, seen at the fixture of the diamond and on the bottom of the workpiece are held at constant 300 degrees Kelvin.

Just as in the last section, the first subject of investigation is the chip shape produced by the various simulations. This is done in figure 7.80. The first observation to be made is that the tool is now filled with particles. Those are rigid, but are subject to the thermal solver, using the Brookshaw approximation (4.70). Next it can be seen that the chip shape is dependent on the thermal model. The chip with no thermal conduction exhibits the largest amount of thermal softening, resulting in a straight flow chip. Thermal conduction throughout the workpiece, then the tool reduces thermal softening. The results with and without heat conduction into the tool look quite similar, even though diamond has very high thermal conductivity. This is because the cutting distance considered has been quite short. Finally, simulation E shows that heat conduction due to friction is quite severe.

The heating of the diamond is again highlighted in figure 7.81. The friction generates large amounts of heat which then almost immediately flow into the diamond. It is again noted that only a very short cutting distance is considered, making the rapid heating of the diamond all the more remarkable.

Finally, the cutting forces are shown in figure 7.82. This time, a more uniform picture is painted. Introducing thermal softening by adiabatic heating reduces the forces somewhat. Additional features, which in turn reduce the amount of thermal softening, do no change the forces in any meaningful capacity anymore. The presence of thermal softening seems much more important than the amount of it, i.e. how much the thermal softening is lessened by heat conduction or increased again by frictional heat.

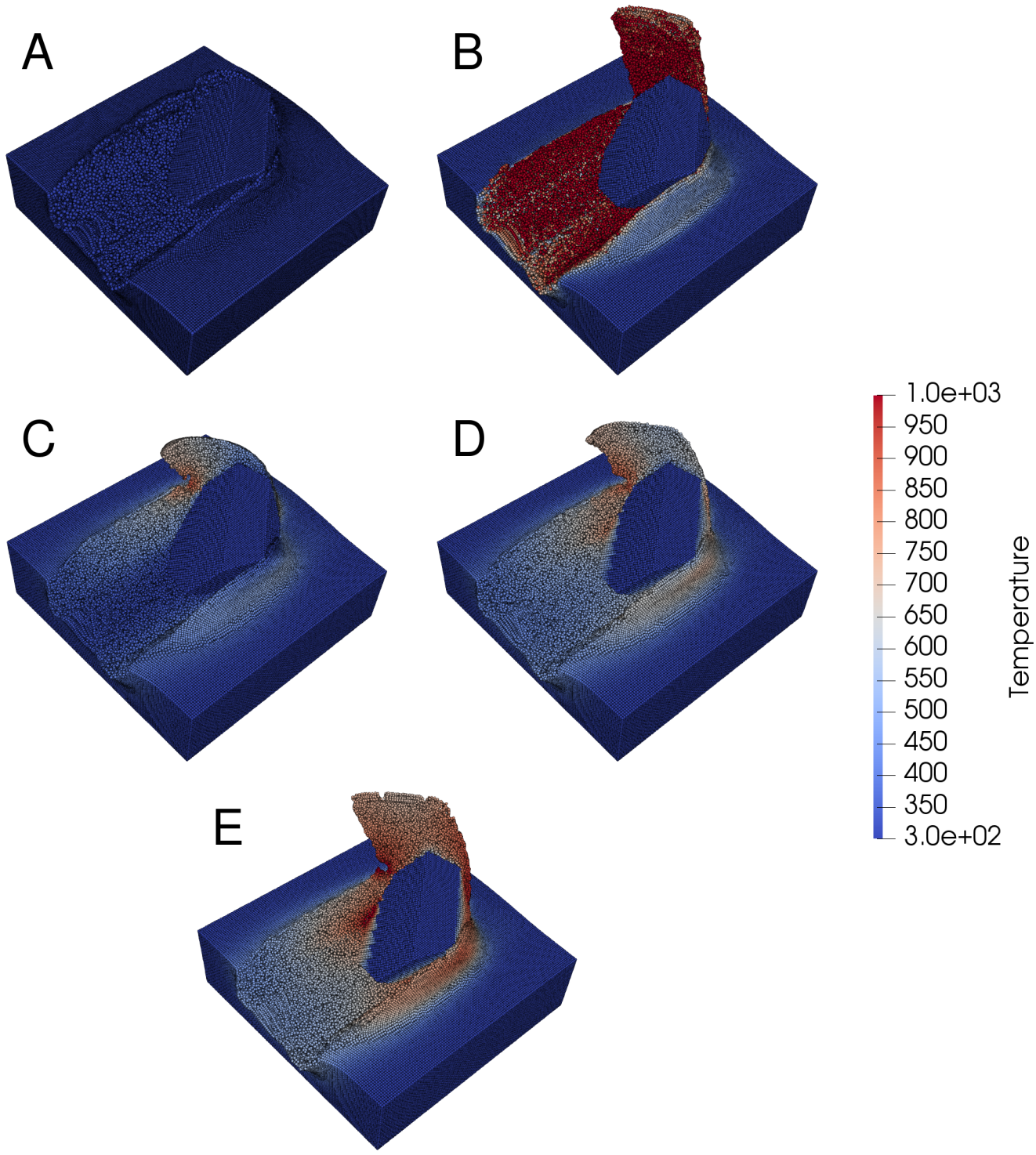

Figure 7.80: Features of the thermal model. See beginning of this section for specification.

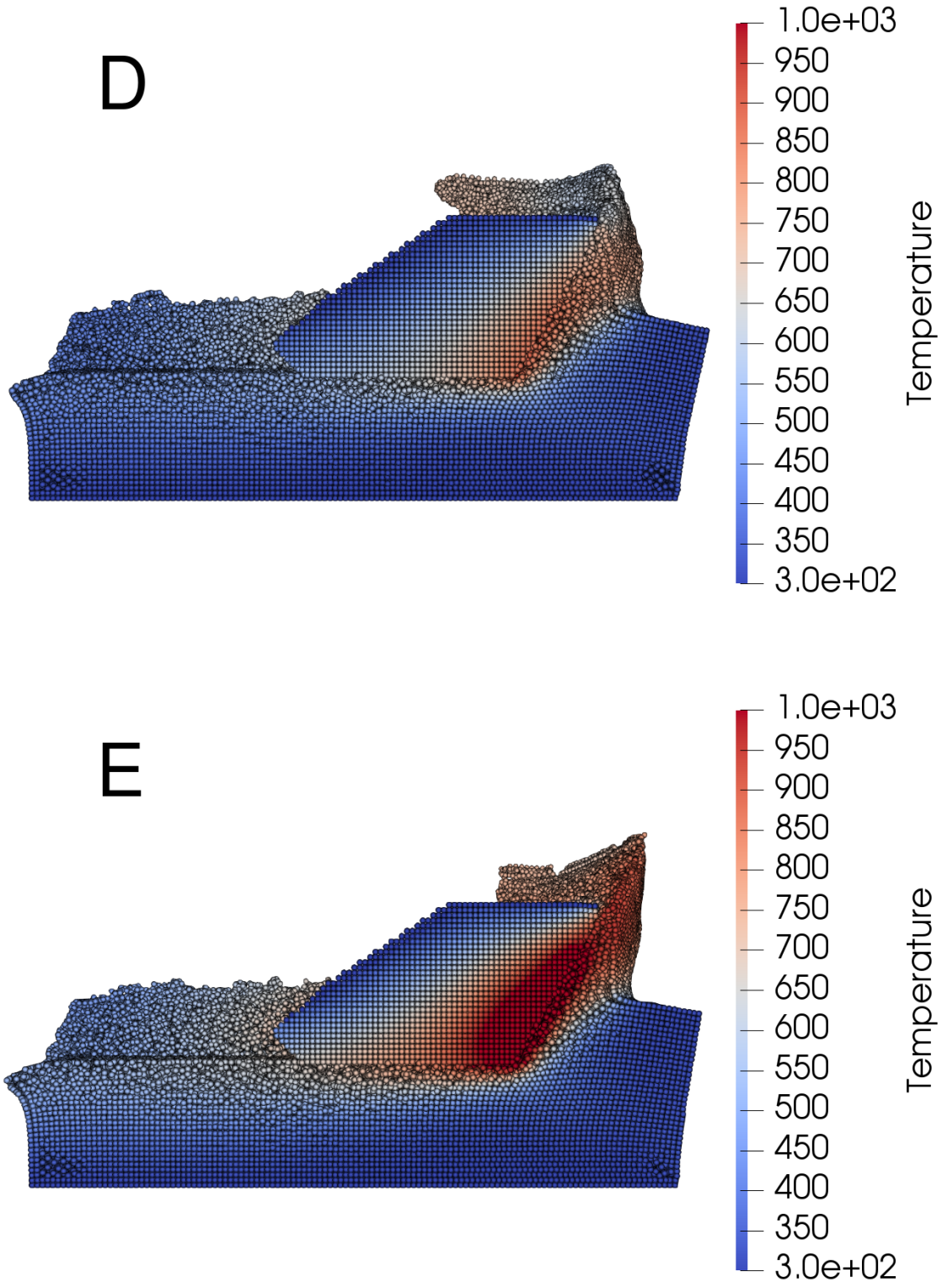

Figure 7.81: Cross sections of simulation D and E, top and bottom.

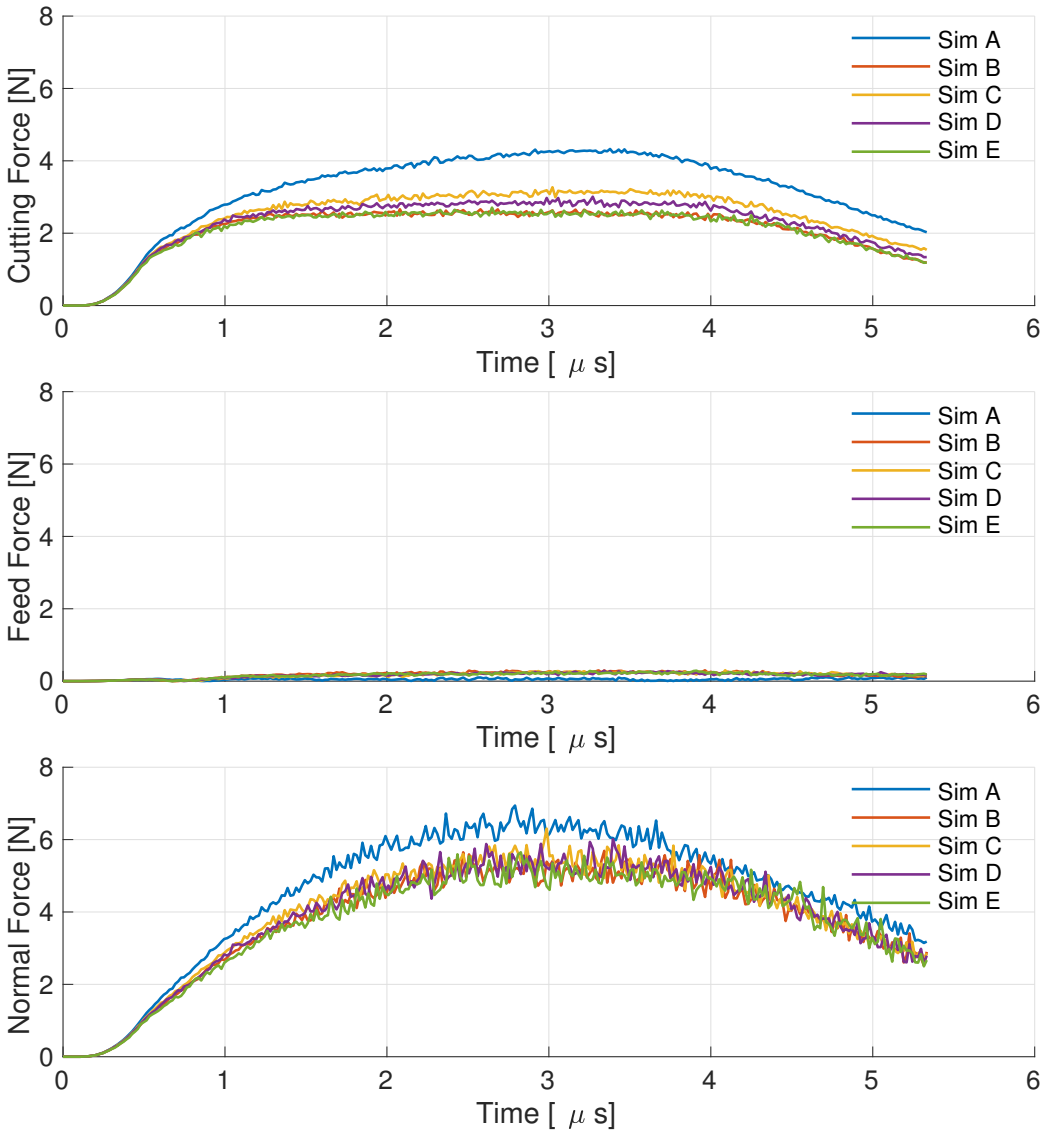

Figure 7.82: Cutting, feed and normal forces (top to bottom) of simulations A-E. Absolute values are plotted, from the tool perspective, the normal forces point upward in figure 7.70.

7.6.2.3 *Interaction Between Grains*

So far, only single grains have been simulated. It is clear that grains interact during the cutting operation of the EGT in the sense that grains interact with material that has been worked on by a leading grain, thus hardening and/or heating the material. Hence, it is interesting to simulate multiple grains to study the effect of different grain patterns and try to gauge the effect of pre-hardened material and material removal rate. To investigate such effects in a meaningful way, thermal steady state conditions need to be ensured. Since this is hardly possible using multiple grains, a low resolution sentinel simulation over a cutting distance of $\approx 0.6cm$ was run using a single grain. In figure 7.83 it is shown that this simulation reaches steady state with regard to tool temperatures.

Now, this sentinel simulation was performed using low resolution because of the very high cutting length, while the interaction simulations are to be performed using at least moderate resolutions. This was overcome by upsampling the steady state temperature from the low resolution tool to the high resolution one using simple SPH interpolation (4.12). This is illustrated again in figure 7.84.

In a first step, three patterns, denoted simulation A-C are investigated. These can be seen in figure 7.85.

Simulation A features non overlapping cutting paths, i.e. if the trailing grains run into pre-hardened material it is solely due to burr formation of the leading grains. Situation B narrows the gap by a factor of two, leading to a situation where the trailing grains encounter a lot of heavily deformed material. Both of these patterns would be highly unusual in real EGTs, see also section 7.6.3, they serve a illustrative purpose only. Finally, the third pattern is equivalent to situation B, but with the front middle grain missing. The purpose of this simulation is to represent the situation where one grain is ripped out of the binding material of the EGT, a situation which may occur in reality, often leading to chain reactions, ripping out multiple grains in succession, see [202].

As for the simulation geometry, W_{wp} was increased to accommodate all the diamonds to $W_{\text{wp}} = 6 \cdot W_{\text{tool}}$, while the feed was increased to $20\mu m$ to ensure deep cutting groves and significant burr mass. n_{seed} was further reduced to 9 to keep simulation times in check, resulting in about 500'000 particles.

Result frames are given in figure 7.86, and cutting forces are shown in figure 7.87. The top view of the result frames permits a clear view of the grain paths, showing 5 or 4 cuts, respectively, depending on the arrangement. The forces are as expected, showing equal forces for the leading grains in all cases. The forces of the trailing grains in situation A and B is equal, as expected, and lower in situation B, demonstrating that the reduced cutting area outweighs the influence of the hardening by quite a bit.

The goal is now to quantify that influence, i.e. try to decompose hardening and softening effects. To achieve this, various model features are either activated or deactivated. A leading grain from any of the situations A-C serves as the reference, denoted "Force I". To assess the effect of the reduced cutting area, the state variables of the material meeting a trailing

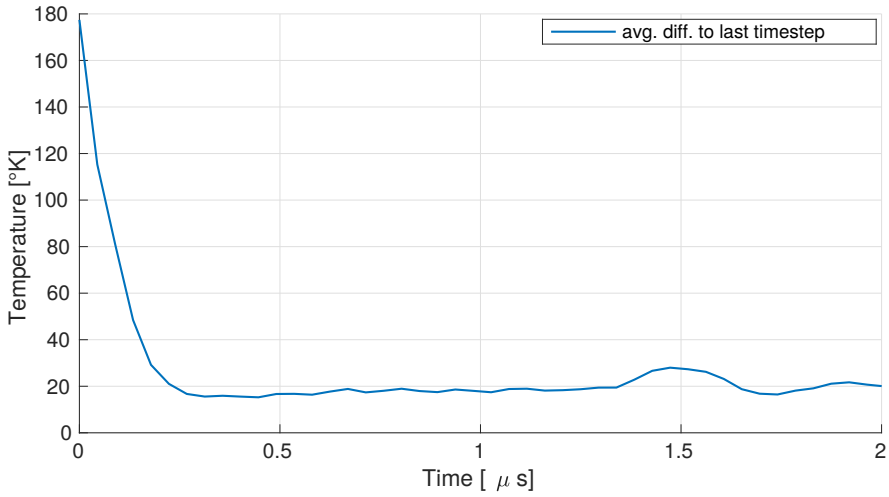

Figure 7.83: Recording average difference to previous timestep in the low resolution sentinel simulation. As can be seen, steady state is reached.

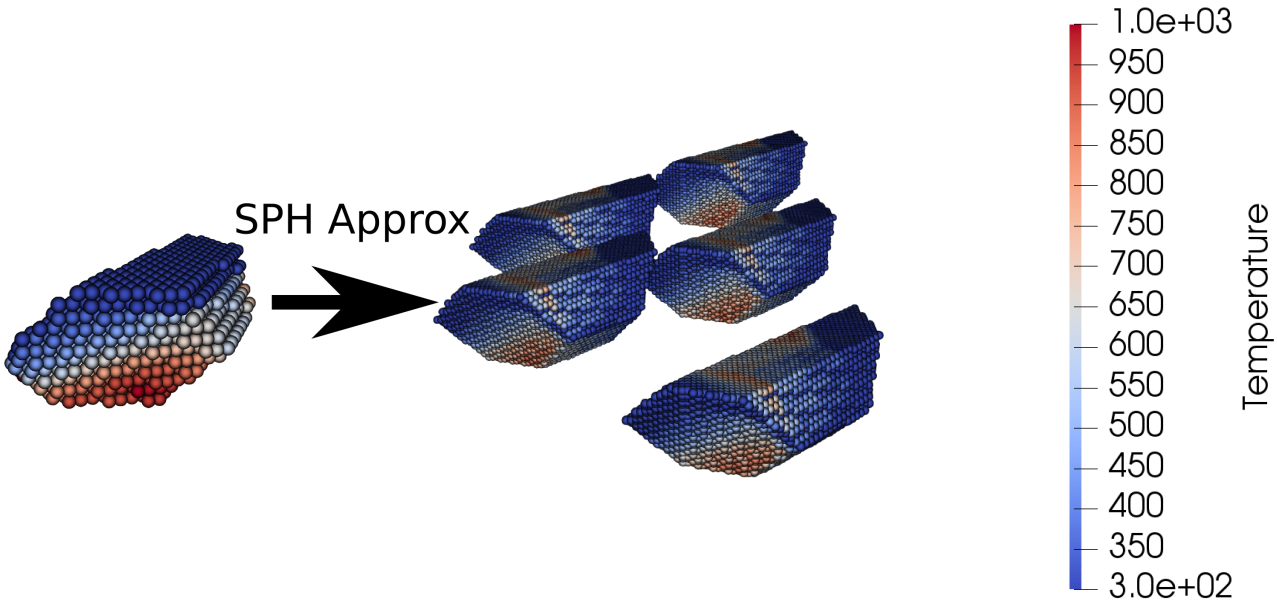

Figure 7.84: Preparing high resolution tools for multigrain simulations using the thermal field from a low resolution sentinel simulation using SPH approximation.

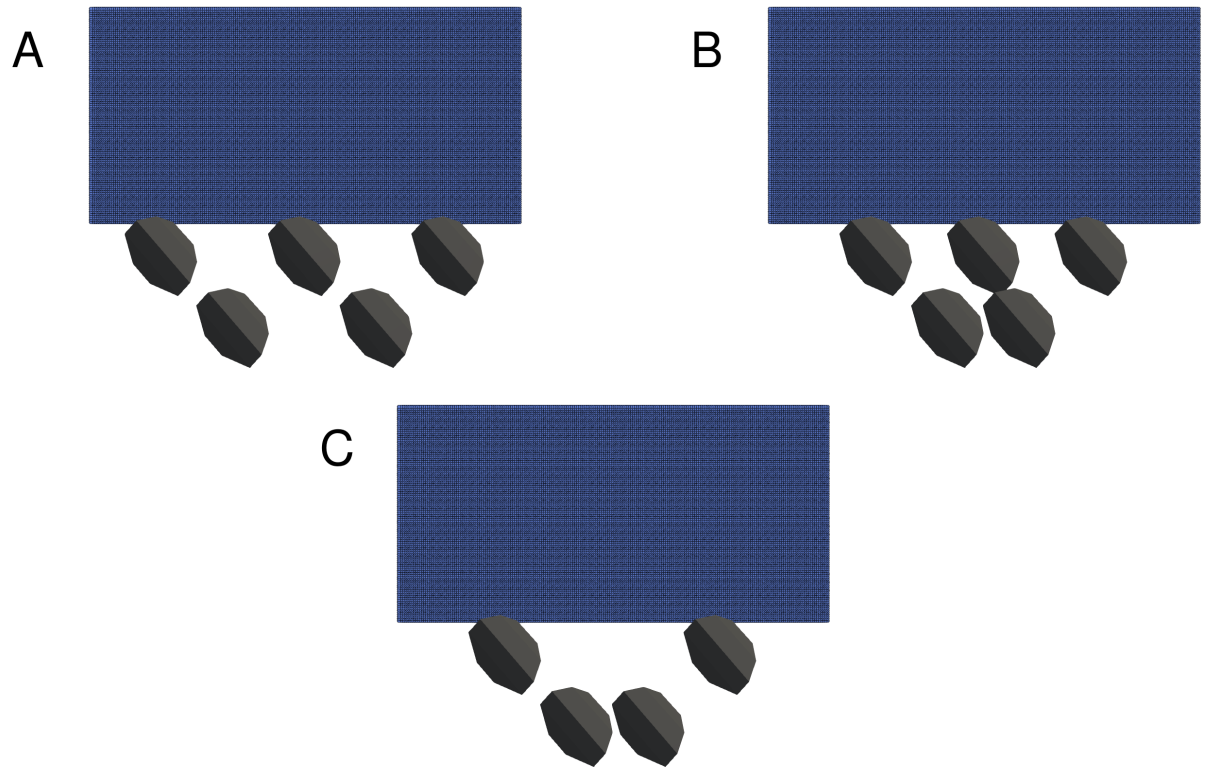

Figure 7.85: Initial conditions for simulations A, B and C.

grain were reset; thermal softening, strain- and strain rate hardening were reset to zero. This is "Force II", simulating the effect of reducing the cutting area. See figure 7.89 for an explanation how this was achieved. Afterwards, only strain and strain rate hardening were activated for the trailing grain, resulting in "Force III", and finally, thermal softening was added for the trailing grain, completing the plastic model again and resulting in "Force IV". The results of this analysis can be seen in figure 7.88. It is especially remarkable how strong the thermal softening effect is compared to the strain- and strain rate hardening.

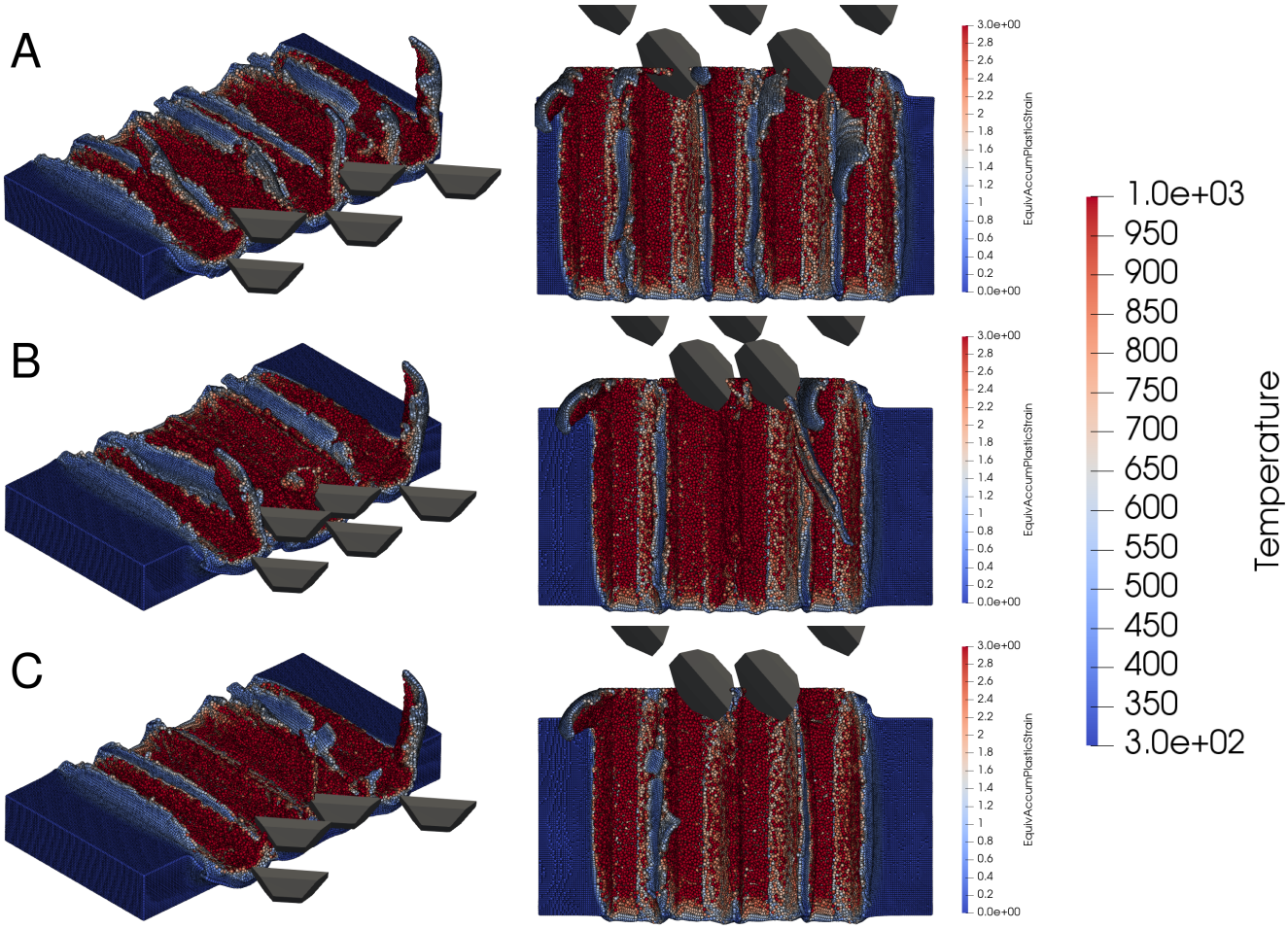

Figure 7.86: Simulations A - C. Left side shows an isometric view at the end of the simulation, right side shows a top view.

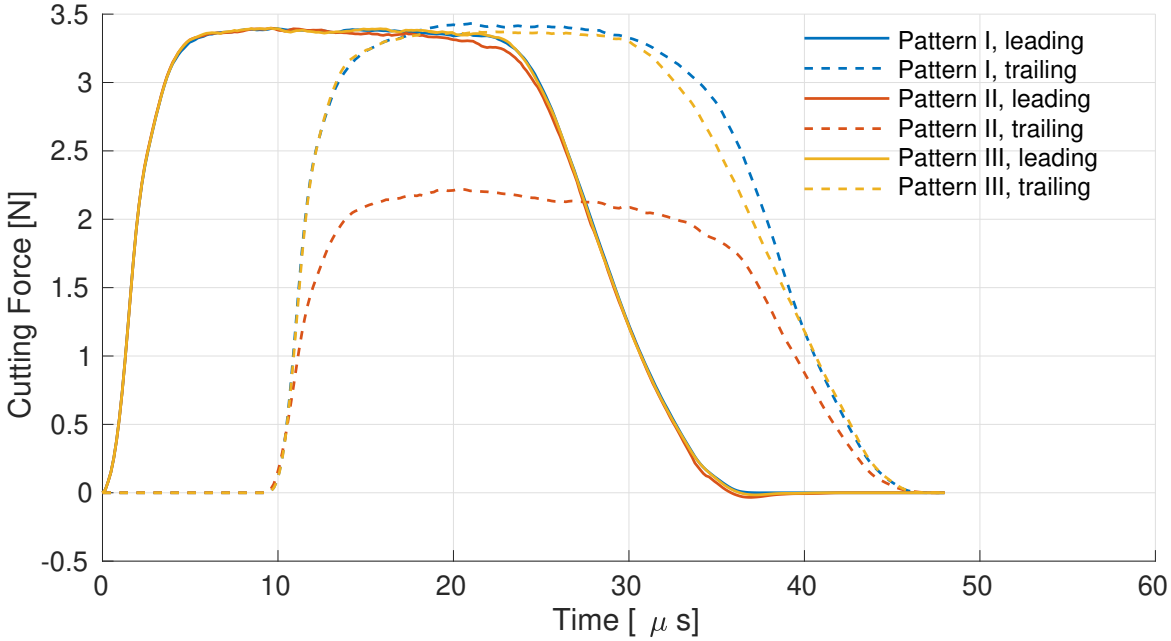

Figure 7.87: Cutting forces for simulations A-C.

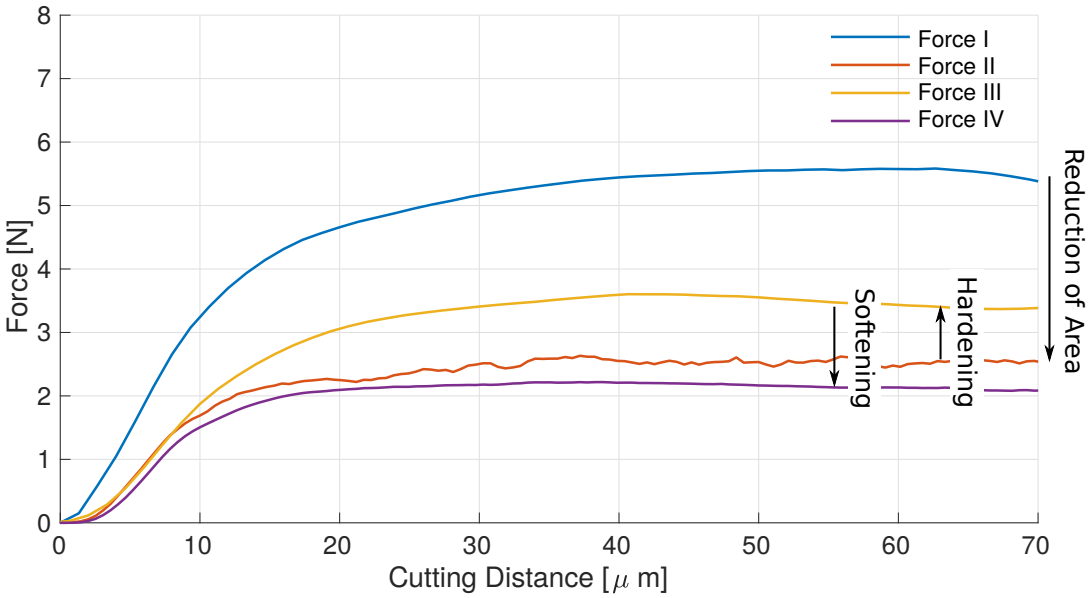

Figure 7.88: Decomposing the cutting force according to the scheme described.

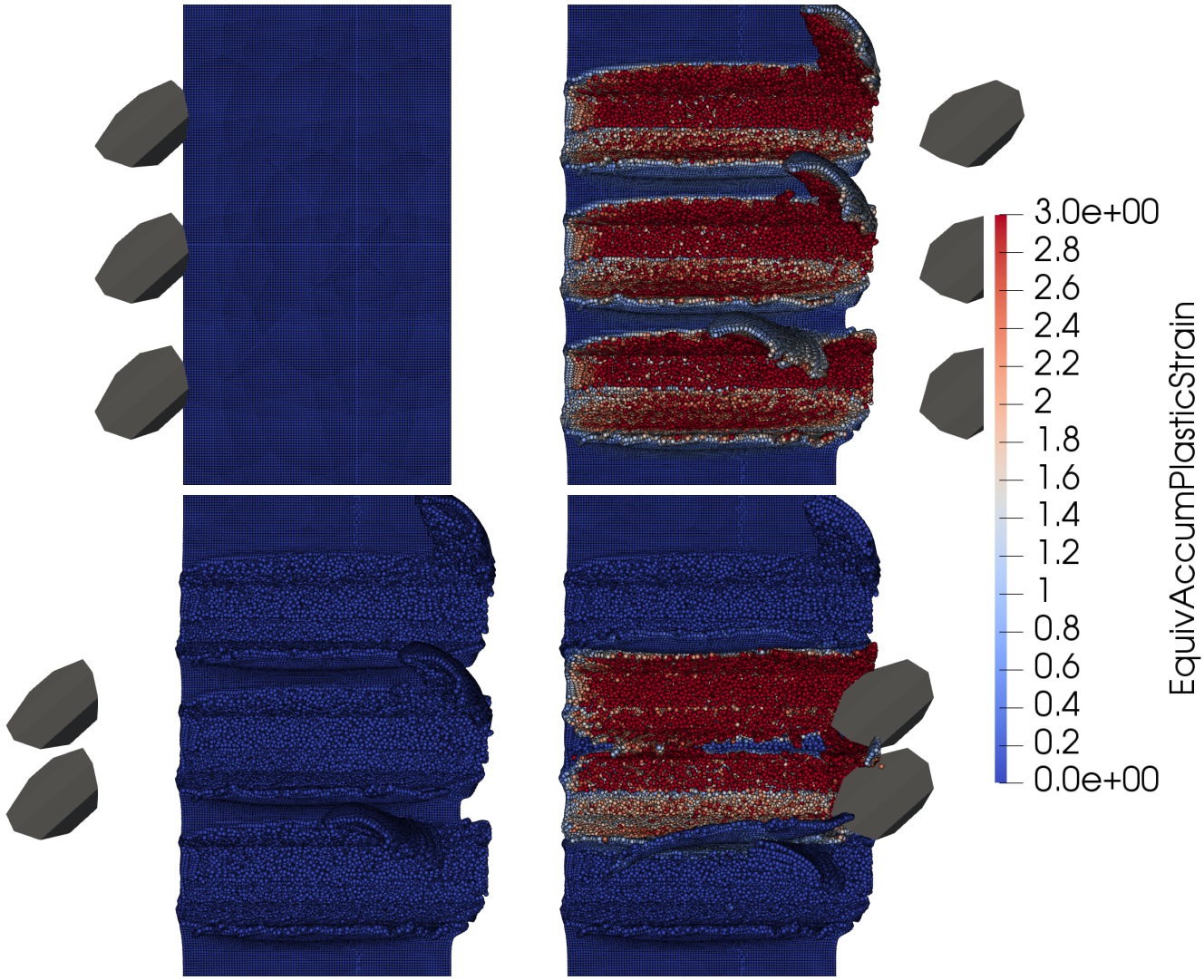

Figure 7.89: Initial conditions for cut with 3 diamonds (top left), last frame for cut with 3 diamonds (top right), which serves as geometrical initial conditions with reset state variables for the second cut with 2 diamonds (bottom left), final frame for simulation with 2 grains (bottom right)

7.6.3 *Simulations of the full grinding wheel*

While previous sections investigated scaled grains it was observed that for moderate simulation resolutions, well in line with previous results [227], simulating the full wheel becomes possible, if the wheel diameter is scaled along with the diamond with a ratio of 1:10. This offers the possibility to simulate some full wheels with different grain patterns, and evaluating these grains patterns with respect to grain wear.

To construct the wheel the parametrization given by [202] was employed, in somewhat simplified fashion, e.g. the rotation of the diamonds was left constant. The parametrization is two staged: first the pattern for arranging the diamond grains is defined, then the shapes of these diamonds. The parametrization for the pattern is described in figure 7.90, and the values used, taken directly from [202] are reproduced in table 7.10.

The shapes of the diamonds were determined by a blending of tetra- and octohedra, using a shape parameter C_f, that was uniformly sampled from $[0 \dots 1]$. This represents the sampling `ABN800` by manufacturer Diamond Innovations (now, as of 2019, acquired by Sandvik). The sampling of the diamond grains is again illustrated in figure 7.91. The resulting wheels are exhibited in figure 7.92.

The simulation set up is sketched in figure 7.93. As mentioned, the tool diameter is scaled from the 5mm given in [202] to 0.5 mm. The tool velocity in horizontal direction is 1.8 m/s. The rotational velocity was set such that the wheel completes a full revolution during $t = \underline{v}_{\text{tool}}/L$. Using this scaled geometry the RPM are unrealistically high at about 54'000, but the resulting cutting speed of about 280 m/s remains consistent with high velocity grinding. To make the simulations feasible with regards to runtime, a small box with a length of about 5 times the typical diamond edge length was placed into the domain, and only particles inside this box are active. The box travels with the same horizontal velocity as the tool. n_{seed} was set to 4, resulting in about 70'000 particles in the box, at any given time.

To measure the grain wear, the Usui wear model (7.10) known from section 7.5.3.1 is employed again. That is, the average accumulated wear per diamond is recorded. The same parameters $\xi = 2500\text{K}$ and $\Psi = 7.8 \times 10^{-9}\text{Pa}^{-1}$ from [197] are employed. Note that these parameters are not appropriate with regard to the tool material. However, since the simulations are only compared between each other, this is not relevant for the purpose at hand.

An example frame of the simulation using Pattern B is shown in figure 7.94. As discussed, only a small section of the work piece is simulated at any given time. The diamonds were

Table 7.10: The patterns used to generate the grinding tools, taken from [202] without modification.

-	Δx [mm]	Δz [mm]	Δz_2 [μ]	α [rad]
Pattern A	2.0	0.5	3.2	$\pi/3$
Pattern B	1.0	0.5	1.6	$\pi/3$
Pattern C	0.5	0.5	0.8	$\pi/3$

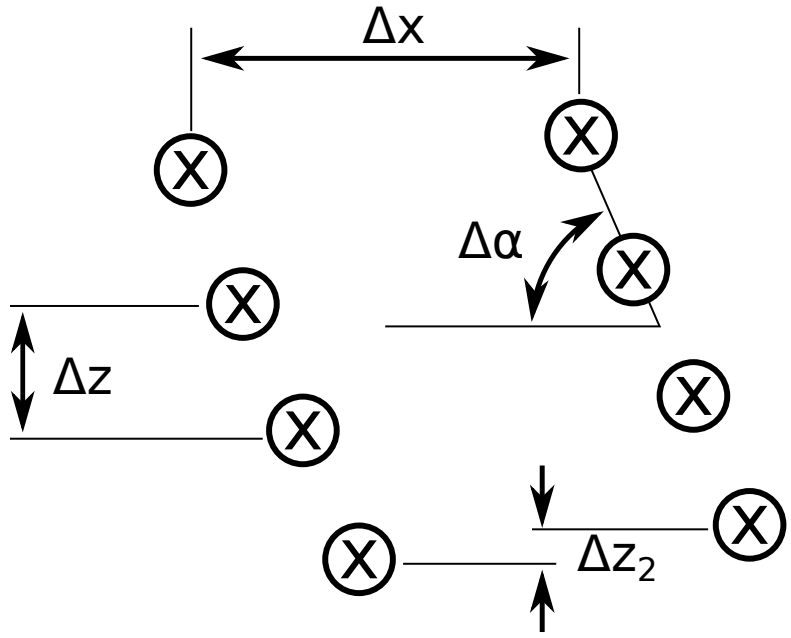

Figure 7.90: Parametrization of the grinding tool according to [202]. x determines the locations of the diamonds.

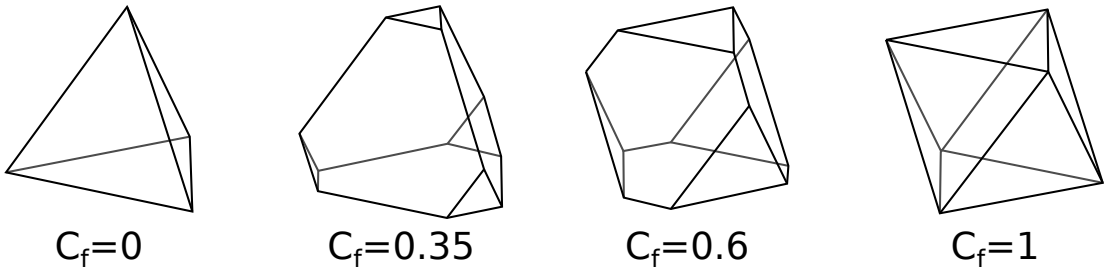

Figure 7.91: Blending between tetra- and octahedrons to obtain the diamond shapes. C_f was randomly sampled from $[0 \dots 1]$.

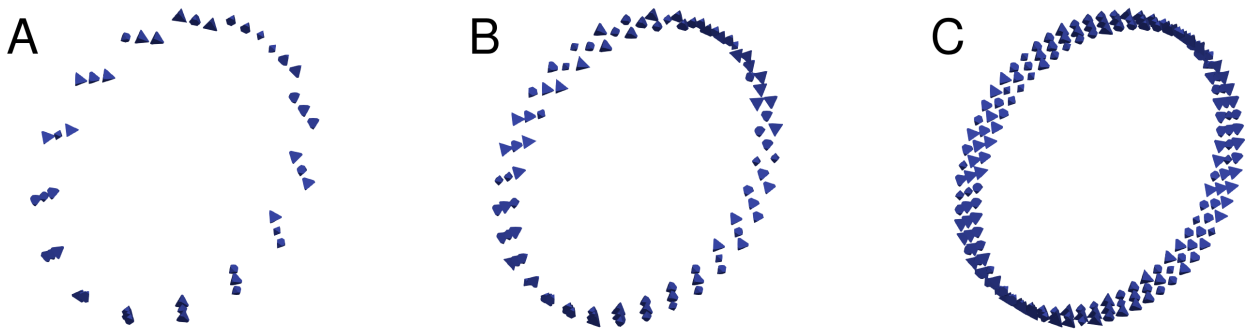

Figure 7.92: Resulting wheels using Patterns A-C.

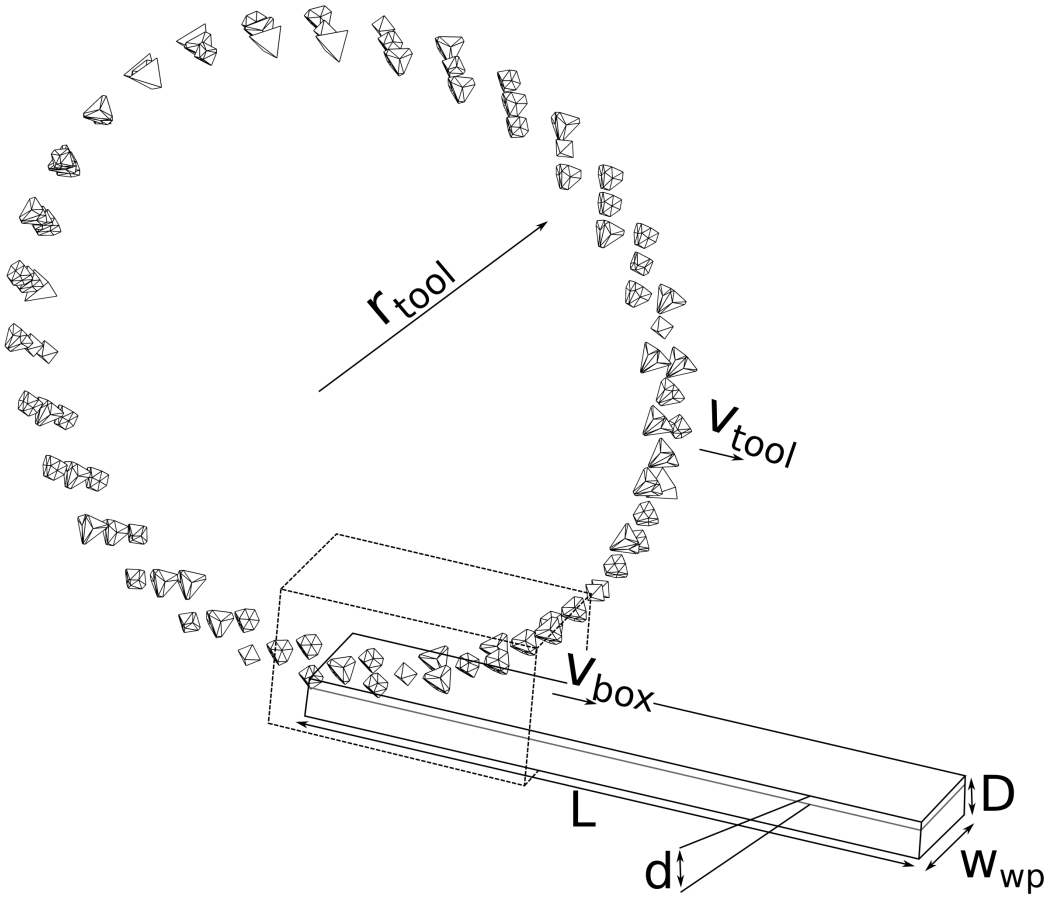

Figure 7.93: Sketch of the simulation set up of the full wheel simulations. $L = 2 \cdot r_{\text{tool}}$, $D = 3 \cdot d$. Only particles inside the box are active. The box travels with $\underline{v}_{\text{box}} = \underline{v}_{\text{tool}}$. d is set to 65% of the edge length of the diamonds.

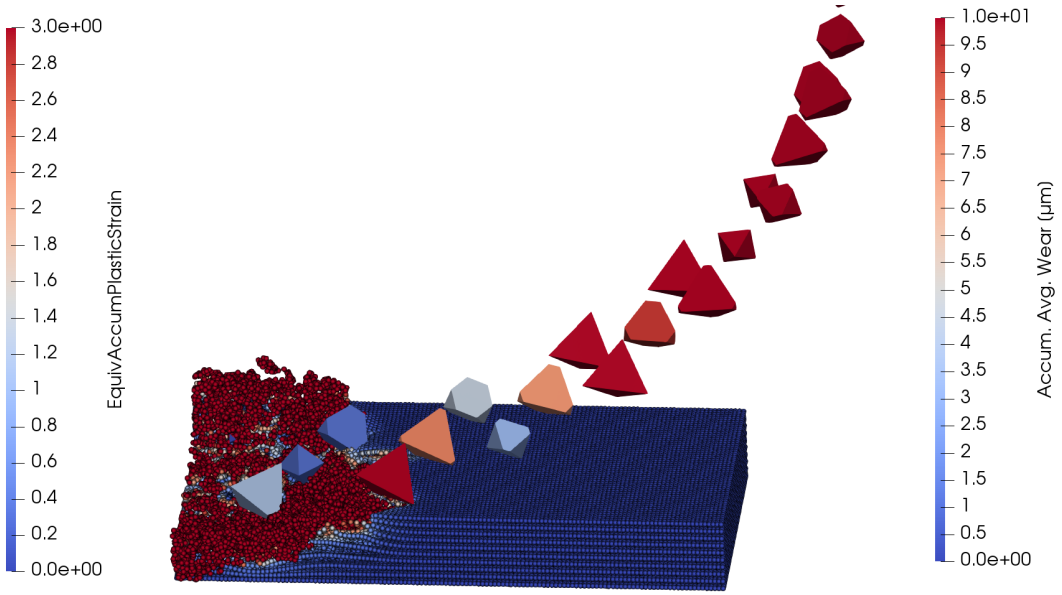

Figure 7.94: Example frame of a full wheel simulation. The grains are colored according to how badly they were damaged during the interaction with the work piece.

Table 7.11: Average accumulated wear, averaged over diamonds

Pattern	Avg. Accum. Wear [μ m]
A	10.0
B	8.4
C	4.4

colored according to their accumulated wear. Figures 7.95 show the final wear distribution for all diamonds on the wheel. The wear patterns conform to expectations that there is less wear per diamond in the more densely populated patterns because each grain is responsible for less material to be removed. This is further confirmed by table 7.11. It can also be seen from figures 7.95 that there are some clear outliers in each pattern. Figures 7.96 gives some insight into the mechanism responsible for this. Finally, figure 7.97 exhibits the number of grains in contact at any given time to enable some additional insight into the material removal process.

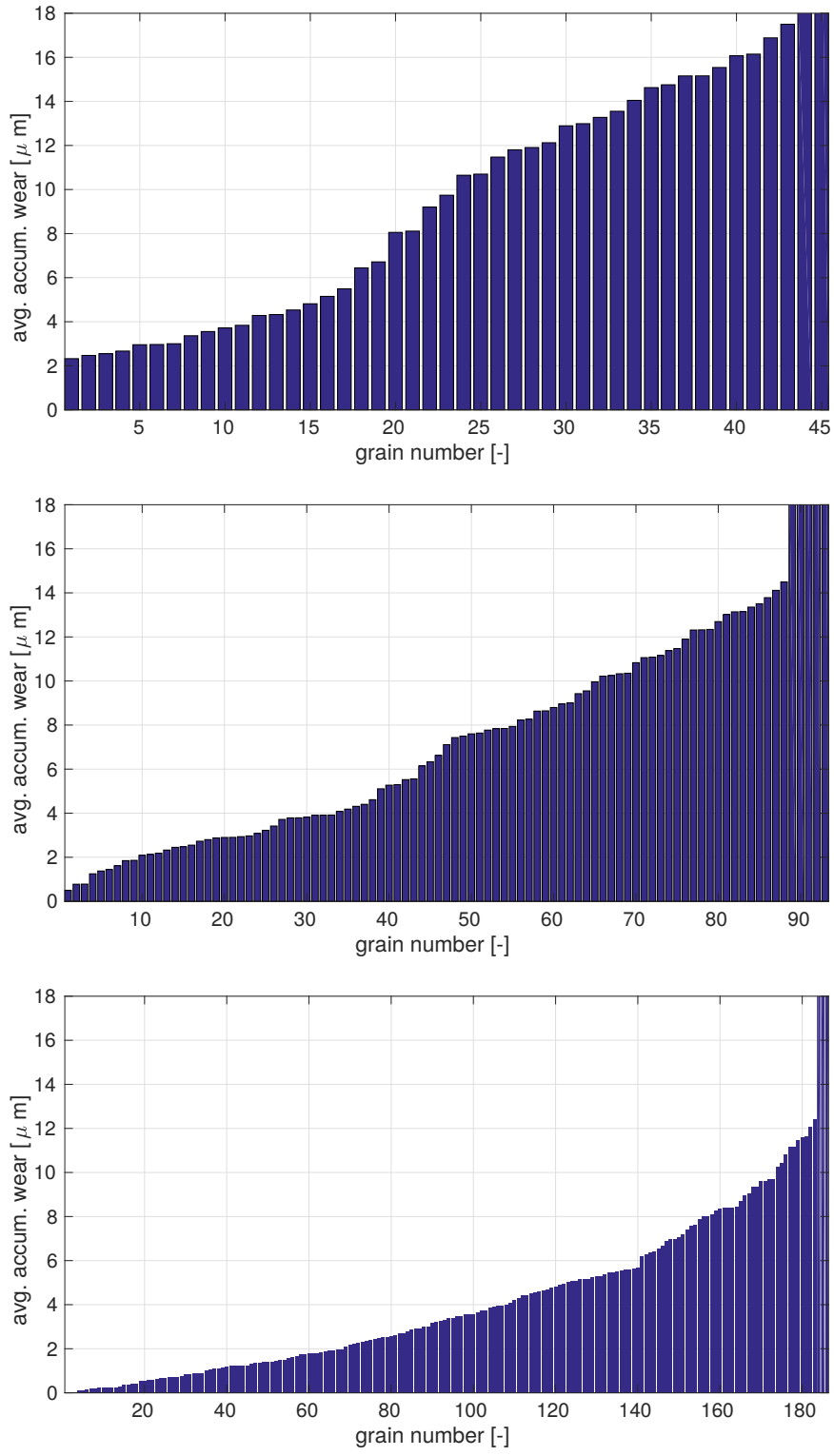

Figure 7.95: Wear patterns as exhibited by simulations / patterns A-C, top to bottom. See figure 7.96 for an explanation for the outliers and figure 7.97 for an explanation for the kink in pattern C around grain number 140.

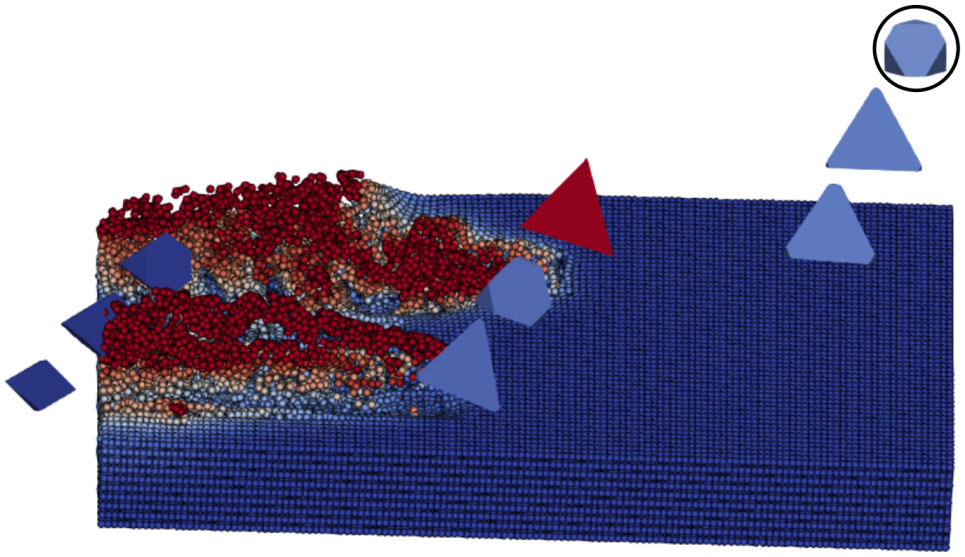

Figure 7.96: This figure shows an investigation on how the very worn outliers in figure 7.95 develop. As can be seen, a small grain (circled) fails to remove any material at all and is unworn. A quite large grain then follows, having to remove a lot of material, which may also be pre-heated and hardened, causing a disproportionate amount of wear.

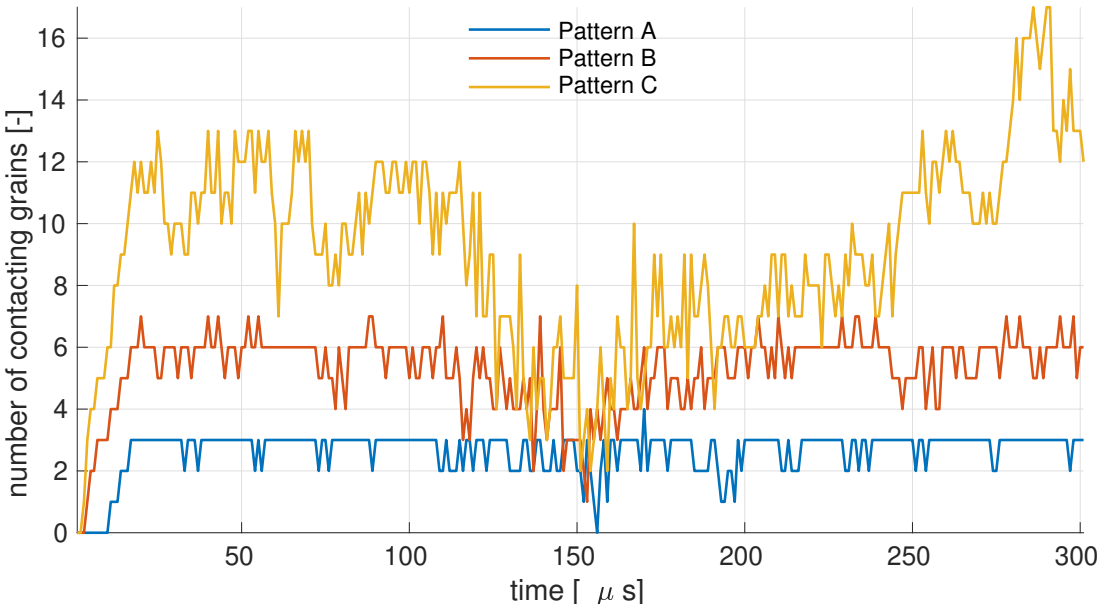

Figure 7.97: Number of contacting grains over time in patterns A-C. As can be seen, patterns A and B show a very even number of diamonds in contact at any given time, while pattern C shows a drop in the middle of the simulation. This means that the feed velocity is too low with respect to the RPM for this densely populated wheel. This also explains the jump around grain number 140 in figure 7.95.

8

CONCLUSIONS AND OUTLOOK

The simulation of metal cutting is a challenging undertaking, especially for the state of the art simulation method in solid mechanics, i.e. the FEM. Topological changes by material separation and extremely large deformations require a constant remeshing of the computational grid. This prompts various numerical issues and causes very long runtimes, requiring the use of super computers to obtain results in acceptable time frames.

A different class of numerical methods, the meshless, or particle methods, is not limited in the amount of deformation it can represent. They are thus an excellent alternative to the FEM in metal cutting, which is not only supported by quite a few publications with this regard, but also by the implementation of the SPH, the most well known member of the family of meshless methods, into commercial programs `LSDYNA` and `ABAQUS`.

Still, the currently available meshless codes, be it presented in literature only, open source or commercial, miss key features. Either using severely limited material modeling, lack contemporary stabilization techniques or do not use advanced, i.e. linearly complete basis functions. Another resource which was untapped so far is the parallelization of meshless codes using GPUs, at least in the context of meshless metal cutting simulations.

This thesis closes these gaps. A wide array of meshless techniques was implemented. These methods are subjected to rigorous benchmarking to determine the most suitable methods with regard to metal cutting. Having determined a strong arsenal of algorithms, these are then exploited to obtain a number of results previously not available to researchers. While not exhaustive, the following list quickly restates some of the key results:

- The only recently introduced Transport Velocity Formulation for solids [282] (TVF) was implemented in 3D for the first time. It's viability was demonstrated using colliding rubber spheres and cylinders
- The TVF was adapted for metal cutting. The first metal cutting simulations using the TVF and artificial stresses proposed by Monaghan [91] are shown. Cutting forces and chip shapes are predicted with reasonable accuracy, even at low simulation resolutions
- The viability of a smooth contact algorithm in metal cutting was shown. It has to be noted that similar results are also available in [246], [244].
- The first proof of concept metal cutting simulations in the total Lagrangian frame are shown, be it using meshless methods or the FEM.
- The 2 dimensional GPU accelerated solver was employed to optimize a metal cutting process completely autonomously. This shows that studies similar to the one in [137] can be performed at a fraction of the hardware costs. In the same context, a thermal

model which is able to establish thermal contact between tool and workpiece was introduced. This is novel to meshless metal cutting methods.

- The same solver was used to perform orthogonal metal cutting simulations at unprecedented resolutions, enabling detailed reproduction of the chip shape, which was previously often cited as a weak point of meshless methods.
- The solver is shown to be 60 times more efficient than an equivalent serial implementation on the CPU. Time to solution compared to `LSDYNA`, `ABAQUS` is improved by an even larger factor, however, this due to the fact that in these solvers orthogonal metal cutting can only be represented using three dimensional geometry.
- The three dimensional solver is used to simulate the especially demanding single grain cutting process, featuring large negative rake angles. While being done before in [3] using FEM and in [227] using SPH, the simulation resolution is increased 25 fold, while lowering the time to solution by a factor of 6.
- The tremendous numerical performance of the 3 dimensional metal cutting solver on the GPU enables some simulations previously not possible using meshless methods at acceptable runtimes, like the simulation of diamond grains at experimental scale, interactions of multiple, realistic diamond grains, and even concept simulations of the full grinding wheel.

It is worth noting that, at the time of writing, a lot of the codes developed for this thesis are either already open source, or to be open sourced upon publication of the subject matter in a journal. This is in stark contrast to most of the work available, that either relies on commercial solvers, uses in-house codes which are kept private, or offers source code only on a per collaboration basis.

Future research on the subject can go into a lot of directions. Some suggestions include:

- Total Lagrangian formulations offer many beneficial stability and runtime properties over their updated Lagrangian peers, which this thesis finally settled on. Combination with proper geometry tracking techniques, like the Particle Level Set methods, could enhance the proof of concept simulations exhibited in this work to proper simulation results.
- The purely meshless approach could be surrendered in favor of methods which rely on a simple, Cartesian background grid, like rSPH [38] or the Material Point Method [17]. While an implementation of that method is given in [6] for metal cutting, the runtimes reported are excessive, leaving a lot of room for improvements.
- Still on the subject of numerics, a lot of potential lies in combination of meshless methods with mesh bound methods, especially the ALE family of methods. Such hybrid methods could for example benefit from an ALE mesh around the tool tip for improved contact modeling.
- While the constitutive modeling in this thesis can be considered state of the art with regard to metal cutting, other areas of research use material models which are significantly more advanced, especially with regard to plasticity. Using the parallelized GPU code, complex material modeling would potentially be possible, at moderate runtimes.

- The short evaluation cycles enabled by GPGPU parallelization were exploited to optimize the metal cutting process with regard to cost in this work. However, the same mechanism could also be exploited to inversely fit material parameters to cutting experiments, making parameter identification directly in the metal cutting process a possibility for the first time. In light of the extreme strain rates encountered in metal cutting, and the relatively low strain rates typical for material tests, this would be a crucial development towards realistic material modeling.

9

APPENDIX

This chapter contains some lengthy proofs or other remarks which would distract from the discussion at hand, would they be incorporated into the main text.

9.1 POTENTIAL BASED FORCES

The intermediate goal is to find an expression for the term $\nabla_{\underline{u}_i} \underline{\underline{\varepsilon}}$. In this chapter, the symbol for the displacement field shall be $\underline{\mathcal{U}}$. Albeit slightly unconventional, this is to clearly distinguish between the displacement and it's first component u. The derivatives of the first component of the displacement field $\underline{\mathcal{U}} = [u, v, w]$ are given as:

$$\nabla u = \begin{bmatrix} u_{,x} \\ u_{,y} \\ u_{,z} \end{bmatrix} \approx \sum_j \nabla \overset{\circ}{W}_{ij} u_j \tag{9.1}$$

where $\overset{\circ}{W}_{ij}$ is the RKPM Kernel function. Note that the implementation in this thesis uses in fact Randles-Libersky corrected Kernel functions. However, the proof is only slightly more complicated if Randles-Libersky Kernel functions would be used.

For any given particle i we can now take the derivative of that expression with respect to the displacement $\underline{\mathcal{U}}_j = (u_j, v_j, w_j)$ at some point $\underline{x}_j \neq \underline{x}_i$. Note that this is the derivative with respect to the *symbol* u_j (and v_j, w_j, respectively) in the already discretized equation. Also, since a single point j is considered the summation over j is dropped (from notation only):

$$\frac{\partial}{\partial u_j} \nabla u = \nabla \overset{\circ}{W}_{ij} = \underline{d}_j \tag{9.2}$$

$$\frac{\partial}{\partial v_j} \nabla u = \underline{0} \tag{9.3}$$

$$\frac{\partial}{\partial w_j} \nabla u = \underline{0} \tag{9.4}$$

$$\tag{9.5}$$

Analogous results are found for ∇v, ∇w. The three components of $\underline{d}_j$ are designated as d_j^x, d_j^y and d_j^z. With this procedure, expressions for all combinations of $\frac{\partial}{\partial \alpha} \beta_{,\gamma}$, with $\alpha \in \{u, v, w\}$,

$\beta \in \{u, v, w\}$ and $\gamma \in \{x, y, z\}$ are obtained. It is now possible to find $\nabla_{\underline{u}_i} \underline{\underline{\varepsilon}}$. The six independent components of $\underline{\underline{\varepsilon}}$ are readily expanded as:

$$\begin{bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{zz} \\ \varepsilon_{xy} \\ \varepsilon_{yz} \\ \varepsilon_{zx} \end{bmatrix} = \begin{bmatrix} 2u_{,x} + u_{,x}\, u_{,x} + v_{,x}\, v_{,x} + w_{,x}\, w_{,x} \\ 2y_{,y} + u_{,y}\, u_{,y} + v_{,y}\, v_{,y} + w_{,y}\, w_{,y} \\ 2w_{,z} + u_{,z}\, u_{,z} + v_{,z}\, v_{,z} + w_{,z}\, w_{,z} \\ u_{,y} + v_{,x} + u_{,x}\, u_{,y} + v_{,x}\, v_{,y} + w_{,x}\, w_{,y} \\ v_{,z} + w_{,y} + u_{,y}\, u_{,z} + v_{,y}\, v_{,z} + w_{,y}\, w_{,z} \\ w_{,x} + u_{,z} + u_{,z}\, u_{,x} + v_{,z}\, v_{,x} + w_{,z}\, w_{,x} \end{bmatrix} \tag{9.6}$$

For the first component ε_{xx}:

$$\nabla_{\underline{u}_j} \varepsilon_{xx} = \begin{bmatrix} \underbrace{\frac{\partial}{\partial u_j} \varepsilon_{xx}}_{\text{I}}, & \underbrace{\frac{\partial}{\partial v_j} \varepsilon_{xx}}_{\text{II}}, & \underbrace{\frac{\partial}{\partial w_j} \varepsilon_{xx}}_{\text{III}} \end{bmatrix} \tag{9.7}$$

$$\begin{aligned} \text{I} &= \frac{\partial}{\partial u_j} 2u_{,x} + \frac{\partial}{\partial u_j} u_{,x}\, u_{,x} + \underbrace{\frac{\partial}{\partial u_j} v_{,x}\, v_{,x}}_{0} + \underbrace{\frac{\partial}{\partial u_j} w_{,x}\, w_{,x}}_{0} \\ &= 2d_j^x + 2d_j^x \cdot u_{,x} \\ &= 2d_j^x (1 + u_{,x}) \end{aligned} \tag{9.8}$$

$$\begin{aligned} \text{II} &= \underbrace{\frac{\partial}{\partial v_j} 2u_{,x}}_{0} + \underbrace{\frac{\partial}{\partial v_j} u_{,x}\, u_{,x}}_{0} + \frac{\partial}{\partial v_j} v_{,x}\, v_{,x} + \underbrace{\frac{\partial}{\partial v_j} w_{,x}\, w_{,x}}_{0} \\ &= 2d_j^x \cdot v_{,x} \end{aligned} \tag{9.9}$$

$$\begin{aligned} \text{III} &= \underbrace{\frac{\partial}{\partial w_j} 2u_{,x}}_{0} + \underbrace{\frac{\partial}{\partial w_j} u_{,x}\, u_{,x}}_{0} + \underbrace{\frac{\partial}{\partial w_j} v_{,x}\, v_{,x}}_{0} + \frac{\partial}{\partial w_j} w_{,x}\, w_{,x} \\ &= 2d_j^x \cdot w_{,x} \end{aligned} \tag{9.10}$$

Thus:

$$\begin{aligned} \nabla_{\underline{u}_j} \varepsilon_{xx} &= \begin{bmatrix} 2(u, x + 1) d_j^x & 2d_j^x v, x & 2d_j^x v, x \end{bmatrix} \\ &= 2\underline{\underline{F}}_x \cdot d_j{}^x \end{aligned} \tag{9.11}$$

Where $\underline{\underline{F}}_x$ is the first row of deformation gradient $\underline{\underline{F}}$. With analogous results for the derivatives of ε_{yy} and ε_{zz}:

$$\nabla_{\underline{u}_j} \varepsilon_{yy} = 2\underline{\underline{F}}_y \cdot d_j{}^y \tag{9.12}$$

$$\nabla_{\underline{u}_j} \varepsilon_{zz} = 2\underline{\underline{F}}_z \cdot d_j{}^z \tag{9.13}$$

Now, a mixed component shall be derived:

$$\nabla_{\underline{u}_j} \varepsilon_{xy} = \begin{bmatrix} \underbrace{\frac{\partial}{\partial u_j} \varepsilon_{xy}}_{\text{I}} & \underbrace{\frac{\partial}{\partial v_j} \varepsilon_{xy}}_{\text{II}} & \underbrace{\frac{\partial}{\partial w_j} \varepsilon_{xy}}_{\text{III}} \end{bmatrix} \tag{9.14}$$

$$
\begin{aligned}
\mathrm{I} &= \frac{\partial}{\partial u_j} u_{,y} + \underbrace{\frac{\partial}{\partial u_j} v_{,x}}_{0} + \frac{\partial}{\partial u_j}(u_{,x}\, u_{,y}) + \underbrace{\frac{\partial}{\partial u_j} v_{,x}\, v_{,y}}_{0} + \underbrace{\frac{\partial}{\partial u_j} w_{,x}\, w_{,y}}_{0} \\
&= \frac{\partial}{\partial u_j} u_{,y} + (\frac{\partial}{\partial u_j} u_{,x})\, u_{,y} + u_{,x}\, (\frac{\partial}{\partial u_j} u_{,y}) \\
&= d_j^y + d_j^x u_{,y} + u_{,x}\, d_j^y \\
&= d_j^y \cdot (1 + u_{,x}) + d_j^x u_{,y} \qquad (9.15) \\
\mathrm{II} &= \underbrace{\frac{\partial}{\partial v_j} u_{,y}}_{0} + \frac{\partial}{\partial v_j} v_{,x} + \underbrace{\frac{\partial}{\partial v_j}(u_{,x}\, u_{,y})}_{0} + \frac{\partial}{\partial v_j} v_{,x}\, v_{,y} + \underbrace{\frac{\partial}{\partial v_j} w_{,x}\, w_{,y}}_{0} \\
&= \frac{\partial}{\partial v_j} v_{,x} + \frac{\partial}{\partial v_j} v_{,x}\, v_{,y} \\
&= \frac{\partial}{\partial v_j} v_{,x} + (\frac{\partial}{\partial v_j} v_{,x})\, v_{,y} + v_{,x}\, (\frac{\partial}{\partial v_j} v_{,y}) \\
&= d_j^x + d_j^x v_{,y} + v_{,x}\, d_j^y \\
&= d_j^x \cdot (1 + v_{,y}) + d_j^y v_{,x} \qquad (9.16) \\
\mathrm{III} &= \underbrace{\frac{\partial}{\partial v_j} u_{,y}}_{0} + \underbrace{\frac{\partial}{\partial v_j} v_{,x}}_{0} + \underbrace{\frac{\partial}{\partial v_j}(u_{,x}\, u_{,y})}_{0} + \frac{\partial}{\partial v_j} v_{,x}\, v_{,y} + \frac{\partial}{\partial v_j} w_{,x}\, w_{,y} \\
&= \frac{\partial}{\partial v_j}(w_{,x})\, w_{,y} + w_{,x}\, \frac{\partial}{\partial v_j}(w_{,y}) \\
&= d_j^x w_{,y} + w_{,x}\, d_j^y \qquad (9.17)
\end{aligned}
$$

Thus:

$$
\begin{aligned}
\nabla_{\underline{u}_j} \varepsilon_{xy} &= \begin{bmatrix} d_j^y \cdot (1 + u_{,x}) + d_j^x u_{,y} & d_j^x \cdot (1 + v_{,y}) + d_j^y v_{,x} & d_j^x w_{,y} + w_{,x}\, d_j^y \end{bmatrix} \\
&= d_j^y \underline{\underline{F}}_x + d_j^x \underline{\underline{F}}_y \qquad (9.18)
\end{aligned}
$$

With analogous results for the derivatives of ε_{xz} and ε_{zy}:

$$
\nabla_{\underline{u}_j} \varepsilon_{xz} = \underline{\underline{F}}_z \cdot d_j{}^x + \underline{\underline{F}}_x \cdot d_j{}^z \qquad (9.19)
$$

$$
\nabla_{\underline{u}_j} \varepsilon_{zy} = \underline{\underline{F}}_z \cdot d_j{}^y + \underline{\underline{F}}_y \cdot d_j{}^z \qquad (9.20)
$$

$$
(9.21)
$$

The derivative of the full displacement matrix $\underline{\underline{\varepsilon}}$ can now be written as:

$$
\begin{aligned}
\nabla_{\underline{u}_j} \underline{\underline{\varepsilon}} =& 2\underline{\underline{F}}_x \cdot d_j{}^x + 2\underline{\underline{F}}_y \cdot d_j{}^y + 2\underline{\underline{F}}_z \cdot d_j{}^z + \\
& 2(d_j^y \underline{\underline{F}}_x + d_j^x \underline{\underline{F}}_y) + 2(\underline{\underline{F}}_z \cdot d_j{}^x + \underline{\underline{F}}_x \cdot d_j{}^z) + 2(\underline{\underline{F}}_z \cdot d_j{}^y + \underline{\underline{F}}_y \cdot d_j{}^z) \qquad (9.22)
\end{aligned}
$$

After some reordering:

$$
\begin{aligned}
\nabla_{\underline{u}_j} \underline{\underline{\varepsilon}} = 2(&\underline{\underline{F}}_x \cdot d_j{}^x + \underline{\underline{F}}_x \cdot d_j{}^y + \underline{\underline{F}}_x \cdot d_j{}^z \\
&\underline{\underline{F}}_y \cdot d_j{}^x + \underline{\underline{F}}_y \cdot d_j{}^y + \underline{\underline{F}}_y \cdot d_j{}^z \\
&\underline{\underline{F}}_z \cdot d_j{}^x + \underline{\underline{F}}_z \cdot d_j{}^y + \underline{\underline{F}}_z \cdot d_j{}^z) \qquad (9.23)
\end{aligned}
$$

Which can be shortened to:

$$\nabla_{\underline{u}_j \underline{\underline{\varepsilon}}} = 2 \cdot \underline{\underline{F}}^T \underline{d}_j \tag{9.24}$$

Finally, using the symmetry of $\underline{\underline{\sigma}}$:

$$\begin{aligned} \underline{f}_j &= -\underline{\sigma} \nabla_{\underline{u}_i \underline{\underline{\varepsilon}}} \\ &= -\underline{\underline{\sigma}} (2 \cdot \underline{\underline{F}}^T \underline{d}_j) \\ &= -2 \underline{\underline{F}} \, \underline{\underline{\sigma}} \, \underline{d}_j \end{aligned} \tag{9.25}$$

Note that this is the force per unit volume. To come up with the force that particle i induces on particle j one needs to multiply with the volume of i:

$$\underline{f}_j = -2 v_i \underline{\underline{F}} \, \underline{\underline{\sigma}} \, \underline{d}_j \tag{9.26}$$

It remains to determine $\underline{f}_i$. The equilibrium equation for $\underline{f}_i$ is:

$$\underline{f}_i = \underline{\underline{\sigma}} \nabla_{\underline{u}_i \underline{\underline{\varepsilon}}} \tag{9.27}$$

The analysis is thus restarted with:

$$\nabla u = \begin{bmatrix} u_{,x} \\ u_{,y} \\ u_{,z} \end{bmatrix} = \sum_j \nabla \overset{\circ}{W}_{ij} u_j \tag{9.28}$$

But this time the derivative is taken with respect to $\underline{u}_i$. Using RKPM the function value of the particle itself is not present in the formula for the gradient. Thus, all derivatives are zero:

$$\frac{\partial}{\partial u_j} \nabla u = \underline{0} \tag{9.29}$$

$$\frac{\partial}{\partial v_j} \nabla u = \underline{0} \tag{9.30}$$

$$\frac{\partial}{\partial w_j} \nabla u = \underline{0} \tag{9.31}$$

Trivially, $\underline{f}_i = 0$. This is in contrast to using a formulation which does in fact include u_i in the approximation procedure for the gradient, as for example in [178].

9.2 MONAGHANS ALGORITHM IN TENSION

Both the aritficial stresses by Monaghan [91] as well as the transport velocity formulation [282] are presented as means against the tensile instability. While, as discussed, the "tensile" instability is a misnomer and the same instability mode can occur in compression just as well, it seems imperative to subject the algorithms to a tensile test. This was done in [91] but not in [282]. If the algorithms are subjected to the tensile test as designed in 7.2.1, the results are bad to such a degree that an implementation error may be suspected, see figure

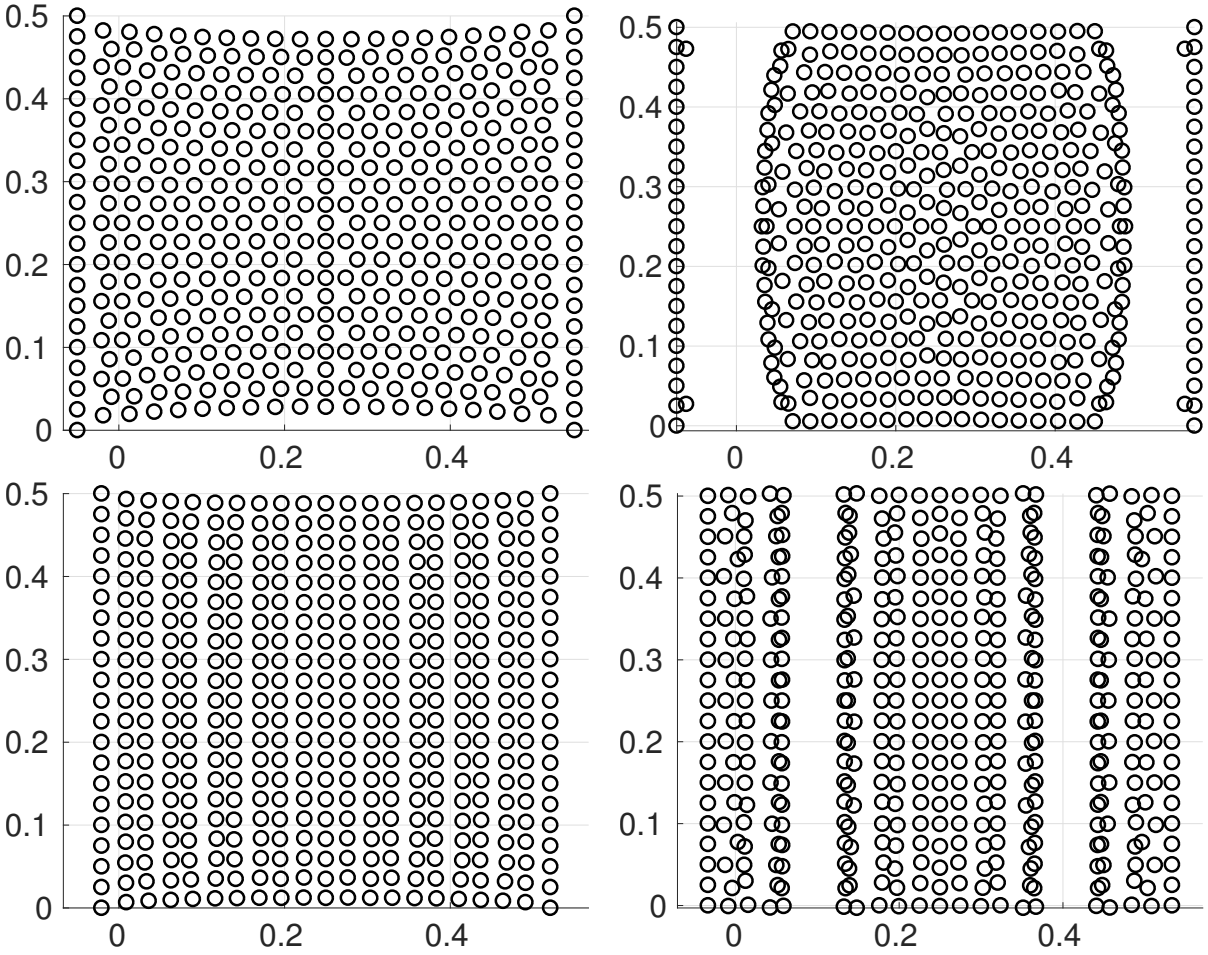

Figure 9.1: Monaghans algorithm and the TVF scheme in the tensile test designed in 7.2.1, Monaghan on top, TVF on the bottom. The build up of the instability mode is displayed on the left, subsequent failure due to numerical fracture is shown to the right.

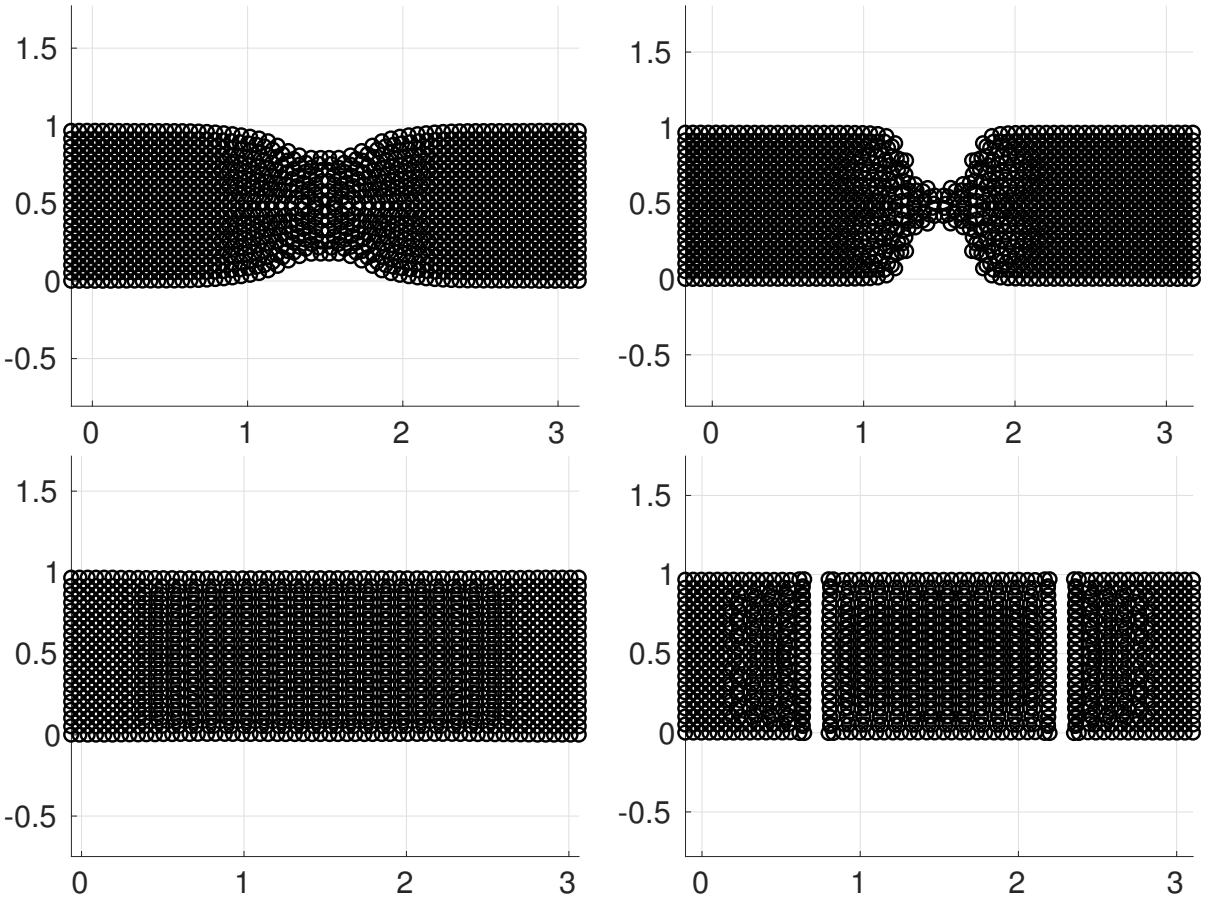

Figure 9.2: Monaghans algorithm and the TVF scheme in the tensile test designed in [91], Monaghan on top, TVF on the bottom. Monaghans algorithm indeed recreates the netching and notching described in the original publication while the TVF scheme fails to do so.

9.1. However, this is not the same model as presented in [91]. To exclude any doubts, the same model is recreated. It differs from the model as presented in 7.2.1 in the aspect ratio, i.e. 1:3 instead of 1:1 and the number of particles fixed on each side, which was increased to six columns. Addtionaly, an extremely simple implementation of a perfectly plastic material is applied:

$$f = min\left\{\sqrt{\frac{\sigma_y/3}{J^2}}, 1\right\} \tag{9.32}$$

$$\underline{\underline{\sigma}} = f \cdot \underline{\underline{\sigma}}^{\text{trial}} \tag{9.33}$$

where J_2 is the second invariant computed from the trial stress and σ_y is the yield point of the material at hand. The same necking and notching phenomena as in the reference can now be observed for the Monaghan algorithm but not for the TVF scheme, see frames 9.2. In the original publication [91] it is argued that the eventual numerical fracture happens after the fraction would occur physically, however, it is not clear why this should be the case for an any arbitrary material. Other algorithms, see 7.2.1 are able to sustain much larger strains in tension while the algorithms at hand seem to be able to resolve much larger deformation in other test cases like the rubber ring impact 7.2.2 than in the tension test at hand, be it for

the model presented in this thesis or for the one in [91]. This suggests that the instability at hand may be introduced by something other then the presence of tensile deformation, possibly by imposing the Dirichlet boundary conditions.

9.3 COMPUTING THE GREEN NAGHDI RATE

The Green-Naghdi stress rate [92] reads:

$$\mathring{\underline{\underline{S}}} = \dot{\underline{\underline{S}}} + \underline{\underline{S}} \cdot \underline{\underline{\Omega}} - \underline{\underline{\Omega}} \cdot \underline{\underline{S}} \tag{9.34}$$

where $\underline{\underline{\Omega}} = \dot{\underline{\underline{R}}} \cdot \underline{\underline{R}}^T$. It is not straight forward to compute $\underline{\underline{\Omega}}$ since

$$\underline{\underline{F}} = \underline{\underline{R}} \cdot \underline{\underline{U}} \tag{9.35}$$
$$\dot{\underline{\underline{F}}} = \dot{\underline{\underline{R}}} \cdot \underline{\underline{U}} + \underline{\underline{R}} \cdot \dot{\underline{\underline{U}}} \tag{9.36}$$

$\dot{\underline{\underline{R}}}$ is thus not readily obtained. A procedure towards this end is given in [60]. The paper uses a left handed polar decomposition instead of the more common right handed one:

$$\underline{\underline{F}} = \underline{\underline{R}} \cdot \underline{\underline{U}} = \underline{\underline{V}} \cdot \underline{\underline{R}} \tag{9.37}$$

The derivation starts with the expression for $\underline{\underline{L}}$ derived above and inserted into the left-handed polar decomposition for $\underline{\underline{F}}$.

$$\begin{aligned}
\underline{\underline{L}} &= \dot{\underline{\underline{F}}} \cdot \underline{\underline{F}}^{-1} \\
&= \partial / (\partial t) \left(\underline{\underline{V}}\ \underline{\underline{R}} \right) \cdot \left(\underline{\underline{V}}\ \underline{\underline{R}} \right)^{-1} \\
&= \left(\dot{\underline{\underline{V}}}\ \underline{\underline{R}} + \underline{\underline{V}}\ \dot{\underline{\underline{R}}} \right) \cdot \left(\underline{\underline{R}}^{-1} \underline{\underline{V}}^{-1} \right) \\
&= \left(\dot{\underline{\underline{V}}}\ \underline{\underline{R}}\ \underline{\underline{R}}^{-1} \underline{\underline{V}}^{-1} + \underline{\underline{V}}\ \dot{\underline{\underline{R}}}\ \underline{\underline{R}}^{-1} \underline{\underline{V}}^{-1} \right) \\
&= \dot{\underline{\underline{V}}}\ \underline{\underline{V}}^{-1} + \underline{\underline{V}}\ \underline{\underline{\Omega}}\ \underline{\underline{V}}^{-1}
\end{aligned} \tag{9.38}$$

Where in the last step the identity $\underline{\underline{R}}^{-1} = \underline{\underline{R}}^T$ was used. Some interesting observations can be made from (9.38). $\underline{\underline{\Omega}}$ can be expressed as linear combination of , $\underline{\underline{V}}$ and $\dot{\underline{\underline{V}}}$, i.e. without using $\dot{R}$. In this case (9.38) is a system of simultaneous equations for $\underline{\underline{\Omega}}$ since $\Omega_{ik} = \epsilon_{ijk}\omega_j$. That is, $\underline{\underline{\Omega}}$ has only three independent components (in 3D). However (9.38) is not sufficient for obtaining $\underline{\underline{\Omega}}$ since the time derivative of the stretch tensor $\dot{\underline{\underline{V}}}$ is still present. To eliminate $\dot{\underline{\underline{V}}}$, (9.38) is first post-multiplied with $\underline{\underline{V}}$ and subsequently the transpose of each side of the equation is subtracted. Note that the description in the original paper [60] is wrong (it is

suggested that the relation was post- as well as pre-multiplied with $\underline{\underline{V}}$ and the transpose was added instead of subtracted).

$$\underline{\underline{L}} = \underline{\underline{\dot{V}}}\,\underline{\underline{V}}^{-1} + \underline{\underline{V}}\,\underline{\underline{\Omega}}\,\underline{\underline{V}}^{-1} \tag{9.39}$$

$$\underline{\underline{L}}\,\underline{\underline{V}} = \underline{\underline{\dot{V}}}\,\underline{\underline{V}}^{-1}\underline{\underline{V}} + \underline{\underline{V}}\,\underline{\underline{\Omega}}\,\underline{\underline{V}}^{-1}\underline{\underline{V}} \tag{9.40}$$

$$\underline{\underline{L}}\,\underline{\underline{V}} = \underline{\underline{\dot{V}}} + \underline{\underline{V}}\,\underline{\underline{\Omega}} \tag{9.41}$$

$$\underline{\underline{L}}\,\underline{\underline{V}} - (\underline{\underline{L}}\,\underline{\underline{V}})^T = \underline{\underline{\dot{V}}} - \underline{\underline{\dot{V}}}^T + \underline{\underline{V}}\,\underline{\underline{\Omega}} - (\underline{\underline{V}}\,\underline{\underline{\Omega}})^T \tag{9.42}$$

$$\underline{\underline{L}}\,\underline{\underline{V}} - \underline{\underline{V}}^T\underline{\underline{L}}^T = \underline{\underline{V}}\,\underline{\underline{\Omega}} - \underline{\underline{\Omega}}^T\underline{\underline{V}}^T \tag{9.43}$$

$$(\underline{\underline{D}} + \underline{\underline{W}})\underline{\underline{V}} - \underline{\underline{V}}^T(\underline{\underline{D}} + \underline{\underline{W}})^T = \underline{\underline{V}}\,\underline{\underline{\Omega}} + \underline{\underline{\Omega}}\,\underline{\underline{V}} \tag{9.44}$$

$$(\underline{\underline{D}} + \underline{\underline{W}})\underline{\underline{V}} - \underline{\underline{V}}(\underline{\underline{D}} - \underline{\underline{W}}) = \underline{\underline{V}}\,\underline{\underline{\Omega}} + \underline{\underline{\Omega}}\,\underline{\underline{V}} \tag{9.45}$$

$$(\underline{\underline{D}}\,\underline{\underline{V}} - \underline{\underline{V}}\,\underline{\underline{D}}) + (\underline{\underline{W}}\,\underline{\underline{V}} + \underline{\underline{V}}\,\underline{\underline{W}}) = \underline{\underline{V}}\,\underline{\underline{\Omega}} + \underline{\underline{\Omega}}\,\underline{\underline{V}} \tag{9.46}$$

Where the symmetries $\underline{\underline{V}} = \underline{\underline{V}}^T$ (thus $\dot{V} = \dot{V}^T$), $D = D^T$ as well as the anti symmetries $W = -W^T$ and $\underline{\underline{\Omega}} = -\underline{\underline{\Omega}}^T$ have been used. (9.46) now describes a linear relation between W, D, (which can be obtained from $\underline{\underline{L}}$) and $\underline{\underline{V}}$ (which can be obtained by the polar decomposition of $\underline{\underline{F}}$).

It is not directly possible to use ((9.46)) as simultaneous equations for the components of Ω since the system is underdetermined (3 equations for 9 components in 3D and 2 equations for 4 components in 2D). However, the antisymmetry of Ω and other tensors can be exploited after slight rearrangement:

$$(\underline{\underline{D}}\,\underline{\underline{V}} - \underline{\underline{V}}\,\underline{\underline{D}}) + (\underline{\underline{W}}\,\underline{\underline{V}} + \underline{\underline{V}}\,\underline{\underline{W}}) = \underline{\underline{V}}\,\underline{\underline{\Omega}} + \underline{\underline{\Omega}}\,\underline{\underline{V}} \tag{9.47}$$

$$\underline{\underline{D}}\,\underline{\underline{V}} - \underline{\underline{V}}\,\underline{\underline{D}} = \underline{\underline{V}}\,\underline{\underline{\Omega}} - \underline{\underline{V}}\,\underline{\underline{W}} + \underline{\underline{\Omega}}\,\underline{\underline{V}} - \underline{\underline{W}}\,\underline{\underline{V}} \tag{9.48}$$

$$\underbrace{\underline{\underline{D}}\,\underline{\underline{V}} - \underline{\underline{V}}\,\underline{\underline{D}}}_{Z} = \underline{\underline{V}}\,\underbrace{(\underline{\underline{\Omega}} - \underline{\underline{W}})}_{H} + \underbrace{(\underline{\underline{\Omega}} - \underline{\underline{W}})}_{H}\underline{\underline{V}} \tag{9.49}$$

$$\tag{9.50}$$

Thus:

$$\underline{\underline{Z}} = \underline{\underline{V}}\,\underline{\underline{H}} + \underline{\underline{H}}\,\underline{\underline{V}} \tag{9.51}$$

Since H is the difference between the two antisymmetric tensors $\underline{\underline{\Omega}}$ and W it is again antisymmetric and thus has only three (or two in 2D, respectively) independent components. Writing Z in expanded form:

$$\begin{bmatrix} 0 & -V_{xz}H_{yz} + V_{yz}H_{xz} + (V_{xx} + V_{yy})H_{xy} & V_{xy}H_{yz} + (V_{xx} + V_{zz})H_{xz} + V_{yz}H_{xy} \\ V_{xz}H_{yz} - V_{yz}H_{xz} - (V_{xx} + V_{yy})H_{xy} & 0 & (V_{yy} + V_{zz})H_{yz} + V_{xy}H_{xz} - V_{xz}H_{xy} \\ -V_{xy}H_{yz} - (V_{xx} + V_{zz})H_{xz} - V_{yz}H_{xy} & -(V_{yy} + V_{zz})H_{yz} - V_{xy}H_{xz} + V_{xz}H_{xy} & 0 \end{bmatrix} \tag{9.52}$$

Now, since:

$$Z_{ik} = \epsilon_{ijl}h_l = \begin{bmatrix} 0 & -z_3 & z_2 \\ z_3 & 0 & -z_1 \\ -z_2 & z_1 & 0 \end{bmatrix} \tag{9.53}$$

the construction of the vector

$$\underline{z} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} Z_{zy} \\ Z_{xz} \\ Z_{yx} \end{bmatrix} = \begin{bmatrix} -(V_{yy} + V_{zz})H_{yz} - V_{xy}H_{xz} + V_{xz}H_{xy} \\ V_{xy}H_{yz} + (V_{xx} + V_{zz})H_{xz} + V_{yz}H_{xy} \\ V_{xz}H_{yz} - V_{yz}H_{xz} - (V_{xx} + V_{yy})H_{xy} \end{bmatrix} \tag{9.54}$$

is motivated. Factoring out the $H_{\tilde{\xi}\tilde{\xi}}$ terms:

$$\underline{z} = \begin{bmatrix} (V_{yy} + V_{zz}) - V_{xy} - V_{xz} \\ -V_{xy} + (V_{xx} + V_{zz}) - V_{yz} \\ -V_{xz} - V_{yz} + (V_{xx} + V_{yy}) \end{bmatrix} \cdot \begin{bmatrix} -H_{yz} \\ H_{xz} \\ -H_{xy} \end{bmatrix} \tag{9.55}$$

Reordering the terms:

$$\underline{z} = \begin{bmatrix} (V_{yy} + V_{zz}) - V_{xy} - V_{xz} \\ (V_{xx} + V_{zz}) - V_{xy} - V_{yz} \\ (V_{xx} + V_{yy}) - V_{xz} - V_{yz} \end{bmatrix} \cdot \begin{bmatrix} -H_{yz} \\ H_{xz} \\ -H_{xy} \end{bmatrix} \tag{9.56}$$

And finally expanding the bracketed expressions to the trace yields the desired result:

$$\underline{z} = \begin{bmatrix} (V_{xx} + V_{yy} + V_{zz}) - V_{xx} - V_{xy} - V_{xz} \\ (V_{xx} + V_{yy} + V_{zz}) - V_{xy} - V_{yy} - V_{yz} \\ (V_{xx} + V_{yy} + V_{zz}) - V_{xz} - V_{yz} - V_{zz} \end{bmatrix} \cdot \begin{bmatrix} -H_{yz} \\ H_{xz} \\ -H_{xy} \end{bmatrix} \tag{9.57}$$

$$= (\underline{\underline{I}} \cdot tr(\underline{\underline{V}}) - \underline{\underline{V}}) \cdot \underline{\tilde{h}} \tag{9.58}$$

A numerical recipe, given the derivatives of the velocity field and the displacement field, can now be formulated as follows:

1. Compute $L_{ij} = \frac{\partial v_i}{\partial x_j}$ and $F_{ij} = \frac{\partial u_i}{\partial X_j} + \delta_{ij}$
2. Compute the polar decomposition of $\underline{\underline{F}} = \underline{\underline{V}}\,\underline{\underline{R}}$ and compute the rate of deformation and spin tensors $\underline{\underline{D}} = 1/2\,(\underline{\underline{L}} + \underline{\underline{L}}^T)$, $\underline{\underline{W}} = 1/2\,(\underline{\underline{L}} - \underline{\underline{L}}^T)$
3. Form $\underline{\underline{Z}} = \underline{\underline{D}}\,\underline{\underline{V}} - \underline{\underline{V}}\,\underline{\underline{D}}$
4. Solve system (9.58) for the components of $\underline{\tilde{h}}$
5. Build $\underline{\underline{H}}$ from $\underline{\tilde{h}}$ and obtain $\underline{\underline{\Omega}} = \underline{\underline{H}} + \underline{\underline{W}}$

The process for 2D is a lot easier. The following expression is considered again:

$$\underline{\underline{Z}} = \underline{\underline{V}}\,\underline{\underline{H}} + \underline{\underline{H}}\,\underline{\underline{V}} \tag{9.59}$$

However, this time the right hand side is simpler since H (and thus Z) have only one independent component:

$$\begin{bmatrix} 0 & Z_{xy} \\ -Z_{xy} & 0 \end{bmatrix} = \begin{bmatrix} 0 & H_{xy}V_{xx} + H_{xy}V_{yy} \\ -H_{xy}V_{xx} - H_{xy}V_{yy} & 0 \end{bmatrix} \tag{9.60}$$

The single equation for H_{xy} is immediately obtained as:

$$H_{xy} = \frac{Z_{xy}}{V_{xx} + V_{yy}} \tag{9.61}$$

and the procedure above can be started from step 5. directly

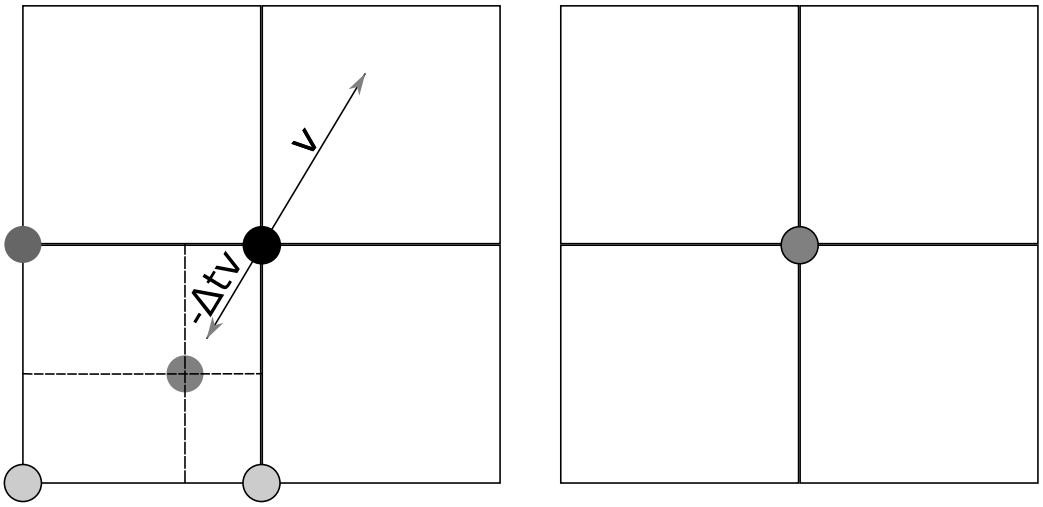

Figure 9.3: Semi lagrangian advection is performed by traveling backwards along the current velocity, then interpolating the local value at the location, for example linearly, then updating the origin point with that value.

9.4 A REMARK TO NUMERICAL DISSIPATION

Generally, numerical dissipation is the phenomenon exhibited by a wide array of numerical methods that spuriously lose energy, or some other quantity, over the course of the simulation. For example by damping of sharp features, loss of velocity in a project past impact, or mass loss in a transport simulation. In this section, this concept is illuminated by reproduction of a short remark in [250].

To understand the remark it is first important to understand semi-Lagrangian advection on a Eulerian grid. The procedure works as follows. For all grid points $\underline{p}_i$ with velocity $\underline{v}_i$ and transported quantity ϱ_i

- Travel backwards along velocity $\underline{v}_i$ in an Eulerian step $\underline{p}_i^* = \underline{p} - \Delta t \cdot \underline{v}_i$.
- At $\underline{p}_i^*$, interpolate quantity ϱ from surrounding points to get ϱ^*
- Set $\varrho_i \leftarrow \varrho^*$

This is illustrated in figure 9.3 in 2D, but the following derivation is contained to one dimension.

Consider the Eulerian transport equation of quantity ϱ:

$$\frac{\partial \varrho}{\partial t} + v\frac{\partial \varrho}{\partial x} = 0 \tag{9.62}$$

Integrate in time using the Semi Lagrangian scheme using linear interpolation:

$$\varrho_i^{n+1} = \alpha \cdot \varrho_{i-1}^n + (1-\alpha) \cdot \varrho_i^n \tag{9.63}$$

with $\alpha = v\frac{\Delta t}{\Delta x}$ being the fraction of the cell the particle traveled (keeping CFL condition):

$$\varrho_i^{n+1} = v\frac{\Delta t}{\Delta x} \cdot \varrho_{i-1}^n + \left(1 - v\frac{\Delta t}{\Delta x}\right) \cdot \varrho_i^n \tag{9.64}$$

This is rearranged:

$$\varrho_i^{n+1} = \varrho_i^n - v\Delta t\frac{\varrho_i^n - \varrho_{i-1}^n}{\Delta x} \tag{9.65}$$

A Taylor series expansion of ϱ_{i-1}^n about ϱ_i^n is inserted:

$$\varrho_{i-i}^n = \varrho_i^n - \left(\frac{\partial \rho^n}{\partial x}\right)|_{x=x_i}\Delta x + \frac{1}{2}\left(\frac{\partial^2 \rho^n}{\partial^2 x}\right)|_{x=x_i}\Delta x^2 + \mathcal{O}(\Delta x^3) \tag{9.66}$$

Reinserting into (9.65) yields:

$$\varrho_i^{n+1} = \varrho_i^n - v\Delta t\left(\frac{\partial \rho^n}{\partial x}\right)|_{x=x_i} - \frac{1}{2}v\Delta t\left(\frac{\partial^2 \rho^n}{\partial^2 x}\right)|_{x=x_i}\Delta x - \mathcal{O}(\Delta x^2) \tag{9.67}$$

This is then written in continuous form:

$$\frac{\partial \varrho}{\partial t} + v\frac{\partial \varrho}{\partial x} = v\frac{\partial^2 \varrho}{\partial x^2} \tag{9.68}$$

Which reveals that the PDE 9.62 now features an additional diffuse / damping term.

BIBLIOGRAPHY

[1] David Adalsteinsson and James A Sethian.
"A fast level set method for propagating interfaces."
In: *Journal of computational physics* 118.2 (1995), pp. 269–277.

[2] S Adami, XY Hu, and Nikolaus A Adams.
"A transport-velocity formulation for smoothed particle hydrodynamics."
In: *Journal of Computational Physics* 241 (2013), pp. 292–307.

[3] Mansur Akbari et al. "A new value for Johnson Cook damage limit criterion in machining with large negative rake angle as basis for understanding of grinding."
In: *Journal of Materials Processing Technology* 234 (2016), pp. 58–71.

[4] Berni J Alder and T E Wainwright.
"Studies in molecular dynamics. I. General method."
In: *The Journal of Chemical Physics* 31.2 (1959), pp. 459–466.

[5] Takashi Amada et al. "Particle-based fluid simulation on GPU."
In: *ACM Workshop on General-Purpose Computing on Graphics Processors and SIGGRAPH*. ACM SIGGRAPH. 2004.

[6] Ravindra Ambati et al.
"Application of material point methods for cutting process simulations."
In: *Computational Materials Science* 57 (2012), pp. 102–110.

[7] Gene M Amdahl. "Validity of the single processor approach to achieving large scale computing capabilities."
In: *Proceedings of the April 18-20, 1967, spring joint computer conference*. ACM. 1967, pp. 483–485.

[8] PJ Arrazola and T Ozel. "Numerical modelling of 3D hard turning using arbitrary Lagrangian Eulerian finite element method."
In: *International Journal of Machining and Machinability of Materials* 4.1 (2008), pp. 14–25.

[9] PJ Arrazola et al. "Recent advances in modelling of metal machining processes."
In: *CIRP Annals-Manufacturing Technology* 62.2 (2013), pp. 695–718.

[10] Tony Atkins. *The science and engineering of cutting: the mechanics and processes of separating, scratching and puncturing biomaterials, metals and non-metals*.
Butterworth-Heinemann, 2009.

[11] JC Aurich et al. "Development of a superabrasive grinding wheel with defined grain structure using kinematic simulation."
In: *CIRP Annals-Manufacturing Technology* 52.1 (2003), pp. 275–280.

[12] Eyup Bagci.
"3-D numerical analysis of orthogonal cutting process via mesh-free method."
In: *International Journal of Physical Sciences* 6.6 (2011), pp. 1267–1282.

[13] MA Balbaa and Mohamed NA Nasr. "Prediction of residual stresses after laser-assisted machining of Inconel 718 using SPH."
In: *Procedia CIRP* 31 (2015), pp. 19–23.

[14] Herbert Balke. *Einführung in die technische Mechanik: Festigkeitslehre*.
Springer-Verlag, 2014.

[15] Biswajit Banerjee. "An evaluation of plastic flow stress models for the simulation of high-temperature and high-strain-rate deformation of metals."
In: *arXiv preprint cond-mat/0512466* (2005).

[16] C Bradford Barber, David P Dobkin, and Hannu Huhdanpaa.
"The quickhull algorithm for convex hulls."
In: *ACM Transactions on Mathematical Software (TOMS)* 22.4 (1996), pp. 469–483.

[17] SG Bardenhagen and EM Kober.
"The generalized interpolation material point method."
In: *Computer Modeling in Engineering and Sciences* 5.6 (2004), pp. 477–496.

[18] J Bauschinger. "On the change of the elastic limit and the strength of iron and steel, by drawing out, by heating and cooling, and by repetition of loading (summary)."
In: *Minutes of Proceedings of the Institution of Civil Engineers with Other Selected and Abstracted Papers* 87 (1886), p. 463.

[19] Markus Becker, Markus Ihmsen, and Matthias Teschner.
"Corotated SPH for Deformable Solids." In: *NPH*. Citeseer. 2009, pp. 27–34.

[20] Nathan Bell and Jared Hoberock.
"Thrust: A productivity-oriented library for CUDA."
In: *GPU computing gems Jade edition*. Elsevier, 2011, pp. 359–371.

[21] T Belytschko and BJ Hsieh.
"Non-linear transient finite element analysis with convected co-ordinates."
In: *International Journal for Numerical Methods in Engineering* 7.3 (1973), pp. 255–271.

[22] Ted Belytschko, Yun Yun Lu, and Lei Gu. "Element-free Galerkin methods."
In: *International journal for numerical methods in engineering* 37.2 (1994), pp. 229–256.

[23] Ted Belytschko et al. "A unified stability analysis of meshless particle methods."
In: *International Journal for Numerical Methods in Engineering* 48.9 (2000), pp. 1359–1400.

[24] Ted Belytschko et al. "Meshless methods: an overview and recent developments."
In: *Computer methods in applied mechanics and engineering* 139.1 (1996), pp. 3–47.

[25] Ted Belytschko et al. *Nonlinear finite elements for continua and structures*.
John wiley & sons, 2013.

[26] Gino van den Bergen.
"Efficient collision detection of complex deformable models using AABB trees."
In: *Journal of Graphics Tools* 2.4 (1997), pp. 1–13.

[27] Halil Bil, S Engin Kılıç, and A Erman Tekkaya. "A comparison of orthogonal cutting data from experiments with three different finite element models."
In: *International Journal of Machine Tools and Manufacture* 44.9 (2004), pp. 933–944.

[28] Jeremiah U Brackbill, Douglas B Kothe, and Hans M Ruppel.
"FLIP: a low-dissipation, particle-in-cell method for fluid flow."
In: *Computer Physics Communications* 48.1 (1988), pp. 25–38.

[29] Richard P Brent. *Algorithms for minimization without derivatives*.
Courier Corporation, 2013.

[30] L Brookshaw.
"A method of calculating radiative heat diffusion in particle simulations."
In: *Proceedings of the Astronomical Society of Australia*. Vol. 6. 1985, pp. 207–210.

[31] John Charles Butcher. "The numerical analysis of ordinary differential equations: Runge-Kutta and general linear methods." In: (1987).

[32] Madalina Calamaz, Dominique Coupard, and Franck Girot.
"A new material model for 2D numerical simulation of serrated chip formation when machining titanium alloy Ti–6Al–4V."
In: *International Journal of Machine Tools and Manufacture* 48.3 (2008), pp. 275–288.

[33] Madalina Calamaz et al.
"Toward a better understanding of tool wear effect through a comparison between experiments and SPH numerical modelling of machining hard materials."
In: *International journal of refractory metals and hard materials* 27.3 (2009), pp. 595–604.

[34] Oana Cazacu, Brian Plunkett, and Frédéric Barlat.
"Orthotropic yield criterion for hexagonal closed packed metals."
In: *International Journal of Plasticity* 22.7 (2006), pp. 1171–1194.

[35] Seung-Hoon Cha, Shu-ichiro Inutsuka, and Sergei Nayakshin.
"Kelvin–Helmholtz instabilities with Godunov smoothed particle hydrodynamics."
In: *Monthly Notices of the Royal Astronomical Society* 403.3 (2010), pp. 1165–1174.

[36] Jean-Louis Chaboche.
"Constitutive equations for cyclic plasticity and cyclic viscoplasticity."
In: *International journal of plasticity* 5.3 (1989), pp. 247–302.

[37] JL Chaboche. "A review of some plasticity and viscoplasticity constitutive theories."
In: *International Journal of Plasticity* 24.10 (2008), pp. 1642–1693.

[38] AK Chaniotis, D Poulikakos, and P Koumoutsakos. "Remeshed smoothed particle hydrodynamics for the simulation of viscous and heat conducting flows."
In: *Journal of Computational Physics* 182.1 (2002), pp. 67–90.

[39] Jiun-Shyan Chen, Michael Hillman, and Sheng-Wei Chi.
"Meshfree methods: progress made after 20 years."
In: *Journal of Engineering Mechanics* 143.4 (2017), p. 04017001.

[40] Jiun-Shyan Chen et al.
"A Lagrangian reproducing kernel particle method for metal forming analysis."
In: *Computational Mechanics* 22.3 (1998), pp. 289–307.

[41] JK Chen, JE Beraun, and TC Carney. "A corrective smoothed particle method for boundary value problems in heat conduction."
In: *International Journal for Numerical Methods in Engineering* 46.2 (1999), pp. 231–252.

[42] JK Chen, JE Beraun, and CJ Jih.
"An improvement for tensile instability in smoothed particle hydrodynamics."
In: *Computational Mechanics* 23.4 (1999), pp. 279–287.

[43] Rongjun Cheng and Kim Meow Liew. "The reproducing kernel particle method for two-dimensional unsteady heat conduction problems."
In: *Computational Mechanics* 45.1 (2009), pp. 1–10.

[44] THC Childs. "Surface energy, cutting edge radius and material flow stress size effects in continuous chip formation of metals."
In: *CIRP Journal of Manufacturing Science and Technology* 3.1 (2010), pp. 27–39.

[45] THC Childs and K Maekawa.
"Computer-aided simulation and experimental studies of chip flow and tool wear in the turning of low alloy steels by cemented carbide tools."
In: *Wear* 139.2 (1990), pp. 235–250.

[46] JP Choquin and S Huberson. "Particles simulation of viscous flow."
In: *Computers & fluids* 17.2 (1989), pp. 397–410.

[47] Paul W Cleary and Joseph J Monaghan.
"Conduction modelling using smoothed particle hydrodynamics."
In: *Journal of Computational Physics* 148.1 (1999), pp. 227–264.

[48] Olivier Comas et al. "Efficient nonlinear FEM for soft tissue modelling and its GPU implementation within the open source framework SOFA."
In: *International Symposium on Biomedical Simulation*. Springer. 2008, pp. 28–39.

[49] A Connolly and L Iannucci.
"Godunov SPH with an operator-splitting procedure for materials with strength."
In: *Blucher Mechanical Engineering Proceedings* 1.1 (2012), pp. 3554–3568.

[50] A Connolly et al.
"SECOND ORDER GODUNOV SPH FOR HIGH VELOCITY IMPACT DYNAMICS."
In: ().

[51] Georges-Henri Cottet and Petros D Koumoutsakos. *Vortex methods: theory and practice*. Cambridge university press, 2000.

[52] GH Cottet and S Mas-Gallic. "A particle method to solve the Navier-Stokes system."
In: *Numerische Mathematik* 57.1 (1990), pp. 805–827.

[53] AJC Crespo et al. "DualSPHysics, new GPU computing on SPH models."
In: *Proc. 6th International SPHERIC Workshop*. 2011, pp. 348–354.

[54] Alejandro JC Crespo et al. "DualSPHysics: Open-source parallel CFD solver based on Smoothed Particle Hydrodynamics (SPH)."
In: *Computer Physics Communications* 187 (2015), pp. 204–216.

[55] Timothy A Davis.
"Algorithm 832: UMFPACK V4. 3—an unsymmetric-pattern multifrontal method."
In: *ACM Transactions on Mathematical Software (TOMS)* 30.2 (2004), pp. 196–199.

[56] Arnaldo Carvalho De Melo. "The new linux'perf'tools." In: *Slides from Linux Kongress*. Vol. 18. 2010.

[57] Boris Delaunay. “Sur la sphere vide.” In: *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk* 7.793-800 (1934), pp. 1–2.

[58] ZS Deligonul and S Bilgen. “Solution of the Volterra equation of renewal theory with the Galerkin technique using cubic splines.” In: *Journal of Statistical Computation and Simulation* 20.1 (1984), pp. 37–45.

[59] Berend Denkena and Hans Kurt Tönshoff. *Spanen: grundlagen*. Springer-Verlag, 2011.

[60] John K Dienes. “On the analysis of rotation and stress rate in deforming bodies.” In: *Acta mechanica* 32.4 (1979), pp. 217–232.

[61] Gary A Dilts. “Moving-least-squares-particle hydrodynamics—I. Consistency and stability.” In: *International Journal for Numerical Methods in Engineering* 44.8 (1999), pp. 1115–1155.

[62] Eckart Doege, Heinz Meyer-Nolkemper, and Imtiaz Saeed. *Fließkurvenatlas metallischer Werkstoffe: mit Fließkurven für 73 Werkstoffe und einer grundlegenden Einführung*. Hanser, 1986.

[63] Klaus Dolag, Matthias Bartelmann, and Harald Lesch. “SPH simulations of magnetic fields in galaxy clusters.” In: *arXiv preprint astro-ph/9906329* (1999).

[64] F Ducobu, E Rivière-Lorphèvre, and E Filippi. “Experimental and numerical investigation of the uncut chip thickness reduction in Ti6Al4V orthogonal cutting.” In: *Meccanica* 52.7 (2017), pp. 1577–1592.

[65] D Dudzinski and A Molinari. “A modelling of cutting for viscoplastic materials.” In: *International Journal of Mechanical Sciences* 39.4 (1997), pp. 369–389.

[66] CT Dyka and RP Ingel. *Addressing Tension Instability in SPH Methods.* Tech. rep. NAVAL RESEARCH LAB WASHINGTON DC, 1994.

[67] CT Dyka and RP Ingel. “An approach for tension instability in smoothed particle hydrodynamics (SPH).” In: *Computers & structures* 57.4 (1995), pp. 573–580.

[68] CT Dyka, PW Randles, and RP Ingel. “Stress points for tension instability in SPH.” In: *International Journal for Numerical Methods in Engineering* 40.13 (1997), pp. 2325–2341.

[69] Peter Eberhard and Timo Gaugele. “Simulation of cutting processes using mesh-free Lagrangian particle methods.” In: *Computational Mechanics* 51.3 (2013), pp. 261–278.

[70] A Eghtesad, AR Shafiei, and M Mahzoon. “Study of dynamic behavior of ceramic–metal FGM under high velocity impact conditions using CSPM method.” In: *Applied Mathematical Modelling* 36.6 (2012), pp. 2724–2738.

[71] Jeff D Eldredge, Anthony Leonard, and Tim Colonius. “A general deterministic treatment of derivatives in particle methods.” In: *Journal of Computational Physics* 180.2 (2002), pp. 686–709.

[72] Douglas Enright, Frank Losasso, and Ronald Fedkiw. “A fast and accurate semi-Lagrangian particle level set method.” In: *Computers & structures* 83.6 (2005), pp. 479–490.

[73] Douglas Enright et al.
"A hybrid particle level set method for improved interface capturing."
In: *Journal of Computational physics* 183.1 (2002), pp. 83–116.

[74] R Fatehi, M Fayazbakhsh, and M Manzari.
"On discretization of second-order derivatives in smoothed particle hydrodynamics."
In: *Proceedings of World Academy of Science, Engineering and Technology*. Vol. 30.
Citeseer. 2008, pp. 243–246.

[75] R Fatehi and MT Manzari. "Error estimation in smoothed particle hydrodynamics and a new scheme for second derivatives."
In: *Computers & Mathematics with Applications* 61.2 (2011), pp. 482–498.

[76] Carlos Felippa. *Lecture notes in Matrix Finite Element Methods in Dynamics*. 2013.

[77] Dalia Fishelov. "A new vortex scheme for viscous flows."
In: *Journal of computational physics* 86.1 (1990), pp. 211–224.

[78] DP Flanagan and T Belytschko.
"A uniform strain hexahedron and quadrilateral with orthogonal hourglass control."
In: *International journal for numerical methods in engineering* 17.5 (1981), pp. 679–706.

[79] F Fleissner. "Pasimodo v1. 9.3, software package and template files."
In: *Inpartik & ITM University of Stuttgart, Tübingen* (2012).

[80] Kirk Fraser. "Robust and efficient meshfree solid thermo-mechanics simulation of friction stir welding." PhD thesis. Université du Québec à Chicoutimi, 2017.

[81] Isaac Fried and Arthur R Johnson. "A note on elastic energy density functions for largely deformed compressible rubber solids."
In: *Computer Methods in Applied Mechanics and Engineering* 69.1 (1988), pp. 53–64.

[82] David A Fulk and Dennis W Quinn.
"An analysis of 1-D smoothed particle hydrodynamics kernels."
In: *Journal of Computational Physics* 126.1 (1996), pp. 165–180.

[83] Nico Galoppo et al. "LU-GPU: Efficient algorithms for solving dense linear systems on graphics hardware."
In: *Proceedings of the 2005 ACM/IEEE conference on Supercomputing*.
IEEE Computer Society. 2005, p. 3.

[84] Sahil Garg and Mohit Pant.
"Meshfree Methods: A Comprehensive Review of Applications."
In: *International Journal of Computational Methods* (2017), p. 1830001.

[85] J Davison de St Germain et al.
"Uintah: A massively parallel problem solving environment." In: *High-Performance Distributed Computing, 2000. Proceedings. The Ninth International Symposium on*.
IEEE. 2000, pp. 33–41.

[86] JB Gibson et al. "Dynamics of radiation damage."
In: *Physical Review* 120.4 (1960), p. 1229.

[87] Robert A Gingold and Joseph J Monaghan.
"Smoothed particle hydrodynamics: theory and application to non-spherical stars."
In: *Monthly notices of the royal astronomical society* 181.3 (1977), pp. 375–389.

[88] Dominik Göddeke, Robert Strzodka, and Stefan Turek.
"Performance and accuracy of hardware-oriented native-, emulated-and mixed-precision solvers in FEM simulations." In: *International Journal of Parallel, Emergent and Distributed Systems* 22.4 (2007), pp. 221–256.

[89] Gene H Golub and Charles F Van Loan. *Matrix computations*. Vol. 3. JHU Press, 2012.

[90] Prashant Goswami et al. "Interactive SPH simulation and rendering on the GPU."
In: *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association. 2010, pp. 55–64.

[91] JP Gray, JJ Monaghan, and RP Swift. "SPH elastic dynamics."
In: *Computer methods in applied mechanics and engineering* 190.49 (2001), pp. 6641–6662.

[92] Albert Edward Green and Paul Mansour Naghdi.
"A general theory of an elastic-plastic continuum."
In: *Archive for rational mechanics and analysis* 18.4 (1965), pp. 251–281.

[93] Simon Green. "Particle simulation using cuda."
In: *NVIDIA whitepaper* 6 (2010), pp. 121–128.

[94] Dietmar Gross and Thomas Seelig.
Fracture mechanics: with an introduction to micromechanics. Springer, 2017.

[95] Eduard Grüneisen. "Theorie des festen Zustandes einatomiger Elemente."
In: *Annalen der Physik* 344.12 (1912), pp. 257–306.

[96] XY Gu, CY Dong, and T Cheng.
"MPM simulations of high-speed machining of Ti6Al4V titanium alloy considering dynamic recrystallization phenomenon and thermal conductivity."
In: *Applied Mathematical Modelling* (2017).

[97] Jim Guilkey et al. *Uintah user guide*. Tech. rep. SCI Institute Technical Report, 2009.

[98] YB Guo, Q Wen, and KA Woodbury. "Dynamic material behavior modeling using internal state variable plasticity and its application in hard machining simulations."
In: *Journal of Manufacturing Science and Engineering* 128.3 (2006), pp. 749–759.

[99] John O Hallquist et al. "LS-DYNA theory manual."
In: *Livermore software Technology corporation* 3 (2006), pp. 25–31.

[100] William Rowan Hamilton. *Lectures on quaternions*. Hodges and Smith, 1853.

[101] Lianghao Han et al. "Fast deformation simulation of breasts using GPU-based dynamic explicit finite element method."
In: *International Workshop on Digital Mammography*. Springer. 2010, pp. 728–735.

[102] Francis H Harlow and MW Evans.
"A machine calculation method for hydrodynamic problems." In: *LAMS-1956* (1955).

[103] Alireza Hashemian and Hossein Shodja.
"Gradient reproducing kernel particle method."
In: *Journal of Mechanics of Materials and Structures* 3.1 (2008), pp. 127–152.

[104] Martin Heinstein and Dan Segalman.
"Simulation of orthogonal cutting with smooth particle hydrodynamics."
In: *Sandia National Laboratories* (1997).

[105] Hibbitt, Karlsson, and Sorensen. *ABAQUS: Theory manual*. Hibbitt, Karlsson & Sorensen, 1997.

[106] Simone E Hieber and Petros Koumoutsakos. "A Lagrangian particle level set method." In: *Journal of Computational Physics* 210.1 (2005), pp. 342–367.

[107] Simone E Hieber and Petros Koumoutsakos. "A Lagrangian particle method for the simulation of linear and nonlinear elastic models of soft tissue." In: *Journal of Computational Physics* 227.21 (2008), pp. 9195–9215.

[108] Simone E Hieber, Jens H Walther, and Petros Koumoutsakos. "Remeshed smoothed particle hydrodynamics simulation of the mechanical behavior of human organs." In: *Technology and Health Care* 12.4 (2004), pp. 305–314.

[109] David Hilbert. "Ueber die stetige Abbildung einer Line auf ein Flachenstuck." In: *Mathematische Annalen* 38.3 (1891), pp. 459–460.

[110] Rodney Hill. "Aspects of invariance in solid mechanics." In: *Advances in applied mechanics*. Vol. 18. Elsevier, 1979, pp. 1–75.

[111] Charles AR Hoare. "Quicksort." In: *The Computer Journal* 5.1 (1962), pp. 10–16.

[112] Roger W Hockney and James W Eastwood. *Computer simulation using particles*. crc Press, 1988.

[113] RC Hoetzlein. "Fast fixed-radius nearest neighbors: interactive million-particle fluids." In: *GPU Technology Conference*. 2014, p. 18.

[114] Jin Hongbin and Ding Xin. "On criterions for smoothed particle hydrodynamics kernels in stable field." In: *Journal of Computational Physics* 202.2 (2005), pp. 699–709.

[115] S Inutsuka. "Godunov-type SPH." In: *Memorie della Societa Astronomica Italiana* 65 (1994), p. 1027.

[116] EN ISO. "6892-1. Metallic materials-Tensile testing-Part 1: Method of test at room temperature." In: *International Organization for Standardization* (2009).

[117] Mazen Issa et al. "Prediction of serrated chip formation in orthogonal metal cutting by advanced adaptive 2D numerical methodology." In: *International Journal of Machining and Machinability of Materials* 9.3-4 (2011), pp. 295–315.

[118] Gustav Jaumann. "Geschlossenes System physikalischer und chemischer Differentialgesetze." In: *Sitzungsberichte Akad. Wiss. Wien, IIa* (1911), pp. 385–530.

[119] IS Jawahir et al. "Surface integrity in material removal processes: Recent advances." In: *CIRP Annals-Manufacturing Technology* 60.2 (2011), pp. 603–626.

[120] Xiaozhong Jin, Gang Li, and NR Aluru. "On the equivalence between least-squares and kernel approximations in meshless methods." In: *Computer Modeling in Engineering and sciences* 2.4 (2001), pp. 447–462.

[121] Xia Jin et al. "Meshless algorithm for soft tissue cutting in surgical simulation." In: *Computer methods in biomechanics and biomedical engineering* 17.7 (2014), pp. 800–811.

[122] Gordon R Johnson. "Artificial viscosity effects for SPH impact computations." In: *International Journal of Impact Engineering* 18.5 (1996), pp. 477–488.

[123] Gordon R Johnson and William H Cook. "A constitutive model and data for metals subjected to large strains, high strain rates and high temperatures." In: *Proceedings of the 7th International Symposium on Ballistics*. Vol. 21. 1. The Netherlands. 1983, pp. 541–547.

[124] Gordon R Johnson and William H Cook. "Fracture characteristics of three metals subjected to various strains, strain rates, temperatures and pressures." In: *Engineering fracture mechanics* 21.1 (1985), pp. 31–48.

[125] John Edward Jones. "On the determination of molecular fields.—II. From the equation of state of a gas." In: *Proc. R. Soc. Lond. A* 106.738 (1924), pp. 463–477.

[126] Pierre Joyot et al. "A numerical simulation of steady state metal cutting." In: *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science* 212.5 (1998), pp. 331–341.

[127] Sukky Jun, Wing Kam Liu, and Ted Belytschko. "Explicit reproducing kernel particle methods for large deformation problems." In: *International Journal for Numerical Methods in Engineering* 41.1 (1998), pp. 137–166.

[128] Fadi Kahwash, Islam Shyha, and Alireza Maheri. "Meshfree formulation for modelling of orthogonal cutting of composites." In: *Composite Structures* 166 (2017), pp. 193–201.

[129] Malvin H Kalos and Paula A Whitlock. *Monte carlo methods*. John Wiley & Sons, 2008.

[130] Donald Ervin Knuth. *The art of computer programming*. Vol. 3. Pearson Education, 1997.

[131] Peter Kogge. "The tops in flops." In: *IEEE Spectrum* 48.2 (2011).

[132] SP Korzilius, WHA Schilders, and MJH Anthonissen. "An Improved CSPM Approach for Accurate Second-Derivative Approximations with SPH." In: *Journal of Applied Mathematics and Physics* 5.01 (2016), p. 168.

[133] KRISTIAN Krabbenhøft. "Basic computational plasticity." In: *Lecture Notes* (2002).

[134] RD Krieg. "A practical two surface plasticity theory." In: *Journal of applied mechanics* 42.3 (1975), pp. 641–646.

[135] E Kullig and S Wippler. "Numerical integration and FEM-implementation of a viscoplastic Chaboche-model with static recovery." In: *Computational Mechanics* 38.6 (2006), pp. 1–13.

[136] Fredy Kuster et al. "Simulation zur Optimierung von Schleifwerkzeugen mit definierter Kornanordnung." In: *Diamond business* 2010.2 (2010), pp. 28–33.

[137] Mathew Kuttolamadom et al. "High performance computing simulations to identify process parameter designs for profitable titanium machining." In: *Journal of Manufacturing Systems* 43 (2017), pp. 235–247.

[138] E Scott Larsen and David McAllister. "Fast matrix multiplies using graphics hardware." In: *Proceedings of the 2001 ACM/IEEE conference on Supercomputing*. ACM. 2001, pp. 55–55.

[139] Guangyao Li and Ted Belytschko.
"Element-free Galerkin method for contact problems in metal forming analysis."
In: *Engineering Computations* 18.1/2 (2001), pp. 62–78.

[140] Shaofan Li and Wing Kam Liu. *Meshfree particle methods*.
Springer Science & Business Media, 2007.

[141] Larry D Libersky and AG Petschek.
"Smooth particle hydrodynamics with strength of materials."
In: *Advances in the free-Lagrange method including contributions on adaptive gridding and the smooth particle hydrodynamics method*. Springer, 1991, pp. 248–257.

[142] Larry D. Libersky et al. "High Strain Lagrangian Hydrodynamics: A Three-Dimensional SPH Code for Dynamic Material Response."
In: *Journal of Computational Physics* 109.1 (1993), pp. 67–75. ISSN: 0021-9991.

[143] Larry D Libersky et al.
"Recent improvements in SPH modeling of hypervelocity impact."
In: *International Journal of Impact Engineering* 20.6 (1997), pp. 525–532.

[144] LarryD. Libersky and A.G. Petschek.
"Smooth particle hydrodynamics with strength of materials."
In: *Advances in the Free-Lagrange Method Including Contributions on Adaptive Gridding and the Smooth Particle Hydrodynamics Method*.
Ed. by HaroldE. Trease, MartinF. Fritts, and W.Patrick Crowley. Vol. 395.
Lecture Notes in Physics. Springer Berlin Heidelberg, 1991, pp. 248–257.
ISBN: 978-3-540-54960-4.

[145] Jérôme Limido et al.
"A new approach of high speed cutting modelling: SPH method."
In: *Journal de Physique IV (Proceedings)*. Vol. 134. EDP sciences. 2006, pp. 1195–1200.

[146] Jérôme Limido et al. "SPH method applied to high speed cutting modelling."
In: *International journal of mechanical sciences* 49.7 (2007), pp. 898–908.

[147] MB Liu et al. "Computer simulation of high explosive explosion using smoothed particle hydrodynamics methodology."
In: *Computers & fluids* 32.3 (2003), pp. 305–322.

[148] MB Liu et al. "Numerical simulation of incompressible flows by SPH."
In: *International Conference on Scientific & Engineering Computational, Beijing*. 2001.

[149] MB Liu et al. "Smoothed particle hydrodynamics for numerical simulation of underwater explosion." In: *Computational Mechanics* 30.2 (2003), pp. 106–118.

[150] Wing Kam Liu et al. "Multiresolution reproducing kernel particle method for computational fluid dynamics."
In: *International journal for numerical methods in fluids* 24.12 (1997), pp. 1391–1415.

[151] Wing Kam Liu, Sukky Jun, Yi Fei Zhang, et al.
"Reproducing kernel particle methods."
In: *International journal for numerical methods in fluids* 20.8-9 (1995), pp. 1081–1106.

[152] WK Liu et al. "Reproducing kernel particle for structural dynamics."
In: *Int J Numer Methods Fluid* 38 (1995), pp. 1655–79.

[153] I Llanos et al. "Finite element modeling of oblique machining using an arbitrary Lagrangian–Eulerian formulation."
In: *Machining science and technology* 13.3 (2009), pp. 385–406.

[154] Leon B Lucy. "A numerical approach to the testing of the fission hypothesis."
In: *The astronomical journal* 82 (1977), pp. 1013–1024.

[155] Jaroslav Mackerle.
"Finite-element analysis and simulation of machining: a bibliography (1976–1996)."
In: *Journal of Materials Processing Technology* 86.1 (1999), pp. 17–44.

[156] Martin Madaj and Miroslav Piška.
"On the SPH orthogonal cutting simulation of A2024-T351 alloy."
In: *Procedia CIRP* 8 (2013), pp. 152–157.

[157] Lawrence E Malvern. *Introduction to the Mechanics of a Continuous Medium*. Monograph. 1969.

[158] Salvatore Marrone et al.
"Fast free-surface detection and level-set function definition in SPH solvers."
In: *Journal of Computational Physics* 229.10 (2010), pp. 3652–3663.

[159] TD Marusich and Modeling Ortiz.
"Modelling and simulation of high-speed machining." In: *International Journal for Numerical Methods in Engineering* 38.21 (1995), pp. 3675–3694.

[160] Vishal Mehra et al.
"Tensile instability and artificial stresses in impact problems in SPH."
In: *Journal of Physics: Conference Series*. Vol. 377. 1. IOP Publishing. 2012, p. 012102.

[161] ME Merchant.
"An interpretive look at 20th century research on modeling of machining."
In: *Machining Science and Technology* 2.2 (1998), pp. 157–163.

[162] Hubert W Meyer Jr and David S Kleponis. *An analysis of parameters for the Johnson-Cook strength model for 2-in-thick Rolled Homogeneous Armor*. Tech. rep. Army Research Lab Aberdeen Proving Ground MD, 2001.

[163] Gustav Mie. "Zur kinetischen Theorie der einatomigen Körper."
In: *Annalen der Physik* 316.8 (1903), pp. 657–697.

[164] A V Mitrofanov, V I Babitsky, and Vadim V Silberschmidt.
"Finite element simulations of ultrasonically assisted turning."
In: *Computational materials science* 28.3-4 (2003), pp. 645–653.

[165] Alain Molinari, Rachid Cheriguene, and Henar Miguélez.
"Numerical and analytical modeling of orthogonal cutting: the link between local variables and global contact characteristics."
In: *International journal of mechanical sciences* 53.3 (2011), pp. 183–206.

[166] JJ Monaghan. "On the problem of penetration in particle methods."
In: *Journal of Computational physics* 82.1 (1989), pp. 1–15.

[167] JJ Monaghan. "SPH and Riemann solvers."
In: *Journal of Computational Physics* 136.2 (1997), pp. 298–307.

[168] JJ Monaghan. "SPH compressible turbulence."
In: *Monthly Notices of the Royal Astronomical Society* 335.3 (2002), pp. 843–852.

[169] JJ Monaghan and RA Gingold. "Shock simulation by the particle method SPH."
In: *Journal of computational physics* 52.2 (1983), pp. 374–389.

[170] Joe J Monaghan. "Simulating free surface flows with SPH."
In: *Journal of computational physics* 110.2 (1994), pp. 399–406.

[171] Joe J Monaghan. "Smoothed particle hydrodynamics."
In: *Reports on progress in physics* 68.8 (2005), p. 1703.

[172] Joseph J Monaghan. "SPH without a tensile instability."
In: *Journal of Computational Physics* 159.2 (2000), pp. 290–311.

[173] Gordon E Moore. "Cramming more components onto integrated circuits."
In: *Proceedings of the IEEE* 86.1 (1998), pp. 82–85.

[174] Guy M Morton.
"A computer oriented geodetic data base and a new technique in file sequencing."
In: (1966).

[175] Zi Mroz. "On the description of anisotropic workhardening."
In: *Journal of the Mechanics and Physics of Solids* 15.3 (1967), pp. 163–175.

[176] Matthias Müller et al. "Interaction of fluids with deformable solids."
In: *Computer Animation and Virtual Worlds* 15.3-4 (2004), pp. 159–171.

[177] Matthias Müller et al. "Meshless deformations based on shape matching."
In: *ACM Transactions on Graphics (TOG)* 24.3 (2005), pp. 471–478.

[178] Matthias Müller et al. "Point based animation of elastic, plastic and melting objects."
In: *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Eurographics Association. 2004, pp. 141–151.

[179] Johannes Catharinus Nagtegaal, David Moore Parks, and JR Rice.
"On numerically accurate finite element solutions in the fully plastic range."
In: *Computer methods in applied mechanics and engineering* 4.2 (1974), pp. 153–177.

[180] B Nayroles, G Touzot, and P Villon. "Generalizing the finite element method: diffuse approximation and diffuse elements."
In: *Computational mechanics* 10.5 (1992), pp. 307–318.

[181] Vinh Phu Nguyen et al.
"Meshless methods: a review and computer implementation aspects."
In: *Mathematics and computers in simulation* 79.3 (2008), pp. 763–813.

[182] Gan Nianfei, Li Guangyao, and Long Shuyao. "3D adaptive RKPM method for contact problems with elastic–plastic dynamic large deformation."
In: *Engineering analysis with boundary elements* 33.10 (2009), pp. 1211–1222.

[183] Weilong Niu et al. "Modeling of orthogonal cutting process of A2024-T351 with an improved SPH method." In: *The International Journal of Advanced Manufacturing Technology* 95.1-4 (2018), pp. 905–919.

[184] CUDA Nvidia. "Compute unified device architecture programming guide."
In: (2007).

[185] CUDA Nvidia. "NVIDIA Tesla P100 Whitepaper." In: (2016).

[186] CUDA Nvidia. *Programming guide*. 2010.

[187] George G O'Brien, Morton A Hyman, and Sidney Kaplan. "A study of the numerical solution of partial differential equations." In: *Studies in Applied Mathematics* 29.1-4 (1950), pp. 223–251.

[188] Atsuyuki Okabe et al. *Spatial tessellations: concepts and applications of Voronoi diagrams*. Vol. 501. John Wiley & Sons, 2009.

[189] Lars Olovsson, Larsgunnar Nilsson, and Kjell Simonsson. "An ALE formulation for the solution of two-dimensional metal cutting problems." In: *Computers & structures* 72.4-5 (1999), pp. 497–507.

[190] Tahsin Tecelli Opoz and Xun Chen. "Numerical simulation of single grit grinding." In: (2010).

[191] Miguel Ortiz and Peter M Pinsky. *Global analysis methods for the solution of the elastoplastic and viscoplastic dynamic problems*. Tech. rep. California Univ., Berkeley (USA). Dept. of Civil Engineering, 1982.

[192] H Ostad and S Mohammadi. "A stabilized particle method for large deformation dynamic analysis of structures." In: *International Journal of Structural Stability and Dynamics* 12.04 (2012), p. 1250026.

[193] Hassan Ostad-Hossein and Soheil Mohammadi. "A field smoothing stabilization of particle methods in elastodynamics." In: *Finite Elements in Analysis and Design* 44.9 (2008), pp. 564–579.

[194] J Michael Owen et al. "Adaptive smoothed particle hydrodynamics: Methodology. II." In: *The Astrophysical Journal Supplement Series* 116.2 (1998), p. 155.

[195] Tugrul Özel. "The influence of friction models on finite element simulations of machining." In: *International Journal of Machine Tools and Manufacture* 46.5 (2006), pp. 518–530.

[196] Tuğrul Özel and Erol Zeren. "Finite element method simulation of machining of AISI 1045 steel with a round edge cutting tool." In: *Proceedings of the 8th CIRP International Workshop on Modeling of Machining Operations*. 2005, pp. 533–542.

[197] T Özel et al. "Investigations on the effects of multi-layered coated inserts in machining Ti–6Al–4V alloy with experiments and finite element simulations." In: *CIRP Annals-Manufacturing Technology* 59.1 (2010), pp. 77–82.

[198] Olivier Pantalé et al. "2D and 3D numerical models of metal cutting with damage effects." In: *Computer methods in applied mechanics and engineering* 193.39-41 (2004), pp. 4383–4399.

[199] Gi-Ho Park, Klaus Krohne, and Er Ping Li. "Introduction to the smoothed particle hydrodynamics method in electromagnetics." In: *Electromagnetic Compatibility and 19th International Zurich Symposium on Electromagnetic Compatibility, 2008. APEMC 2008. Asia-Pacific Symposium on*. IEEE. 2008, pp. 582–585.

[200] Anatoly N Parshikov and Stanislav A Medin.
"Smoothed particle hydrodynamics using interparticle contact algorithms."
In: *Journal of computational physics* 180.1 (2002), pp. 358–382.

[201] HD Patterson. "The errors of lattice sampling."
In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1954), pp. 140–149.

[202] FW Pinto, GE Vargas, and Konrad Wegener.
"Simulation for optimizing grain pattern on Engineered Grinding Tools."
In: *CIRP Annals-Manufacturing Technology* 57.1 (2008), pp. 353–356.

[203] Stéphan Popinet. *The GNU triangulated surface library*. 2012.

[204] William H Press et al. *Numerical recipes in C*. Vol. 2.
Cambridge university press Cambridge, 1996.

[205] Daniel J Price. "Smoothed particle hydrodynamics and magnetohydrodynamics."
In: *Journal of Computational Physics* 231.3 (2012), pp. 759–794.

[206] Daniel J Price. "SPLASH: An interactive visualisation tool for Smoothed Particle Hydrodynamics simulations."
In: *Publications of the Astronomical Society of Australia* 24.3 (2007), pp. 159–173.

[207] JM Rodriguez Prieto et al.
"Generation of segmental chips in metal cutting modeled with the PFEM."
In: *Computational Mechanics* (2017), pp. 1–17.

[208] Kunal Puri and Prabhu Ramachandran.
"Approximate Riemann solvers for the Godunov SPH (GSPH)."
In: *Journal of Computational Physics* 270 (2014), pp. 432–458.

[209] Nathan J Quinlan, Mihai Basa, and Martin Lastiwka.
"Truncation error in mesh-free particle methods." In: *International Journal for Numerical Methods in Engineering* 66.13 (2006), pp. 2064–2085.

[210] T Rabczuk, T Belytschko, and SP Xiao.
"Stable particle methods based on Lagrangian kernels."
In: *Computer methods in applied mechanics and engineering* 193.12 (2004), pp. 1035–1063.

[211] T Rabczuk and E Samaniego.
"Discontinuous modelling of shear bands using adaptive meshfree methods."
In: *Computer Methods in Applied Mechanics and Engineering* 197.6-8 (2008), pp. 641–658.

[212] A Rahman. "Correlations in the motion of atoms in liquid argon."
In: *Physical Review* 136.2A (1964), A405.

[213] PW Randles and LD Libersky. "Normalized SPH with stress points." In: *International Journal for Numerical Methods in Engineering* 48.10 (2000), pp. 1445–1462.

[214] PW Randles and LD Libersky.
"Smoothed particle hydrodynamics: some recent improvements and applications."
In: *Computer methods in applied mechanics and engineering* 139.1 (1996), pp. 375–408.

[215] PW Randles, LD Libersky, and AG Petschek.
On neighbors, derivatives, and viscosity in particle codes. Tech. rep.
Los Alamos National Lab., NM (US), 1999.

[216] Karl-A Reckling. *Plastizitätstheorie und ihre Anwendung auf Festigkeitsprobleme*. Springer-Verlag, 2013.

[217] Juan R Reveles. "Development of a total Lagrangian SPH code for the simulation of solids under dynamioc loading." In: (2007).

[218] Robert D Richtmyer and Keith W Morton. "Difference methods for initial-value problems." In: *Malabar, Fla.: Krieger Publishing Co.,— c1994, 2nd ed.* (1994).

[219] JM Rodriguez, Pär Jonsén, and Ales Svoboda. "Simulation of metal cutting using the particle finite-element method and a physically based plasticity model." In: *Computational Particle Mechanics* 4.1 (2017), pp. 35–51.

[220] Juan Manuel Rodriguez, Pär Jonsén, and Ales Svoboda. "Dislocation Density Based Material Model Applied in PFEM-simulation of Metal Cutting." In: *Procedia CIRP* 58 (2017), pp. 193–197.

[221] Juan Manuel Rodriguez et al. "Continuous chip formation in metal cutting processes using the Particle Finite Element Method (PFEM)." In: *International Journal of Solids and Structures* 120 (2017), pp. 81–102.

[222] J Rodriguez et al. "A sensibility analysis to geometric and cutting conditions using the particle finite element method (PFEM)." In: *Procedia CIRP* 8 (2013), pp. 105–110.

[223] Philip L Roe. "Approximate Riemann solvers, parameter vectors, and difference schemes." In: *Journal of computational physics* 43.2 (1981), pp. 357–372.

[224] Matthias Röthlin. "Adaptive Particles for Real-time Turbulent Fluids." MA thesis. 2012.

[225] Niklaus Rüttimann. "Simulation of metal cutting processes using meshfree methods." PhD thesis. Diss., Eidgenössische Technische Hochschule ETH Zürich, Nr. 20646, 2012.

[226] Niklaus Ruttimann, Sebastian Buhl, and Konrad Wegener. "Simulation of single grain cutting using SPH method." In: *Journal of Machine Engineering* 10 (2010).

[227] N Rüttimann et al. "Simulation of hexa-octahedral diamond grain cutting tests using the SPH method." In: *Procedia CIRP* 8 (2013), pp. 322–327.

[228] Nadathur Satish, Mark Harris, and Michael Garland. "Designing efficient sorting algorithms for manycore GPUs." In: *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*. IEEE. 2009, pp. 1–10.

[229] RG Sauvú and GD Morandin. "Simulation of contact in finite deformation problems–algorithm." In: *International Journal of Mechanics and Materials in Design* 1.3 (2004), pp. 287–316.

[230] Jürg Schläfli. "Simulation of fluid-solid interaction." MA thesis. 2005.

[231] Philip Schneider and David H Eberly. *Geometric tools for computer graphics*. Elsevier, 2002.

[232] GS Sekhon and JL Chenot. "Numerical simulation of continuous chip formation during non-steady orthogonal cutting."
In: *Engineering computations* 10.1 (1993), pp. 31–48.

[233] James A Sethian.
"A fast marching level set method for monotonically advancing fronts."
In: *Proceedings of the National Academy of Sciences* 93.4 (1996), pp. 1591–1595.

[234] Paul R Shapiro et al. "Adaptive smoothed particle hydrodynamics, with application to cosmology: methodology."
In: *The Astrophysical Journal Supplement Series* 103 (1996), p. 269.

[235] Milton Clayton Shaw and JO Cookson. *Metal cutting principles*.
Clarendon press Oxford, 1984.

[236] Donald Shepard.
"A two-dimensional interpolation function for irregularly-spaced data."
In: *Proceedings of the 1968 23rd ACM national conference*. ACM. 1968, pp. 517–524.

[237] Chandrakanth Shet and Xiaomin Deng.
"Finite element analysis of the orthogonal metal cutting process."
In: *Journal of Materials Processing Technology* 105.1-2 (2000), pp. 95–109.

[238] Chandrakanth Shet and Xiaomin Deng.
"Residual stresses and strains in orthogonal metal cutting."
In: *International Journal of Machine Tools and Manufacture* 43.6 (2003), pp. 573–587.

[239] Guoqin Shi, Xiaomin Deng, and Chandrakanth Shet.
"A finite element study of the effect of friction in orthogonal metal cutting."
In: *Finite Elements in Analysis and Design* 38.9 (2002), pp. 863–883.

[240] T Shirakashi and E Usui. "Simulation analysis of orthogonal metal cutting process."
In: *J. Japan Soc. Prec. Eng* 42.5 (1976), pp. 340–345.

[241] Juan C Simo and Thomas JR Hughes. *Computational inelasticity*. Vol. 7.
Springer Science & Business Media, 2006.

[242] Darko Smolenicki.
"Chip formation analysis of innovative graphitic steel in drilling processes."
PhD thesis. ETH Zurich, 2017.

[243] Barbara Solenthaler, Jürg Schläfli, and Renato Pajarola.
"A unified particle model for fluid–solid interactions."
In: *Computer Animation and Virtual Worlds* 18.1 (2007), pp. 69–82.

[244] F Spreng and P Eberhard. "Modeling of orthogonal metal cutting using adaptive smoothed particle hydrodynamics." In: *Thermal Effects in Complex Machining Processes*. Springer, 2018, pp. 133–143.

[245] Fabian Spreng and Peter Eberhard.
"Machining process simulations with smoothed particle hydrodynamics."
In: *Procedia CIRP* 31 (2015), pp. 94–99.

[246] Fabian Spreng, Peter Eberhard, and Florian Fleissner.
"An approach for the coupled simulation of machining processes using multibody system and smoothed particle hydrodynamics algorithms."
In: *Theoretical and Applied Mechanics Letters* 3.1 (2013).

[247] Fabian Spreng et al.
"A local adaptive discretization algorithm for Smoothed Particle Hydrodynamics."
In: *Computational Particle Mechanics* 1.2 (2014), pp. 131–145.

[248] Volker Springel and Lars Hernquist.
"Cosmological smoothed particle hydrodynamics simulations: the entropy equation."
In: *Monthly Notices of the Royal Astronomical Society* 333.3 (2002), pp. 649–664.

[249] Anil K Srivastava and Jon Iverson. "An experimental investigation into the high speed turning of Ti-6Al-4V titanium alloy." In: *Int Manuf Sci Eng Conf*. 2010, pp. 401–408.

[250] Jos Stam. "Stable fluids." In: *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co. 1999, pp. 121–128.

[251] Alexey Stomakhin et al. "A material point method for snow simulation."
In: *ACM Transactions on Graphics (TOG)* 32.4 (2013), p. 102.

[252] John S Strenkowski and Kyoung-Jin Moon.
"Finite element prediction of chip geometry and tool/workpiece temperature distributions in orthogonal metal cutting."
In: *Journal of Engineering for Industry* 112.4 (1990), pp. 313–318.

[253] Deborah Sulsky, Zhen Chen, and Howard L Schreyer.
"A particle method for history-dependent materials."
In: *Computer methods in applied mechanics and engineering* 118.1-2 (1994), pp. 179–196.

[254] J Swegle. "SPH in tension."
In: *Memo, Sandia National Laboratories, Albuque, USA* (1992).

[255] JW Swegle, DL Hicks, and SW Attaway.
"Smoothed particle hydrodynamics stability analysis."
In: *Journal of computational physics* 116.1 (1995), pp. 123–134.

[256] JW Swegle et al. *An analysis of smoothed particle hydrodynamics*. Tech. rep.
Sandia National Labs., Albuquerque, NM (United States), 1994.

[257] AO Tay, MG Stevenson, and G de Vahl Davis. "Using the finite element method to determine temperature distributions in orthogonal machining."
In: *Proceedings of the institution of mechanical engineers* 188.1 (1974), pp. 627–638.

[258] Frederick Winslow Taylor. "On the art of cutting metals."
In: *On the Art of Cutting Metals, ASME* 28 (1907).

[259] Zeike A Taylor, Mario Cheng, and Sébastien Ourselin. "High-speed nonlinear finite element analysis for surgical simulation using graphics processing units."
In: *IEEE transactions on medical imaging* 27.5 (2008), pp. 650–663.

[260] Matthias Teschner et al.
"Optimized Spatial Hashing for Collision Detection of Deformable Objects." In: *Vmv*. Vol. 3. 2003, pp. 47–54.

[261] Grit Thürrner and Charles A Wüthrich.
"Computing vertex normals from polygonal facets."
In: *Journal of Graphics Tools* 3.1 (1998), pp. 43–46.

[262] Francesco Tornabene, Nicholas Fantuzzi, and Michele Bacciocchi.
"The strong formulation finite element method: stability and accuracy."
In: *Frattura ed Integrità Strutturale* 29 (2014), p. 251.

[263] Eleuterio F Toro.
Riemann solvers and numerical methods for fluid dynamics: a practical introduction.
Springer Science & Business Media, 2013.

[264] Clifford Truesdell. "The mechanical foundations of elasticity and fluid dynamics."
In: *Journal of Rational Mechanics and Analysis* 1 (1952), pp. 125–300.

[265] Domenico Umbrello. "Finite element simulation of conventional and high speed machining of Ti6Al4V alloy."
In: *Journal of materials processing technology* 196.1-3 (2008), pp. 79–87.

[266] E Usui, T Shirakashi, and T Kitagawa. "Analytical prediction of three dimensional cutting process Part 3: Cutting temperature and crater wear of carbide tool."
In: *Journal of Engineering for industry* 100.2 (1978), pp. 236–243.

[267] Yolanda Vidal Segui, J Bonet, and Antonio Huerta.
"Stabilized updated Lagrangian corrected SPH for explicit dynamic problems."
In: *International journal for numerical methods in engineering* 69.13 (2007), pp. 2687–2710.

[268] R Vignjevic, J Campbell, and L Libersky. "A treatment of zero-energy modes in the smoothed particle hydrodynamics method."
In: *Computer methods in Applied mechanics and Engineering* 184.1 (2000), pp. 67–85.

[269] Morten F Villumsen and Torben G Fauerholdt.
"Simulation of metal cutting using smooth particle hydrodynamics."
In: *LS-DYNA Anwenderforum, C-III* (2008), p. 17.

[270] Georges Voronoi.
"Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Deuxième mémoire. Recherches sur les parallélloèdres primitifs."
In: *Journal für die reine und angewandte Mathematik* 134 (1908), pp. 198–287.

[271] Wei Min Wang.
"Stationary and propagative instabilities in metals-a computational point of view."
In: (1997).

[272] Konrad Wegener. "Zur Berechnung grosser plastischer Deformationen mit einem Stoffgesetz vom Überspannungstyp." In: (1991).

[273] Konrad Wegener et al. "An Experimental Approach for Grain Wear for Improving Kinematic-Geometrical Simulation." In: *Proceedings of the 10th anniversary international conference of the European Society for Precision Engineering and Nanotechnology. 1*. Euspen. 2008, pp. 130–134.

[274] Gerald Wempner.
"Finite elements, finite rotations and small strains of flexible shells."
In: *International Journal of Solids and Structures* 5.2 (1969), pp. 117–153.

[275] Mark L Wilkins. *Calculation of elastic-plastic flow*. Tech. rep.
California Univ Livermore Radiation Lab, 1963.

[276] Samuel Williams, Andrew Waterman, and David Patterson.
"Roofline: an insightful visual performance model for multicore architectures."
In: *Communications of the ACM* 52.4 (2009), pp. 65–76.

[277] A Winnicki, CJ Pearce, and N Bićanić.
"Viscoplastic Hoffman consistency model for concrete."
In: *Computers & Structures* 79.1 (2001), pp. 7–19.

[278] HB Wu and SJ Zhang.
"3D FEM simulation of milling process for titanium alloy Ti6Al4V." In: *The International Journal of Advanced Manufacturing Technology* 71.5-8 (2014), pp. 1319–1326.

[279] Carl-Frederik Wyen.
"Rounded cutting edges and their influence in machining titanium."
PhD thesis. ETH-Zürich, 2011.

[280] GG Ye et al.
"Critical cutting speed for onset of serrated chip flow in high speed machining."
In: *International Journal of Machine Tools and Manufacture* 86 (2014), pp. 18–33.

[281] S Abolfazl Zahedi et al.
"FE/SPH modelling of orthogonal micro-machining of fcc single crystal."
In: *Computational Materials Science* 78 (2013), pp. 104–109.

[282] Chi Zhang, Xiangyu Y Hu, and Nikolaus A Adams.
"A generalized transport-velocity formulation for smoothed particle hydrodynamics."
In: *Journal of Computational Physics* 337 (2017), pp. 216–232.

[283] GY Zhang et al. "A three-dimensional nonlinear meshfree algorithm for simulating mechanical responses of soft tissue."
In: *Engineering Analysis with Boundary Elements* 42 (2014), pp. 60–66.

[284] Yancheng Zhang, JC Outeiro, and Tarek Mabrouki.
"On the selection of Johnson-Cook constitutive model parameters for Ti-6Al-4 V using three types of numerical models of orthogonal cutting."
In: *Procedia CIRP* 31 (2015), pp. 112–117.

Basics notation and operators

Symbol	Meaning
x	no underline - scalar
$\underline{x}$	single underline, vector
$\underline{\underline{X}}$	double underline, uppercase - 2nd order tensor
$\underline{\underline{\underline{X}}}$	triple underline, uppercase - 3rd order tensor
$\underset{\sim}{\underset{\sim}{X}}$	double squiggly underline, uppercase - 4th order tensor
$\tilde{\int}_\Omega f(\underline{x}) d\underline{x}$	Volume integral over domain Ω
$\underline{\underline{Z}} = \underline{x} \otimes \underline{y}$	outer product, $Z_{ij} = x_i y_j$
$z = \underline{x} \cdot \underline{y}$	inner product, $z = x_i \cdot y_i$
$\underline{z} = \underline{\underline{X}} \cdot \underline{y}$	inner product, $z_i = X_{ij} \cdot y_y$
$z = \underline{\underline{X}} : \underline{\underline{Y}}$	double contraction $z = X_{ij} \cdot Y_{ij}$
$\underline{z} = \underline{x} \odot \underline{y}$	Hadamart product, i.e. component-wise multiplication
$tr(\underline{\underline{X}})$	trace of tensor, $tr(\underline{\underline{X}}) = X_{ii}$
$vol(\underline{\underline{X}})$	volumetric part of tensor, $vol(\underline{\underline{X}}) = 1/3 \cdot tr(\underline{\underline{X}}) \cdot \underline{\underline{I}}$
$dev(\underline{\underline{X}})$	deviatoric part of tensor, $dev(\underline{\underline{X}}) = \underline{\underline{X}} - 1/3 \cdot tr(\underline{\underline{X}}) \cdot \underline{\underline{I}}$
∇	Vector containing partial derivatives of coordinates, e.g. $\nabla = [\partial/(\partial x), \partial/(\partial y), \partial/(\partial z)]$ in 3D.
Δx	Increment in quantity x, e.g. time increment Δt
$a \propto b$	Quantitiy a is proportional to quantitiy b
$\lvert x \rvert$	Absolute value of x
$\lvert\lvert \underline{x} \rvert\rvert$	Length of vector of $\lvert\lvert \underline{x} \rvert\rvert = \sqrt{(x_i \cdot x_i)}$
$\lvert\lvert \underline{\underline{X}} \rvert\rvert$	"Length" of tensor of $\lvert\lvert \underline{\underline{X}} \rvert\rvert = \sqrt{\underline{\underline{X}} : \underline{\underline{X}}} = \sqrt{X_{ij} \cdot X_{ij}}$
$\mathcal{O}(\cdot)$	Big-O Notation
$< \cdot >$	Approximated quantity
$\dot{x}$	x derived with regard to time
$\lfloor x \rfloor$	round down
$(.)^R$	projected quantity, usually projected onto inter-particle distance
$(.)^*$	quantity obtained by solution of a Riemann problem
$\hat{\underline{\underline{X}}}$	tensor $\underline{\underline{X}}$ in rotated frame

Specific symbols, roughly in order of appearance in the thesis

Symbol	Meaning
$\underline{x}$	current configuration
$\underline{X}$	reference configuration
$\phi(\cdot)$	mapping between current and reference configuration
$\underline{u}$	displacement

$\underline{v}$	velocity
$\underline{\underline{F}}$	deformation gradient
$\underline{\underline{L}}$	velocity gradient
$\underline{\underline{D}}$	rate of deformation tensor $\underline{\underline{D}} = 1/2(\underline{\underline{L}} + \underline{\underline{L}}^T)$
$\underline{\underline{W}}$	spin tensor $\underline{\underline{W}} = 1/2(\underline{\underline{L}} - \underline{\underline{L}}^T)$
$\underline{\underline{I}}$	identity matrix
$\nabla^0 \underline{x}$	gradient / divergence w.r.t ref. configuration
$\underline{\underline{R}}$	rotation matrix from polar decomp. of $\underline{\underline{F}} = \underline{\underline{U}} \cdot \underline{\underline{R}}$
$\underline{\underline{U}}$	stretch matrix from polar decomp. of $\underline{\underline{F}} = \underline{\underline{U}} \cdot \underline{\underline{R}}$
$\underline{\underline{\varepsilon}}$	strain tensor, e.g. $\underline{\underline{E}}^{\text{lin}}$, or $\underline{\underline{E}}$, see below.
$\underline{\underline{E}}^{\text{lin}}$	Cauchy strain tensor, $\dot{\underline{\underline{E}}}^{\text{lin}} = \underline{\underline{D}}$
$\underline{\underline{E}}$	Green-St. Venant strain tensor, $\underline{\underline{E}} = 1/2(\underline{\underline{F}} \cdot \underline{\underline{F}}^T) - \underline{\underline{I}}$
$\underline{n}$	surface normal
$\underline{n}_0$	surface normal on reference configuration
$\underline{\underline{\sigma}}$	Cauchy stress tensor
$\underline{\underline{\tilde{\sigma}}}$	Principal stresses, i.e. diagonalization of $\underline{\underline{\sigma}}$
$\underline{f}$	force vector
$\underline{t}$	surface traction
$\underline{\underline{P}}$	Nominal stress tensor
$\underline{\underline{\sigma}}^{\text{PK1}}$	first Piola-Kirchhoff stress tensor, $\underline{\underline{\sigma}}^{\text{PK1}} = \underline{\underline{P}}^T$
$\underline{\underline{\sigma}}^{\text{PK2}}$	second Piola-Kirchhoff stress tensor
M	total mass
ϱ	density
J	determinant of deformation gradient $\underline{\underline{F}}$.
$\underline{\mathcal{P}}$	total linear momentum
$\underline{b}$	body forces
$\underline{\mathcal{F}}$	total body force, i.e. sum of $\underline{b}$
$\overset{\approx}{\underline{\underline{C}}}$	stiffness tensor, with regard to Hooke's law
E	Young's modulus
ν	Poissons's ratio
G	shear modulus, 1st Lamé parameter
K	bulk modulus
λ	2nd Lamé parameter
p	pressure
T	temperature
$\underline{\underline{S}}$	deviatoric stress tensor, $\underline{\underline{S}} = \underline{\underline{\sigma}} - p \cdot \underline{\underline{I}}$
c	speed of sound, speed of sound in reference configuration
c_0	speed of sound in reference configuration
$F(\cdot)$	plastic yield surface
$\underline{\underline{\varepsilon}}_{\text{pl}}$	plastic strain tensor
λ_{pl}	plastic multiplier
$G(\cdot)$	plastic potential (non-associated)
I_{1-3}	invariants
J_{1-3}	simplified invariants for deviatoric stress tensors
$\bar{\varepsilon}_{pl}$, $\dot{\bar{\varepsilon}}_{pl}$	equivalent plastic strain, equivalent plastic strain rate

A, B, C, m, n	Johnson-Cook constants
$\mathcal{G}$	objective function in optimization problem
α^{trml}	thermal diffusivity
c_p	heat capacity
k	thermal conductivity
t_q	Taylor-Quinney constant
η	frictional heating parameter
$\dot{Q}$	heat rate
$\underline{f}_{\text{fric}}$	friction force
$\underline{f}_{\text{cont}}$	contact force
μ	Coulomb friction parameter
$\delta(\cdot)$	Dirac delta function
W_{ij}^h	Kernel function
h	smoothing length
ω_i	discretized integration weight (Riemann sum)
m_i	particle mass
B_i	Bernoulli number
$m_0, \underline{m}_1, \underline{\underline{m}}_2$	first three moments of the kernel
$K(\cdot)$	corrected kernel (in the context of RKPM)
$\psi(\cdot)$	correction function contained in $K(\cdot)$
$\overset{\circ}{W}$	RKPM kernel
$\underline{\underline{B}}$	Randles-Libserky correction tensor
$\tilde{W}(\cdot)$	Shephard corrected version of $W(\cdot)$
$\overset{\triangle}{W}$	Randles-Libersky kernel
$\underline{\underline{\mathcal{C}}}$	Covariance matrix
$\underline{\underline{\mathcal{S}}}$	Trafo from spherical to Cart. coordinates
Π_{ij}	Artificial viscosity term
$\underline{\underline{\Upsilon}}$	Artificial viscosity term according to Vidal
$\alpha^{\text{av}}, \beta^{\text{av}}, \epsilon^{\text{av}}$	artificial viscosity constants
$\underline{\underline{\Theta}}$	Monaghans artificial stress
ϵ^{Stress}	Intensity of artificial stress
ϵ^{XSPH}	Strength of XSPH correction
p_b	background pressure
$\underline{\underline{\mathcal{A}}}$	local transformation matrix
$\mathcal{U}$	strain energy density function
$\underline{\underline{M}}$	mass matrix
κ	contact stiffness
g_N	penetration depth
$\mathcal{H}$	angular momentum computed from discrete particle set
$\tilde{\mathcal{H}}$	angular momentum
$\bar{h}$	smoothing length of repulsive tool particles
γ	rake angle
α	clearance angle
r_c	cutting edge radius

d	uncut chip thickness
D	height of workpiece
$\mathcal{W}$	wear rate
Ψ	Usui wear rate parameter
ξ	Usui wear rate parameter
C_f	Blending parameter between tetra- and octahedron

ACRONYMS

Acronym	Meaning
AABB	Axis Aligned Bounding Box
ALE	Arbitrary Lagrangian Eulerian
ALU	Arithmetic Logic Unit
API	Application Programming Interface
BC	Boundary Condition
BLAS	Basic Linear Algebra Subprograms
CEL	Coupled Eulerian Lagrangian
CFD	Computational Fluid Dynamics
CFL	Courant-Friedrichs-Levy Condition
CPU	Central Processing Unit
CSPM	Corrected Smoothed Particle Method
CUDA	Compute Unified Device Architecture
DP	Dual Precision
ECC	Error Correcting Code
EFG	Element Free Galerkin Method
FE	Finite Element
FEM	Finite Element Method
FLIP	Fluid Implicit Particle
FLOP	FLOating point Operation
FLOPS	FLOating point Operations per Second
GFLOPS	Billion FLOating point Operations per Second
GPGPU	General Purpose Computing on the GPU
GPU	Graphics Processing Unit
GRKPM	Gradient Reproducing Kernel Particle Method
ISO	International Organization for Standardization
KiB	1024 Bytes
LDST	LoaD and STore Units
MIMD	Multiple Instruction Multiple Data
MPM	Material Point Method
MRR	Material Removal Rate
PDE	Partial Differential Equation
PFEM	Particle Finite Element Method
PIC	Particle In Cell (Method)
PLS	Particle Level Set
PSE	Particle Strength Exchange
RAM	Random Access Memory
RKPM	Reproducing Kernel Particle Method
SIMD	Single Instruction Multiple Data
SM	Streaming Processor

SP	Single Precision
SPH	Smoothed Particle Hydrodynamics
SSE	Streaming SIMD Extensions
TFLOP	Trillion FLOP
TFLOPS	Trillion FLOPs
TL	Total Lagrangian
TVF	Transport Velocity Formulation
UL	Updated Lagrangian
UMFPACK	Unsymmetric MultiFrontal PACKage

CURRICULUM VITAE

Personal Details

Name	Matthias Röthlin
Address	Pappelnstrasse 12, 8620 Wetzikon
Citizen	Kerns OW, Switzerland
Marital Status	Unmarried

Education

Nov. 2012 - Oct. 2018	PhD Student & Scientific Staff, IWF, ETHZ
Sept. 2010 - Sept. 2012	Master of Computer Science, ETHZ
Sept. 2005 - Sept. 2010	Bachelor of Computer Science, ETHZ

Work Experience

From Nov. 2018	Scientific staff, inspire AG, Zurich
Sept. 2009 - Nov. 2012	Software and Calculation Engineer, AFC, Zurich

COLOPHON

This document was typeset in LaTeX using the typographical look-and-feel `classicthesis`. The bibliography is typeset using `biblatex`.