Diss. ETH No. 17052

Particle Methods for Flow-Structure Interactions

A dissertation submitted to the

SWISS FEDERAL INSTITUTE OF TECHNOLOGY

ZURICH

for the degree of

DOCTOR OF SCIENCE

presented by

Simone Elke Hieber

Dipl.-Ing., University of Stuttgart, Stuttgart, Germany

M.Sc., Michigan Technological University, Houghton, MI, USA

born on February 19th, 1976

in Geislingen/Steige, Germany

accepted on the recommendation of

Prof. Dr. Petros Koumoutsakos, examiner

Prof. Dr. Anthony Leonard, co-examiner

Prof. Dr. Jens H. Walther, co-examiner

2007

Abstract

The accurate modeling and simulation of soft biological tissue subject to flow-structure interactions lie in the core of virtual surgery systems. The challenges of the computational simulation in virtual surgery are the complex three-dimensional shapes of tissue and the involved physical phenomena including various interactions with the environment, such as interactions with body fluids and medical devices. The simulation of these systems requires the generation of complex structures for discretization. They need to be adaptive and accommodate phenomena such as cutting and flow-structure interactions that are of principal importance for virtual surgery systems. The thesis focuses on the development and implementation of novel particle methods that aim to circumvent some of these difficulties. We distinguish three aspects in the methods developed in here, namely:

1. the development of novel Lagrangian particle level sets for capturing complex, deforming surfaces,

2. the development of particle methods based on regularized Smooth Particle Hydrodynamics for the simulation of fluids and elastic solids,

3. the development of immersed boundary techniques for the simulation of complex, deforming solid boundaries in a fluid environment.

The methods are validated on benchmark problems and tested on showcase systems relevant for virtual surgery applications. The development and implementation of a parallel particle-mesh library allow for large scale simulations of challenging continuum mechanics problems.

Zusammenfassung

Die genaue Modellierung und Simulation von biologischem Weichgewebe kann als die Kerntechnologie in der virtuellen Chirurgie betrachtet werden. Die Herausforderungen der rechnergestützten Simulation im Bereich der virtuellen Chirurgie liegen in der Behandlung von komplexen dreidimensionalen geometrischen Formen des Gewebes und vielfältigen physikalischen Gesetzen, die verschiedene Interaktionsmechanismen mit der Umgebung beinhalten. Die Simulation muss daher adaptiv sein und Phenomäne behandeln können, wie das Schneiden von geometrischen Körpern und Interaktionen zwischen Strukturen und Strömungen (*flow-structure interactions*). Diese Dissertation konzentriert sich auf die Entwicklung und Implementierung von Partikelmethoden um diese Systeme zu behandeln. Wir unterscheiden dabei drei Hauptaspekte in den dafür entwickelten Methoden:

1. die Entwicklung der neuartigen *Lagrangian Particle Level Set*-Methode für die Beschreibung der Deformation von komplexen Oberflächen,
2. die Entwicklung von Partikelmethoden für die Simulation von Fluiden and elastischen Materialen basierend auf der *Smooth Particle Hydrodynamics*-Methode,
3. die Entwicklung einer *Immersed Boundary*-Methode für die Simulation von deformierenden komplexen Körpern umgeben von Fluiden.

Die entwickelten Methoden werden anhand von Vergleichtests validiert und an Beispielsystemen relevant für die virtuelle Chirugie getestet. Die Simulation von grossen anspruchsvollen Systemen aus der Kontinuumsmechanik ermöglicht dabei eine neu entwickelte parallele Partikel-Gitter Software-Bibliothek.

To My Family

Acknowledgements

I would like to thank all the people who accompanied me during my PhD studies and contributed to my PhD thesis in many different aspects to make it a great experience.

First of all, I want to thank Petros Koumoutsakos for being my PhD advisor and linking this project to the NCCR Computer Aided and Image Guided Medical Interventions (Co-Me). He is a very creative person and accompanied me during my PhD time with an amazing amount of ideas. He gave me the opportunity to work in a highly dynamic and stimulating environment. He promoted my participation in the *Summer School in Multiscale Modeling and Simulation* and my research visit at the University of Tokyo, Japan. These activities were very inspiring for me, not only from a scientific point of view.

I also want to thank my coreferees Anthony Leonhard (Caltech, Pasadena, USA) and Jens Walther (Technical University of Denmark, Lyngby, Denmark) for their visits at ETH Zurich leading to many helpful discussions.

I wish to thank my colleagues in the CSE Lab and the CoLab for their friendship, support and source of inspiration. Especially, I want to acknowledge Ivo Sbalzarini, Jens Walther, Michael Bergdorf and Philippe Chatelain for being part of the PPM Library-Team.

Moreover, I'm grateful for the Co-Me Network that gave me the opportunity to widen my horizon and to network with different researchers. I enjoyed the collaboration and discussions with the group of Prof. Mazza, namely Alessandro Nava and Davide Valtorta, at the IMES, ETHZ, working on the mechanical characterization of soft biological tissues. Prof. Matthias Teschner (University of Freiburg, Germany), Raimundo Sierra (Harvard Medical School, Boston, USA) and Matthias Harders (BIWI, ETHZ) were great discussion partners about the development of *surgery simulators*. I also want to thank

Prof. Niederer (IBT, ETHZ) for being the leader of the Soft Tissue Modeling group. Special thanks go to Frank Langlotz (MEM Center, Bern), Mireille Reef (BIWI, ETHZ) and Ulrich Spaelter (EPFL) for their help in the organization of the *CoMe Workshop 2004* resulting in a great success. My appreciation goes to the Co-Me project office, namely Vreni Vogt and Bert Müller.

Several ETH students contributed to this project in 7 different semester projects. They are in order of appearance: Sidclei da Silva, Claudio Christen, Hansjörg Sidler, Andreas Ess, Ivan Guanjana, Igor Beati, Bettina Polasek.

I also want to thank my family. I'm grateful to my parents for their understanding and their support for higher education and to my brother's family for showing the diversity of life outside the academic world.

Last, but not least, I wish to thank my husband Thomas cordially for accompanying me in the ups and downs during the PhD. With his motivating character, he was a great partner to enjoy the ups and to overcome the downs. His love is a significant part of my life and, therefore, also part of this thesis.

Contents

List of Acronyms

1D	One dimension
2D	Two dimensions
3D	Three dimensions
AMR	Adaptive Mesh Refinement
ALE	Abitrary Lagrange Euler
CF	Color Function
CFL	Courant-Friedrichs-Lewy
CPU	Central Processing Unit
DPD	Dissipative Particle Dynamics
ER	Endoplasmatic Reticulum
FEM	Finite Element Method
FFT	Fast Fourier Transform
FMM	Fast Multipole Method
GFlop	Giga Floating Point Operations per Second
GHz	Giga Hertz
IVP	Initial Value Problem
I/O	Input/Output
pIBM	Particle Immersed Boundary Method
LS	Level Set
LSM	Level Set Method
MD	Molecular Dynamics
MG	Multi-Grid
MPI	Message Passing Interface

ODE	Ordinary Differential Equation
PDE	Partial Differential Equation
PM	Particle-Mesh
PP	Particle-Particle
P^3M	Particle-Particle Particle-Mesh
PPM	Parallel Particle-Mesh Library
PSE	Particle Strength Exchange
ROB	Recursive Orthogonal Bisection
SAR	Stop-At-Rise
SDF	Signed Distance Function
SPH	Smooth Particle Hydrodynamics
rSPH	Remeshed Smooth Particle Hydrodynamics
VM	Vortex Method
VOF	Volume of Fluid
WENO	Weighted Essentially Non-Oscillatory

List of Symbols

The following list explains the equation symbols used in this thesis. Naming conflicts are unavoidable, but the proper meaning of symbol is always unambiguously determined by the context. Vector and tensor quantities are printed in bold face.

Latin Characters

c_0	Speed of sound
e	Base of natural logarithm, parallel efficiency
f	General function
h	Inter-particle spacing, grid spacing
i, j	Indices
k	Constant
n_d	Normalization constant in dimension d
$\mathbf{n}$	Surface normal
p	Particle, pressure
r	Radius, order
r_c	Cutoff radius
t	Time
$\mathbf{u} = (u, v, w)$	Velocity
$\mathbf{x} = (x, y, z)$	Position
$\mathbf{X}$	Reference position
v	Volume
$\mathbf{B}$	Left Cauchy Green strain tensor

C_d	Drag coefficient
C_L	Lift coefficient
C_{n0}	Polynomial coefficient
D	Volume coefficient
D^β	Differential operator
E	Young's modulus
$\mathbf{F}$	Deformation gradient
G	Shear modulus
H	Measurement of distortion
$\mathbf{I}$	Identity matrix
I_1	First invariant of the Left Cauchy Green strain tensor
J	Volume change, determinant of the Jacobian
L_2	L two Euclidian norm
L_∞	L infinity norm
M	Mach number
N	Number of particles
$\mathcal{O}(\cdot)$	on the order of
R	Specific gas constant
Re	Reynolds number
St	Strouhal number
S	Parallel speed up
$\mathbf{S}$	Deviatoric stress tensor
T	Temperature, time period
U	Strain-energy function
U_0	Characteristic velocity, reference velocity, swimming velocity
$\mathbf{W}$	Interpolation kernel

Greek Characters

$\alpha = (\alpha_1, \ldots, \alpha_d)$	Multiindex in dimension d
$\beta = (\beta_1, \ldots, \beta_d)$	Multiindex in dimension d
γ	Ration of specific heats
ϵ	Characteristic length
ϵ	Strain tensor
$\delta(x)$	Dirac delta function
δ_{ij}	Kronecker delta symbol
Δ	Difference of information
Δt	Time step
κ	Curvature
λ, μ_s	Lame constants
μ	Viscosity
μ_s	Shear modulus
ν	Poisson ratio, kinematic viscosity
ρ	Density
ρ_0	Characteristic or reference density
σ	Cauchy stress tensor
θ	Angle
τ	Shear stress tensor
ξ	Reference position
η, ζ	Mollifier kernel functions
ω	Vorticity
Γ	Circulation, interface
Φ	Level set function
χ	Heaviside function
Ω	Rotation rate, domain

Subscripts and Superscripts

$1, 2, 3$	Cartesian coordinate direction or components
app	Apparent
eff	Effective
ϵ	Mollified to size ϵ
h	Discretized with resolution h
i, j, k	Cartesian coordinate direction or components (Einstein's summation convention)
max	Largest value of the variable
min	Smallest value of the variable
p	At particle position
x, y, z	The component in the specified direction
∞	Asymptotically
$\bar{\cdot}$	Normalized

Special Symbols

∇	Nabla operator, gradient operator
$\frac{\partial}{\partial}$	Partial derivative
∞	Infinity
$\langle \diamond \rangle_p$	Particle approximation of $\diamond$ at the position of particle p
∞	Infinity
$[\cdot, \cdot]$	Closed interval
$(\cdot, \cdot)$	Open interval
D/Dt	Lagrangian derivative with respect to time

Introduction

The development of novel computational methods that are capable of harnessing the increase in available computing power allow today the tackling of complex scientific and engineering problems. At the same time we observe a trend in applying advanced computational concepts that have been developed for areas such as fluid dynamics to emerging fields such as biology and medicine where advanced computing is a promising tool for quantitative prediction and understanding. This thesis exemplifies this approach in the problem of virtual surgery.

One of the core challenges in virtual surgery is the modeling and simulation of soft biological tissue. This requires not only reliable biomechanical models but also a robust and highly flexible simulation tool as the underlying physical laws and geometries are complex. In biomechanical models we often have to consider a nonlinear elasticity law to describe the large deformations that can appear during a medical intervention. Complex physical phenomena, such as interaction with medical devices, need to be taken into account. As the human body consist mostly of water and many vitals are filled with fluid, biological tissue is often in contact with body fluids. The consideration of such fluid-solid interactions, however, is numerically demanding and requires special numerical techniques. Large-scale simulations are required to resolve the complex geometries of biological structure. The cutting of tissue can cause large topological changes leading to a numerical challenge especially for grid-based methods where this requires the creation of new cells and an update of the connectivity information.

We investigate the use of particle methods as a simulation tool for virtual surgery and consider the performance in key requirements

1. Robustness and flexibility with respect to physical laws and geometries
2. Consideration of fluid-solid interactions
3. Behavior with respect of large deformation and topological changes
4. Scalability of large scale simulation on parallel computer architecture.

This thesis reviews and extends computational methods to simulate biological systems for the use in virtual surgery.

The thesis is structured as follows:

Chapter 1. Motivation and Objectives

The motivation of this thesis is the need to improve the efficiency and accuracy of virtual surgery simulations.The focus of virtual surgery is on the development, integration and validation of enabling technologies towards advanced computer aided and image guided systems for medical interventions. It supports the complete treatment process from diagnosis, therapeutic planning and simulation via intra-operative action to postoperative care, monitoring and documentation. In this framework, virtual reality based surgical simulation is an area of special interest. Surgical simulators can only fulfill their mission if they provide a realistic and configurable training environment. *How realistic are virtual surgery nowadays?*

A training environment requires that the simulated organs behave authentically in a biomechanically and physiologically perspective and the environment is represented in a realistic manner. However, due to the high complexity of the simulated system, none of the simulators proposed up to now can even approximately achieve the necessary level of realism in simulation and visualization. Chapter 1 illustrates the state of the art of surgical simulators and shows benefits and risks of the virtual reality training concept. In

Section 1.5, we presents the key components of a surgical simulator and their influence on the training effect.

Chapter 2. Particle Methods

Particle methods have been successfully applied in a wide range of problems, from astrophysics [84, 107], to computational fluid dynamics and [41, 109] and molecular dynamics [74], but they are hardly used for the simulation of soft biological tissue. *Why do we consider particle methods for the use in virtual surgery?*

The advantages of particle methods include adaptivity and multi-resolution capabilities of the computational elements, good stability properties of the discretization, and, similar to discrete systems, an inherent link of the computational elements to the physics that they represent. The flexible handling of geometries and physics makes particle methods an appealing technique for the numerical challenges in virtual surgery simulations. Chapter 2 includes the fundamental background of continuum particle methods as they are used in this study. It illustrates basic numerical techniques, such as derivative approximations (Section 2.3) and the redistribution of particles (Section 2.4).

Chapter 3. Particle Level Set Method

To describe the evolution of interfaces/surface, we distinguish two classes of computational methods: interface capturing and interface tracking methods. Particle methods are often associated with tracking methods, because interfaces evolution in tracking methods is solved in a Lagrangian fashion using markers. In level set methods, however the interface is captured using an implicit function that are traditionally evolved in an Eulerian way. *How can we combine the level set method with Lagrangian particle methods?*

We can naturally solve the underlying level set equation in a Lagrangian frame using particles that carry the level set information as also shown in Section 3.3. The consistent remeshing procedure enforces the regularization of the particle locations when the particle map gets distorted by the advection field. The Lagrangian description of the level set method is inherently adaptive and exact in the case of solid body motions. The efficiency and accuracy of the method is demonstrated in several benchmark problems in two and three dimensions involving pure advection and curvature induced motion of the interface (Section 3.5). Section 3.6 and 3.7 show that the simplicity of the particle description is well suited for real time simulations of surfaces involving cutting and reconnection as in virtual surgery environments.

Chapter 4. Particle Simulations of Fluids

The particle method used in this study bases on remeshed Smoothed Particle Hydrodynamics (rSPH), a continuous particle method to simulate compressible viscous fluids described by the Navier-Stokes equations. *How can continuous particle methods solve the Navier-Stokes equations?*

This chapter shows the governing equations of a compressible viscous fluid and its particle discretization. Moreover, it includes results of a compressible vortex ring simulation and a general discussion on Smooth Particle Hydrodynamics.

Chapter 5. Particle Simulations of Elastic Solids

The correct reproduction of the deformation behavior is a crucial part of any surgical simulator. A correct reproduction requires first of all a reliable model for the stress-strain relationship with suitable parameters. Linear elastic models are suitable up to a stretch of $\lambda = 10\%$. When the deformation exceeds a stretch of 10%, nonlinear effects need to be taken into account. Finally, an accurate numerical solver of the model should deliver the desired simulation results. An established numerical method to solve elasticity problems is the grid-based finite element method that solves the nonlinear elasticity models in the reference coordinate system. *What kind of deformations can we solve with particle methods?*

In Chapter 5, we consider a remeshed Lagrangian particle method to simulate an elastic solid undergoing large deformations. The particle solver can handle elastic solids described by linear and nonlinear constitutive models. In the nonlinear case, we are able to omit the Eulerian description of the deformation gradient by considering its Lagrangian evolution in the governing equations (Section 5.2). Therefore, we present for the first time a Lagrangian particle simulation of elastic solids described by a nonlinear model. The efficiency and accuracy of the method is demonstrated in Section 5.4 and 5.5, showing several benchmark problems in two and three dimensions involving large deformations and comparisons to finite element solutions. The results show similar accuracy as the finite element solution. The finite element solver reveals nonphysical numerical artifacts in the plane strain compression test at large deformations whereas the particle solution remains plausible. The particle solver for nonlinear elastic material is proven to be well suited to simulate liver tissue as shown in Section 5.6.

Chapter 6. Parallelization of Particle Simulations

Particle simulations formally represent a N-body problem with a computational cost that can increase with the square of the number of particles. Although several techniques exist to reduce the computational cost, their wide-spread use is hindered by several computational challenges in the parallelization of these methods. *What kind of concepts are available to reduce the computational cost of particle methods and how do they perform on large scale problems?*

The highly efficient Parallel Particle Mesh (PPM) library solves the key parallelization issues involving particle-mesh interpolations and the balancing of processor particle loading and adaptive trees. Section 6.1 to 6.4 give an introduction to the main issue of parallelizing particle-mesh computations. The computation of particle-particle interactions are addressed in Section 6.5, particle-mesh interpolations in Section 6.6. Section 6.7 illustrates the parallel fast multipole method for handling long-range interactions. Section 6.8 presents the fast fourier transformation to solve the Poisson equation on a mesh representation. The efficiency of the Parallel Particle Mesh (PPM)library is demonstrated in a series of benchmark tests on distributed memory and on a shared-memory vector architecture. Section 6.12.1 includes the benchmark test for the parallel fast multipole method. The poisson solvers (Section 6.12.3) are tested on a three-dimensional test problem using up to 128 processors. The use of the library is shown by the simulation of compressible vortex rings using a novel formulation of smooth particle hydrodynamics (Section 6.12.4).

Chapter 7. Particle Immersed Boundary Method

Flow-structure interactions are involved in several processes of living organisms. The geometrical structures in these processes are very complex presenting a number of a numerical challenges for fluid solvers. *How can we construct a particle method that can handle complex geometries in a fluid environment?*

In Chapter 7, we consider the novel particle Immersed Boundary Method to enforce no-slip boundary conditions on complex geometries. A forcing term appearing in the momentum equation is evaluated on the boundary points such that the no-slip boundary condition is fulfilled on the boundary. The method applied to an isothermal compressible fluid is capable of approximating the flow of incompressible medium at a Mach Number of 0.05. The efficiency and accuracy of the method is demonstrated in several benchmark problems in two and three dimensions involving flow past a cylinder/sphere (Section 7.3). The particle Immersed Boundary Method is shown to be well suited for the simulations of anguilliform swimming (Section 7.4).

Chapter 8. Fluid-Solid Interactions

A fluid-solid system in one-dimension can be described by a unified formulation of the Burger's and the wave equation coupled by a force equilibrium at the fluid-solid interface as presented by Cottet [39]. Cottet solved the unified formulation accurately using a 1D particle solver [39]. *How flexible are particle methods in solving fluid-solid interactions?*

In Chapter 8, we consider a remeshed Smoothed Particle Hydrodynamics (rSPH) solution of this fluid-solid problem. The anisotropic diffusion terms appearing at the interface in the unified formulation are solved numerically either based on the method of particle strength exchange (PSE) or by isotropic one-sided differentiation. In Section 8.3, the rSPH method is compared with an Arbitrary Lagrange Euler (ALE) method and the hybrid finite-difference particle formulation of Cottet [39] in one dimension. The results show that both approaches are more accurate and more robust that the ALE solution but less accurate that the result of Cottet due to implementation details. Both rSPH approaches converge linearly with the anisotropic diffusion approach having a lower absolute error.

Moreover, we extend the model to the compressible Navier-Stokes equation in one and two dimensions in Section 8.4 and 8.5, respectively. Our mathematical model involves the Navier-Stokes equation of compressible viscous fluid and the constitutive law of a linear elastic solid. The governing equations coupling the fluid and the solid model are derived from first principles of continuum mechanics. To demonstrate the performance of the particle simulation we consider various cases testing the behavior of the fluid-solid interface exposed to normal and shear stresses. The particle solution matches well with the analytic solution when available.

Chapter 1

Motivation and Objectives

1.1 Introduction

This thesis is motivated by the need to advance the efficiency and accuracy of virtual surgery environments and, in particular, to increase the realistic aspect of virtual surgery. An effort is being made to increase the accuracy of physically based modeling of flow-structure interactions: a key aspect of many virtual surgery environments.

The effective simulation of flow-structure interactions is a problem inherent to several physical problems ranging from hydrodynamics to material processing. It entails a number of challenging computational problems such as interface modeling, efficient flow solvers and modeling of solids under large deformations.

In this thesis we address several of these computational challenges and contribute to the modeling of flow-structure interactions such that it may be employed in several disciplines.

1.2 Virtual Surgery Simulation

Virtual surgery simulation combines two different concepts, virtual reality and simulation, to form a new paradigm of training system.

Simulation is defined as the technique of representing the real world by a computer program with the restriction that it should imitate the internal processes and not merely the results of the thing being simulated [183]. The term simulator denotes in a technical context a machine that simulates an environment for the purpose of training or research

[183]. Simulation can serve a wide range of purposes. It can lead a better understanding of complex systems, it can be used as a training tool, or it can help to predict the future behavior of a system.

Virtual reality describes a hypothetical three-dimensional visual world created by a computer. User wears special goggles and fiber optic gloves etc., and can enter and move about in this world and interact with objects as if inside [183]. Many applications of virtual reality have been proposed in the medical field, i.e. the visualization of different organs as three dimensional objects for teaching purposes or the reconstruction of patient specific anatomy from radiological data for diagnosis [161].

Figure 1.1: Typical setups in minimally invasive surgery

Simulator systems have been developed for the planning of surgical procedures, e.g. to predict the outcome of a medical intervention. This is of special interest in facial surgery, where the main goal is a modification of the facial tissue to correct abnormalities [190, 166]. Other examples are the planning of radiation doses for cancer therapy [17] or the planning of the placement of dental implants [173]. This category of simulators is patient specific and faces several challenges. The accuracy of the underlying models and methods are crucial in surgical planning while on the contrary, the real time capabilities are important for a training device.

According to Liu *et al.* [102], surgical training simulators can be classified into needle-based, open surgery, and minimally invasive systems (Fig. 1.1). An example for the first type of a surgery simulator is the CathSim Vascular access simulator developed by Immersion Corporation for the training of the intravenous catheter insertion [133]. Open surgery is considerably more difficult to simulate as the interaction between the surgeon and the patient is much more flexible. The field of view, the range of motion and the tactile feedback are remarkably larger for this type of interventions. So far, the simulation has largely been limited to the suturing of open wounds [179, 20].

1.3 Virtual Reality Based Training

Virtual training devices are well established in the instruction of pilots in form of flight simulators. The analogy to the aviation industry has been pointed out early in the development of surgical training simulators [139] and the success story of flight simulators has been a key motivation to build simulators for the medical field. The research on surgical simulators of the past decade has revealed many challenges for a successful implementation of surgical training simulators. The human body is a highly complex system that cannot be completely modeled. Several surgical instruments with several degrees of freedom interact with the objects that can be deformed and fragmented. Moreocer, there is a wide range of user actions leading to reactions of the simulator, e.g. bleeding, that need to be taken into account.

At the beginning of the simulator development for minimally invasive surgery, the expectation was that a satisfactory surgery simulator would be available before the year 2000 [86]. While there have been many efforts for the generation of a wide range of different simulators systems, this expectation has not yet been met. The numerous research efforts in the field are still not sufficient to cope with the many challenges encountered during the last decade. Surgical simulators have been investigated in diagnostic endoscopy investigations [174], laparoscopic surgical procedures [10, 13, 100], hysteroscopy [76] (Fig. 1.2), arthroscopic interventions [191], eye surgery [138, 142] and radiological procedures [73].

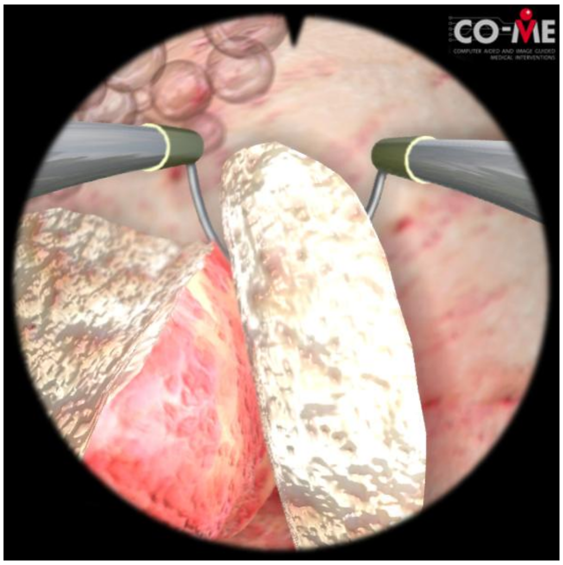


Figure 1.2: Snapshot of a virtual surgery simulation using a hysteroscopy simulator [76]

1.4 Benefits and Risks

Virtual reality based training offers several benefits compared to other training methods, such as training sessions on human cadavers or animals, but it also entails risks. In this section we discuss the main strength and weaknesses of this kind of training system. Virtual reality does not involve real patients or animals, offering a risk-free environment for training. In general, patients prefer to be treated by an experienced surgeon and will show some discomfort and resistance for the participation in medical interventions for training purpose. The computer based simulation enables the tracking of all actions of the trainee, thus providing the basis for an objective assessment. So far, performance measurement is limited to the final result of the intervention and the subjective judgement by the supervising expert [102]. Computer based training devices, however, are capable of objective assessment. Therefore, virtual reality based surgical training simulators could be used for the recertification of practicing medical doctors [12]. The simulators can provide additional training where deficiencies in surgery practice are evident. Currently, training is depending on the actual demand of interventions and the availability of an appropriate case. This training-by-opportunity does not allow scheduled teaching sessions because the cases of demand are unpredictable. In contrary, surgical simulators can be available at all times and places. Therefore, training can be performed during normal working hours reducing the costs and increasing the comfort for both teacher and students. Nowadays, any medical student will experience a limited number of cases during his training period. A virtual reality based simulator can offer a large number of different anatomies and pathologies that can be handled in a small period of time. Training can be repeated to test different procedures and work flows. Thus, the simulator offers surgeons the opportunity to improve his surgical skills. A high fidelity simulator can offer scenes of high complexity and introduce a number of complications during the intervention, in case of hysteroscopy a perforation, loss of pressure, unexpected blood flow etc. Such a problem-based surgical simulation may improve patient safety not only by improving the surgeons skills, but also by teaching surgeons to avoid complications during the surgery and to manage them. It is expected that virtual reality based training can reduce the costs for the training of surgeons.

On the other hand, surgical simulators are also associated with some risks. Several risks are inherent to simulator based teaching [176]. Most dangerous is the risk of wrong teaching. Simulator fidelity may always be imperfect independent from the quality level of underlying model. The simulator system may not perfectly reproduce the tasks performed during actual interventions. Students may acquire a wrong impression of the procedure and adopt an inappropriate behavior or develop a false sense of security in their skills which potentially can harm the patient. Proficiency on a simulator does not ensure proficiency in clinical settings. Therefore, it is crucial to validate the transfer of skills from the simulator to the operating room. The system should not pretend to be more realistic than it actually is. The trainee has to be aware to which extend the simulator imitates reality.

1.5 Key Components

1.5.1 Anatomical Models

Anatomical models are important for the quality of a surgical simulator by offering variable training scenarios. Similar to flight simulation, where different weather conditions, airports and system malfunctions can be defined, surgical simulations also need the same range of configurable training conditions. When a user repeats the training with the same single organ model he will adapt to this special anatomy. Therefore, the training scene should different from session to session preferably as in reality. The simulator system has to be able to generate anatomical models considering the natural variability and to integrate a wide range of different pathologies.

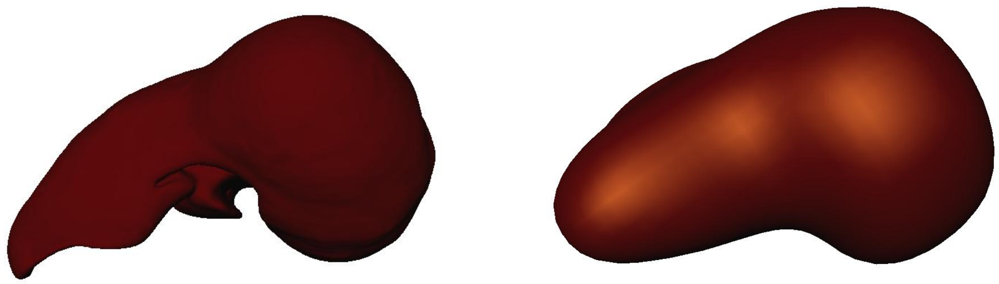


Figure 1.3: Two anatomical models of the human liver: Triangular surface mesh segmented from the Visual Human Dataset [114] (left) and its reconstruction based on a particle representation (≈ 400 particles) [82]

The most common source for the generation of anatomical model [22, 185] is the data set Visible Human Project [114]. In general, the organs of interest are reconstructed from either the female or male anatomy. These data sets provide a high resolution and good image quality, so that the resulting models can be segmented with a high accuracy. Fig. 1.3 shows a liver topology segmented from the Visible Human Project and its reconstruction based on a particle representation. Only two data sets are available of the Visible

Human project, consequently limiting the number of scenes. Alternatively, Computer Tomography (CT) and Magnetic Resonance (MRI) images have been widely used for the generation of anatomical models [126, 191]. As Computer Tomography imaging exposes the subjects to radiation, experts only apply it on diseased patient. Magnetic Resonance imaging is the technique of choice when healthy volunteers are scanned to provide basic anatomical data. Both Computer Tomography and Magnetic Resonance Imaging belong nowadays to standard equipment of most hospitals. They can be used to provide the anatomy of a patient for the training of a patient specific case. The raw data of the scans have to be segmented to provide an anatomical model in form, for example, of a triangular surface. Segmentation methods can be classified according to their degree of automation into either manual, semi-automatic or fully automatic.

1.5.2 Physical Modeling and Simulation

The deformation behavior of the tissues is crucial for a correct perception of the scene both visually and haptically. A simple example is the tactile investigation of tumors with the endoscopic instruments. The surgeon will evaluate both the visual deformation and the force resistance and choose the further proceeding based on this diagnosis. Therefore, the biomechanical properties of the tissues are of special interest to achieve a suitable representation of these effects. The determination of a constitutive model and the evaluation of the mechanical parameters are associated with several problems, because the behavior of tissue depends heavily on its components (muscular tissue, cartilage), stress state (relaxed, isotonic), and vitality (in-vivo, ex-vivo) [104]. Tissues are inhomogeneous, anisotropic, and viscoelastic materials and their properties vary with age, sex, and genre [102]. Three different principles have been proposed for the estimation of tissue parameters without introducing damage, namely indentation [28, 121], aspiration [116], and torsion experiments [171]. Most researcher perform ex-vivo experiments in a controlled environment that has limited relevance. Only a few number of in-vivo experiments have been reported so far, including the study of Mazza *et al.* [104].

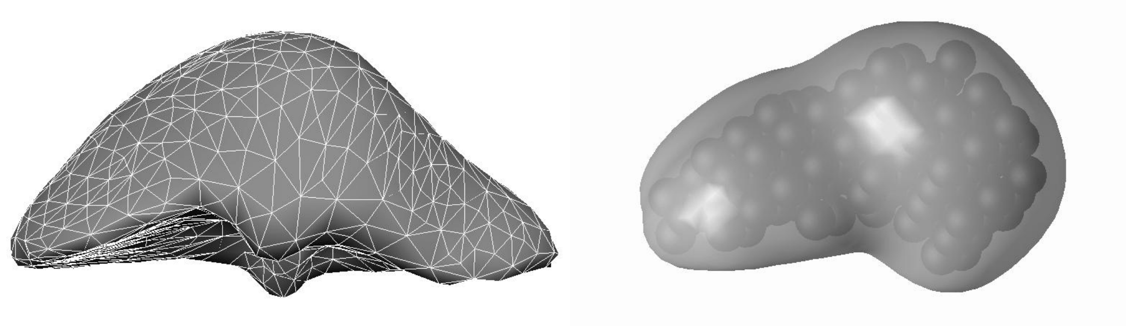


Figure 1.4: Two different liver representation. A finite element model(left) and a particle system (right)

Several numerical methods have been investigated for the simulation of the mechanical behavior of tissue models: Mass spring systems, finite element method and, in this thesis, a particle approach (Fig. 1.4). Mass spring models has been a common technique to represent deformable models for real time simulations [110, 170, 23, 168]. Mass spring models are commonly described by a grid-like structure where the nodes represent mass points and the edges behave like physical springs. They are relatively simple to implement and grids with a large number of nodes can be updated at high rates. The main problem for realistic simulation based on mass spring system is the determination of the correct parameterization of the grid. The choice of the grid topology, the spring constants, and damping coefficients is unclear, and, therefore, the simulation results often in an incorrect imitation of the tissue behavior.

Finite Element methods (FEM) [132] are commonly used for simulations of deformable bodies in continuum mechanics. FEM solvers are accurate numerical solvers where the continuum is discretized by finite elements residing on a grid. Due to their computational cost ordinary finite elements methods are not used often for realtime application. With the restriction to linear elasticity it is possible to achieve real-time performance by using a limited number of nodes. Basdogan *et al.* [11] for example use linear and isotropic elements to achieve the update rates required by the haptic interface. Deformations can be computed solely based on surface elements without the need to compute the deformations of inner elements [22] . An adaptive method has been presented by Tendick *et al.* [185] and Debunne *et al.* [48] where computational elements accumulate in areas of stronger deformations. The consideration of tissue cutting, however, requires a significant effort in modeling and computation for finite element methods due to its purely grid based approach (cf. Section 1.5.3). Moreover, the handling of fluid-solid interactions using FEM is still unclear. This aspect is important for physiological phenomena such as bleeding after the cutting of vessels. To correctly visualize the resulting blood flow.

Particle methods (cf. Chapter 2) aim to become an alternative approach bridging the gap between the efficient but rather heuristic mass-spring models with the accurate, but computationally expensive finite-element methods.

1.5.3 Collision Handling

The handling of interactions between the virtual manipulators, controlled by the user, and the virtual anatomy is dominated by collision handling. Collision handling involves two steps, namely collision detection and collision response. First, collisions between the different objects in the virtual scene, such as endoscopic instruments and organs, have to be detected. Once a collision has been identified, the collision response may be handled, i.e. the correct reaction has to be computed. For example, a needle will penetrate the biological tissue and a scalpel will cut it. Depending on the location of interaction the tissue can be deformed or disrupted or starts to bleed. From numerous kinds of collision handling, the process of cutting tissue appears to be the most complex operation challenging the real time capabilities of most computer based simulations.

Collision detection is commonly achieved by a two level approach. First, a fast evaluation of regions with potential collisions is performed using collision bounding [37]. In a second step, the exact contact positions are evaluated within the identified regions by a collision refinement. Collision bounding requires a pre-processing step to partition the single objects into elementary primitives such as spheres, axis-aligned bounding boxes (AABBs) [37] or oriented bounding boxes (OBBs) [66]. The latter are aligned with the eigenvectors of the objects and represent therefore the smallest Cartesian volume of the object. The respective primitives are then used to bound the potential collisions. To fur-

Figure 1.5: Snapshot of a virtual cutting simulation. Cutting is one of the challenging problems in virtual surgery [21]

ther enhance the computational speed, hierarchies of bounding tests have been presented [66] . The use of such techniques is limited to rigid objects. In case of deformable objects, the data structure holding the primitives has to be re-evaluated which is a time consuming operation in an average application. Heidelberger *et al.* [78] presented a real time method for the evaluation of the intersections based on Layered Depth Images (LDI).

The handling of collisions depends strongly on the interacting objects. Simple contacts lead to deformations, others to the dissection of the tissue. Bielser *et al.* [21] has presented a method for cutting of tissue represented by tetrahedral meshes in real time (cf. Fig. 1.5). Arbitrary intersections of the tool with the mesh are evaluated by a state machine which tracks the topology of every tetrahedra and controls the progressive subdivision of the elements. However, the reliable and realistic simulation of tissue dissections in real-time is not yet fully solved due to the complexity of the problem. The cutting of tissue represents the most complex operation challenging the real time capabilities of most computer based simulations. For a realistic impression, cuts should be allowed along arbitrary trajectories. This causes discontinuities in the grid representations of organs which are computationally expensive to handle.

Chapter 2

Particle Methods

2.1 Introduction

A large number of physical problems can be modeled using particle-based methods. Particle descriptions can be used for the simulation of continuum systems as in the case of discrete fluid or solid elements in Smooth Particle Hydrodynamics (SPH) [109] and vorticity-carrying fluid elements in Vortex Methods (VM) [41, 75, 36] and inherently discrete systems as in gravitational particles for astrophysics [84], Dissipative Particle Dynamics (DPD) [58] for mesoscale polymer descriptions, atomistic Molecular Dynamics (MD) simulations [74], and charged particles in plasma physics [84].

Continuum particle methods are based on the approximation of smooth functions by integrals that are being discretized onto computational elements called particles. A particle p resides at a position $\mathbf{x}_p$ and carries a physical quantity Φ_p. These particle attributes evolve to satisfy the underlying governing equation in a Lagrangian frame of reference [97]. In pure particle methods, the governing equations in form of Partial Differential Equations (PDE) are transformed into sets of Ordinary Differential Equations (ODE) by replacing spacial derivatives of quantities by an approximation based on the particle quantities. The particle approximation involves a superposition over all particle. The dynamics of the particles are governed by sets of ODEs that determine the trajectories of the particles p and the evolution of their properties Φ, thus:

$$\frac{d\mathbf{x}_p}{dt} = u(\mathbf{x}_p, t) = \sum_{q=1}^{N} \mathbf{U}(\mathbf{x}_p, \mathbf{x}_q; \Phi_p, \Phi_q), \quad p = 1, \ldots, N \tag{2.1}$$

$$\frac{d\Phi_p}{dt} = \sum_{q=1}^{N} \mathbf{F}(\mathbf{x}_p, \mathbf{x}_q; \Phi_p, \Phi_q), \quad p = 1, \ldots, N, \tag{2.2}$$

where u is the velocity field. The dynamics of the simulated system are completely defined by the functions $\mathbf{U}$ and $\mathbf{F}$ that represent the physics of the problem. In pure particle methods, K and F emerge from the integral approximations of differential operators;

If the functions $\mathbf{U}$ and $\mathbf{F}$ are local, the algorithmic complexity of the sums in Eq.(2.1) and (2.2) is $\mathcal{O}(N)$. For long-range interactions the cost is $\mathcal{O}(N^2)$, but fast algorithms such as multipole expansions [70] are available to reduce the complexity to $\mathcal{O}(N)$ also in these cases.

The issue of efficient parallel implementation of particle methods is addressed in Chapter 6.

2.2 Function Approximations

A smooth approximation of the field function can be constructed by using a mollification kernel $\zeta_\epsilon(\mathbf{x})$:

$$\Phi_\epsilon(\mathbf{x}) = \Phi \star \zeta_\epsilon = \int \Phi(\mathbf{y})\, \zeta_\epsilon(\mathbf{x} - \mathbf{y})\, d\mathbf{y} \tag{2.3}$$

where ϵ denotes a characteristic length of the kernel.

The kernel is approximating the Dirac delta function, i.e. $\delta(\mathbf{x}) = \lim_{\epsilon \to 0} \zeta_\epsilon(\mathbf{x})$, and is said to be of order r when the following moment conditions [41] are satisfied:

$$\int \zeta_\epsilon(\mathbf{x}) d\mathbf{x} = 1, \tag{2.4}$$

$$\int \mathbf{x}^i \zeta_\epsilon(\mathbf{x}) d\mathbf{x} = 0 \quad \text{if } |i| \leq r - 1, \tag{2.5}$$

$$\int |\mathbf{x}|^r \zeta_\epsilon(\mathbf{x}) d\mathbf{x} \leq \infty \tag{2.6}$$

Monaghan [106] presented a systematic way of constructing kernels $\zeta_\epsilon(\mathbf{x})$. The mollified approximation $\Phi_\epsilon(\mathbf{x})$ can be discretised using the particle locations as quadrature points and a particle approximation of the regularized field function is

$$\Phi_\epsilon^h(\mathbf{x}) \;=\; \Phi^h \star \zeta_\epsilon \;=\; \sum_{p=1}^{N} v_p \Phi_p \zeta_\epsilon(\mathbf{x}-\mathbf{x}_p) \tag{2.7}$$

where $\mathbf{x}_p$, and v_p denote the position and volume of the p-th particle, and $\Phi_p = \Phi(\mathbf{x}_p)$ the field value at the $p = 1, \ldots, N$ particle locations.

As discussed in [41] the error introduced by the quadrature of the mollified approximation of Φ can be distinguished in two parts as

$$\Phi \;-\; \Phi_\epsilon^h \;=\; (\Phi \;-\; \Phi \star \zeta_\epsilon) \;+\; (\Phi \;-\; \Phi^h) \star \zeta_\epsilon \tag{2.8}$$

The first term in Eq. (2.8) denotes the mollification error that can be controlled by appropriately selecting the kernel properties. The second term denotes the quadrature error due to the approximation of the integral on the particle locations. The overall accuracy of the method [41] results in

$$\|\Phi - \Phi_\epsilon^h\|_{0,p} \;\leq\; \|\Phi - \Phi_\epsilon\|_{0,p} \;+\; \|\Phi_\epsilon - \Phi_\epsilon^h\|_{0,p} \sim\; \mathcal{O}(\epsilon^r) \;+\; \mathcal{O}\left(\frac{h^m}{\epsilon^m}\right) \tag{2.9}$$

where $\|(.)\|_{0,p} = \left(\int (.)^p \, d\mathbf{x}\right)^{1/p}$, m is a large enough fixed number and r denotes the order of the first non-vanshing moment of the kernel ζ_ϵ [41]. For equidistant particle locations we obtain $m = \infty$ and for positive kernels such as the Gaussian, $r = 2$.

In this study we choose ζ_ϵ to be a quartic spline kernel with second order of accuracy [29] (cf. Fig. 2.1) is implemented:

$$\zeta_\epsilon(\mathbf{x}) = n_d \bar{\zeta}_\epsilon = n_d \begin{cases} \frac{s^4}{4} - \frac{5s^2}{8} + \frac{115}{192} & 0 \leq s < \frac{1}{2}, \qquad s = \frac{|\mathbf{x}|}{\epsilon}, \\ -\frac{s^4}{6} + \frac{5s^3}{6} - \frac{5s^2}{4} + \frac{5s}{24} + \frac{55}{96} & \frac{1}{2} \leq s < \frac{3}{2}, \\ \frac{(2.5-s)^4}{24} & \frac{3}{2} \leq s < \frac{5}{2}, \\ 0 & s \geq \frac{5}{2}. \end{cases} \tag{2.10}$$

The normalization value n_d depends on the dimension of the problem and is computed as:

$$n_d = \frac{1}{\sum_j v_j \bar{\zeta}_\epsilon(\mathbf{x}-\mathbf{x}_j)} \tag{2.11}$$

ensuring the property of *partition of unity* for the discrete superposition over the particles. The normalization value n_d serves as a correction factor to reduce the error resulting from the integral approximation that leads from Eq.(2.3) to Eq.(2.7). When performing analytical operations, such as differentiations, to the kernel $\zeta_\epsilon(\mathbf{x})$, the normalization value is neglected. Note that the moment conditions expressed by the integrals of the mollifier functions are not often well represented in the case of discrete particle sets. These moment conditions can be ensured by appropriate normalizations [41]. Kernels of arbitrary order

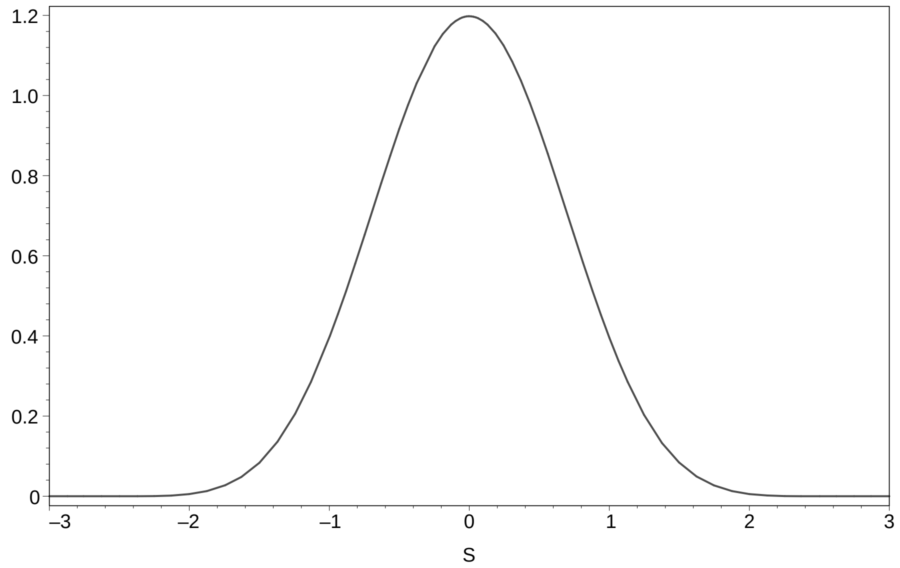


Figure 2.1: Plot of the quartic spline kernel $\bar{\zeta}_\epsilon(s)$ (Eq. (2.10))

[14] are possible by giving up the positivity property of the kernel function. In Appendix B, we present several examples of higher order kernels derived from exponential functions as introduced by Eldredge *et al.* [54].

The error estimates reveal a very important fact for smooth particle approximations. In order to obtain accurate approximations *smooth particles must overlap*.

2.3 Derivative Approximations

Particle approximations of the derivative operators can be constructed through their integral approximations. This can be achieved by taking the derivatives of Eq.(2.3) as convolution and derivative operators commute in unbounded or periodic domains.

2.3.1 General Deterministic Approximation

A general formulation involves the development of integral operators that are equivalent to differential operators [54]. They were first introduced for the integral approximation of the Laplacian [49] in the diffusion equation leading to the particle strength exchange (PSE). Eldredge *et al.* [54] presented a general deterministic integral representation for derivatives.

The differential operator is denoted as

$$D^{\beta}\Phi(\mathbf{x}) \;=\; \frac{\partial^{|\beta|}}{\partial x_1^{\beta_1}\,\partial x_2^{\beta_2}\cdots\partial x_d^{\beta_d}}\,\Phi \tag{2.12}$$

where $\beta = (\beta_1, \beta_2, \ldots, \beta_d)$ is a multiindex with $|\beta| = \beta_1 + \beta_2 + \ldots + \beta_d$ and the physical dimension d. Also $y^{\beta} = y^{\beta_1} y^{\beta_2} \ldots y^{\beta_d}$ and $\beta! = \beta_1!\beta_2! \ldots \beta_d!$.

The general integral operator approximating the action of an operator $D^{\beta}\Phi(\mathbf{x})$ on a function Φ is approximated as:

$$L_{\epsilon}^{\beta}\Phi \;=\; \int (\,\Phi(\mathbf{y}) \mp \Phi(\mathbf{x}))\zeta_{\epsilon}^{\beta}(\mathbf{x}-\mathbf{y})\,d\mathbf{y} \tag{2.13}$$

where the minus sign is chosen for $|\beta|$ even and the plus in case it is odd. The integral operator can be discretized using the particle locations as quadrature points resulting in an overall approximation of the order r:

$$\begin{aligned}
\|D^{\beta}\Phi \;-\; L_{\epsilon}^{\beta,h\Phi}\|_{0,2} \;&=\; \|(D^{\beta}f \;-\; L^{\beta}f) \;+\; (L^{\beta}f \;-\; L^{\beta,h})\|_{0,2} \\
&\leq \|(D^{\beta}\Phi \;-\; L_{\epsilon}^{\beta}\Phi)\|_{0,2} \;+\; \|(L_{\epsilon}^{\beta}\Phi \;-\; L_{\epsilon}^{\beta,h}\Phi\|_{0,2} \\
&\leq C_1\,\epsilon^{r}\|\Phi\|_{r+2,2} \;+\; C_2\,\frac{h^m}{\epsilon^{m+|\beta|-1}}\|\Phi\|_{m,2}
\end{aligned} \tag{2.14}$$

Note again that the first term implies the error due to the integral approximation, related to the moments of the mollifier and the second term is the error introduced by the quadrature, further strengthening the constraint for particle overlap.

The order of a derivative approximation can be determined by the evaluation of the α-Moments of a kernel $\eta(\mathbf{x})$ that are defined as

$$M_\alpha = \int \mathbf{x}^\alpha \eta(\mathbf{x}) d\mathbf{x}, \tag{2.15}$$

A kernel $\eta_\epsilon^\beta(\mathbf{x})$ of order r for the approximation of the operator D^β needs to satisfy the following moment conditions [54]

$$M_\alpha = \begin{cases} (-1)^{|\beta|}\beta!, & \alpha = \beta \\ \quad 0, & |\alpha| = |\beta|, \alpha \neq \beta \\ \quad 0, & |\alpha| \in [1, |\beta| - 1] \cup [|\beta| + 1, |\beta| + r - 1] \end{cases} \tag{2.16}$$

2.3.2 Moment Conditions

In this section, we prove that the derivatives of an r-order kernel satisfying Eqs.(2.4)-(2.6) lead to a derivative approximation of order r by showing the validity of Eq.(2.16).

Let assume the kernel $\eta_\epsilon^\beta(\mathbf{x})$ to be equal to the kernel derivative $D^\beta\zeta_\epsilon(\mathbf{x})$. Then the moment integral M_α (Eq.(2.15)) can be integrated by parts [24]

$$\begin{aligned} M_\alpha &= \int_{-\infty}^{\infty} \mathbf{x}^\alpha D^\beta \zeta_\epsilon(\mathbf{x}) d\mathbf{x} \\ &= \left[\mathbf{x}^\alpha D^{(\beta_1,\dots,\beta_k-1,\dots,\beta_d)} \zeta_\epsilon(\mathbf{x})\right]_{-\infty}^{\infty} \\ &\quad - \int_{-\infty}^{\infty} \alpha_k \, \mathbf{x}^{(\alpha_1,\dots,\alpha_k-1,\dots,\alpha_d)} D^{(\beta_1,\dots,\beta_k-1,\dots,\beta_d)} \zeta_\epsilon(\mathbf{x}) d\mathbf{x} \end{aligned} \tag{2.17}$$

Assuming the compactness of the kernel $D^\beta\zeta_\epsilon(\mathbf{x})$ we obtain

$$M_\alpha = -\alpha_k \int_{-\infty}^{\infty} \mathbf{x}^{(\alpha_1,\dots,\alpha_k-1,\dots,\alpha_d)} D^{(\beta_1,\dots,\beta_k-1,\dots,\beta_d)} \zeta_\epsilon(\mathbf{x}) d\mathbf{x}$$

Applying the integration of parts for β-times yields

$$M_\alpha = (-1)^\beta \frac{\alpha!}{(\alpha-\beta)!} \int_{-\infty}^{\infty} \mathbf{x}^{(\alpha-\beta)} \zeta_\epsilon(\mathbf{x}) d\mathbf{x}. \tag{2.18}$$

Using Eqs.(2.4)-(2.6) we obtain:

$$\int_{-\infty}^{\infty} \mathbf{x}^\alpha D^\beta \zeta_\epsilon(\mathbf{x}) d\mathbf{x} = \begin{cases} (-1)^{|\beta|}\alpha!, & \alpha = \beta \\ \quad 0, & |\alpha| = |\beta|, \alpha \neq \beta \\ \quad 0, & |\alpha| \in [1, |\beta| - 1] \cup [|\beta| + 1, |\beta| + r - 1] \end{cases} \tag{2.19}$$

which is equivalent to Eq.(2.16).

2.3.3 Discretized Formulation

Kernels satisfying Eq.(2.19) are used in particle methods such as Smooth Particle Hydrodynamics (SPH) [107, 109] and remeshed SPH (rSPH)[29]. The derivatives of a field quantity Φ on a particle p are approximated in a conservative form as:

$$\left\langle D^{\beta}\Phi\right\rangle_{p} = \sum_{q} v_{q}\left(\Phi_{q} - \Phi_{p}\right) D^{\beta}\zeta_{\epsilon}(\mathbf{x}_{p} - \mathbf{x}_{q}), \tag{2.20}$$

where v_q is the volume of particle q. The normalization values for the kernels are chosen such that the corresponding non-zero moment condition of the discretized Eq.(2.16) is satisfied. The kernel of Eq. (2.10) has its first three derivatives continuous allowing a smooth approximation of the spatial derivatives of $\Phi(\mathbf{x})$.

2.4 Remeshing

A key aspect of the present method involves the use of a remeshing procedure. In smooth particle methods, as discussed earlier, particles must overlap at all times in order to guarantee the convergence of the method [44]. As it is shown in [40] remeshing is equivalent to a regularisation of the particle description of the advected quantities.

In this work remeshing is employed in order to regularize the distorted particle locations and to redistribute accordingly particle quantities onto a uniform set of particles with the spacing h. The redistribution of particle quantities is achieved using the 3rd order $M'4$ kernel [96] which in one dimension it is expressed as:

$$M_4^{'}(\mathbf{x}, h) = \begin{cases} 1 - \frac{5s^2}{2} + \frac{3s^3}{2} & 0 \leq s < 1, \qquad s = \frac{|\mathbf{x}|}{h} \\ \frac{(1-s)(2-s)^2}{2} & 1 \leq s < 2, \\ 0 & s \geq 2. \end{cases} . \tag{2.21}$$

In higher dimensions the interpolation formulas are tensorial products of their one-dimensional counterparts. Remeshing is necessary only when particles cease to overlap as they adapt to the flow map. In order to determine the rate at which particle remeshing is necessary we introduce a measure of distortion. This measure relies on the fact that the weighted sum $H_p(t)$ over all particles j at the position of particle p must be equal to unity in a regularized particle map

$$H_p(t) = \sum_j v_j(t)\zeta_\epsilon(\mathbf{x}_p(t) - \mathbf{x}_j(t)) \tag{2.22}$$

$$H_p(0) = H_{0,p} = 1. \tag{2.23}$$

The average change of $H_p(t)$ over all particles is a measurement of distortion

$$\Delta H = \frac{1}{N}\sum_j \frac{|H_j(t) - H_{0,j}|}{H_{0,j}}, \tag{2.24}$$

where N is the number of particles, and $H_j(t)$, $H_{0,j}$ are the weighted sums of particle j as presented in Eq.(2.22) and (2.23), respectively. When considering a purely rigid body motion ΔH is zero. In our simulations remeshing is invoked each time the function ΔH exceeds a small prespecified threshold.

At flat interfaces, we apply a one-sided remeshing technique as described in [42]. The particle that resides in the cell closest to the boundary (cell J centered at $\mathbf{x}_J$) is redistributed according to the weights (cf. Fig 2.2)

$$\Lambda_J(x_p, h) = \begin{cases} 1 - \frac{3}{2}v + \frac{1}{2}v^2 & \text{for cell } J, \qquad v = \frac{|\mathbf{x}_p - \mathbf{x}_J|}{h}, \\ v(2-v) & \text{for cell } J+1, \\ \frac{1}{2}v(v-1) & \text{for cell } J+2, \\ 0 & \text{for cell } J+3. \end{cases} \tag{2.25}$$

In the cell $J+1$ the support of the $M'4$ kernel still exceed the border of the domain, and a kernel of smaller support, here the $\Lambda_2(r,h)$, is used

$$\Lambda_2(r,h) = \begin{cases} 1 - s^2 & 0 \le s < \frac{1}{2}, \qquad s = \frac{|r|}{h} \\ \frac{1}{2}(1-s)(2-s) & \frac{1}{2} \le s < \frac{3}{2}, \\ 0 & s \ge \frac{3}{2}. \end{cases} \tag{2.26}$$

Remeshing at complex boundaries requires a normalization scheme of the remeshed quantities. The normalization scheme is similar to the normalization that ensures the partition of unity (Eq. (2.11)).

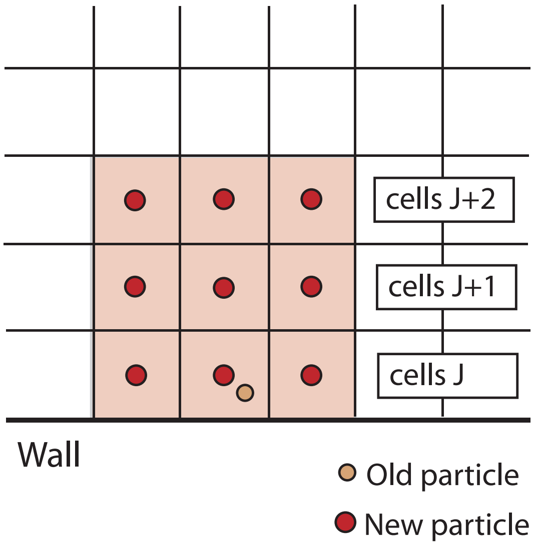


Figure 2.2: Remeshing in a bounded domain

$$\Phi_{j,new} = \frac{V_{new}}{\sum_{i=1}^{N_{old}} V_p M'4(|\mathbf{x}_i - \mathbf{x}_j|, h)} \sum_{i=1}^{N_{old}} \Phi_{i,old} M'4(|\mathbf{x}_i - \mathbf{x}_j|, h) \tag{2.27}$$

where $V_{new} = h^3$ is the volume of the new particle. We redistribute the extensive properties of the particle that requires conservation, e.g. mass and momentum.

2.5 Hybrid Particle-Mesh Methods

In hybrid particle-mesh methods, as introduced by Harlow [77], some of the differential operators are evaluated on a superimposed regular Cartesian mesh. The functions $\mathbf{U}$ and $\mathbf{F}$ of Eqs.(2.1) and (2.2) are evaluated on a mesh through the corresponding field equation. The hybrid method requires:

- the interpolation of the ω_p carried by the particles from the irregular particle locations $\mathbf{x}_p$ onto the regular mesh points (Φ_m)

$$\Phi_m = \sum_{p=1}^{N} Q(\mathbf{x}_m - \mathbf{x}_p)\omega_p, \qquad m = 1, \ldots, M \tag{2.28}$$

- the interpolation of the field quantity F_m from the mesh to the particle locations (F_p).

$$F_p = \sum_{m=1}^{M} R(\mathbf{x}_p - \mathbf{x}_m) F_m, , \qquad p = 1, \ldots, N\,, \tag{2.29}$$

where Q and R are interpolation functions, e.g. the kernel M'4 as defined in Eq.(2.21). The accuracy of the method depends on the smoothness of $\mathbf{U}$ and $\mathbf{F}$, on the interpolation function, and on the discretization scheme employed for the solution of the field equations. To achieve high accuracy, the interpolation functions Q and R must be smooth to minimize local errors, and conserve the moments of the interpolated quantity to minimize far-field errors. In addition, it is necessary that Q is at least of the same order of accuracy as R, to avoid spurious forces [84]. This can be easily achieved by selecting the same type of interpolation, W, for both operations: $Q = R = W$.

2.6 Initial and Boundary Conditions

Initial and boundary conditions are imposed with the governing equations to achieve a well-posed problem. The initial conditions are usually prescribed functions that describe the field quantities evolved in the governing equations. The initial field distributions determine the initial set of particle attributes.

Homogeneous boundary conditions can be solve by particle images in the case of flat boundaries compared to the core size of the mollification kernel. This method was developed for PSE and consists of placing mirror particles in an neighborhood of the particles outside of the simulation domain. Thus, the mollified derivative of Eq.(2.20) for example becomes

$$\left\langle D^\beta \Phi \right\rangle_p = \sum_q v_q \left(\Phi_q - \Phi_p\right) \left(D^\beta \zeta_\epsilon(\mathbf{x}_p - \mathbf{x}_q) \pm D^\beta \zeta_\epsilon(\mathbf{x}_p + \mathbf{x}_q)\right), \tag{2.30}$$

where the positive sign between the kernel functions invokes a zero-flux Neumann boundary condition and the negative sign a Dirichlet boundary condition.

For inhomogeneous boundary conditions the particle attribute need to be adjusted in the vicinity of the boundary [99].

Alternatively, the use of ghost particle is commonly wide spread in association with the SPH methodology introduced by Takeda *et al.* [162]. Ghost particles reside outside the computational domain and their attributes are extrapolated such that the boundary condition is satisfied.

Randles *et al.* [130] presented a normalization technique for handling derivatives without using ghost particles. This method is particularly applied in solid mechanics solved by smoothed particles and has the same effect has the normalization presented in Eqs.(2.11) and (2.20).

Chapter 3

Particle Level Set Method

3.1 Introduction

The accurate and efficient simulation of interface evolution is of fundamental importance for a wide range of problems ranging from multi-phase flows and combustion to virtual surgery environments and computer animation. In these applications we can distinguish two broad classes of computational methods being used to describe the evolution of interfaces, namely: interface capturing and interface tracking methods.

In *capturing methods*, the interface is determined by an implicit function that is advected in the computational domain. The most common interface capturing methods include Volume of Fluid [83] and Level Set methods [120, 145]. Volume of Fluid (VOF) methods are inherently linked to fluid mechanics problems and to Eulerian discretizations of the flow equations. They have enjoyed significant success in simulations of free surface and multiphase flow phenomena [141]. Level Set (*LS*) methods [148, 118, 147, 119, 64, 149] employ an implicit function to describe the advection of the interface and are well suited to problems where interfaces undergo extremely large topological changes. They have been applied with significant success to problems ranging from fluid mechanics to image processing and materials science (see the textbooks [147, 118] and references therein). The LS equation is commonly solved in an Eulerian framework by using high order finite difference methods, such as the fifth-order accurate Hamilton -Jacobi WENO schemes [25]. The accuracy of interface capturing schemes is reduced when the interface develops structures whose length scales are smaller than those afforded by the Eulerian mesh

[134]. In addition time step limitations are introduced by the associated CFL condition for the discretization of the advection term. A number of remedies have been proposed to rectify this situation, such as high order ENO/WENO approximations, semi-Lagrangian techniques [156] and hybrid particle-level set techniques as introduced in [57]. In the latter work, the cells near the interface are seeded with marker particles in order to obtain sub-grid scale accuracy. This hybrid method has been shown to provide superior results for a number of benchmark problems in two and three dimensions. However, a number of open issues remain regarding the manner in which particles are introduced as well as the number of particles necessary to obtain a prescribed accuracy.

Tracking methods solve the interface evolution equation in a Lagrangian fashion, for example by evolving marker particles. The origin of tracking methods can be traced to the 1930's and to calculations made by hand by Rosenhead [135] to describe the evolution of a vortex sheet in incompressible flows. These calculations have been followed 40 years later by the introduction of vortex methods [36] and the method of contour dynamics [187]. A fundamental problem of Lagrangian methods is the distortion of the locations of the computational elements resulting in an inaccurate description of the interface. A regularisation procedure is necessary in order to compensate for this deffect and to maintain the accuracy of the method. In the context of LS methods rather ad-hoc procedures, such as particle insertion and deletion, have been proposed and in [57] it is argued that no a-priori way exists that may help in building a regularisation of these methods.

In this thesis we consider a novel particle level set method that overcomes the difficulties associated with the Lagrangian formulation of level set equations, using techniques originally developed for vortex particle methods. In the particle framework the level set function is mollified using suitable kernel functions and it is subsequently discretized using particles as quadrature points. The level set information can be recovered by a linear superposition of the quantities carried by the individual particles. The accuracy of the method requires that particles overlap (i.e. their distance is smaller than their core size) at all times [41]. This is ensured by a remeshing procedure as it was first introduced in the context of vortex particle methods [98].

The method enjoys the advantages of particle methods such as adaptivity and unconditional stability for the advection of the computational elements. The description of the level set function as a linear superposition of individual particle functions enables straightforward computations of interface normals and curvatures. This enables computations with curvature induced motion of the interface. In addition the simplicity of the method allows real time simulations involving the cutting and reconnection of interfaces.

The outline of the this chapter is as follows: In Section 3.2, we describe the the level set method followed by its particle representation in Section 3.3. The remeshing of the particles as well as the reinitialisation of the level set function is discussed in Section 4. In Section 3.5, we present the validation of our method by considering two and three dimension benchmark problems. Section 3.6 and 3.7 illustrates its use in microchip fabrication simulations and in virtual surgery environments, respectively.

3.2 Level Set Method

The Level Set method [120, 145] defines an interface $\Gamma(t)$ as the zero level set of a high dimensional, scalar function $\Phi(\mathbf{x}, t) : \mathbb{R}^3 \to \mathbb{R}$:

$$\Gamma(t) = \{\mathbf{x} \in \Omega : \Phi(\mathbf{x}, t) = 0\}, \tag{3.1}$$

where Ω is the computational domain. The level set function has the following properties:

$$\begin{aligned} \Phi(\mathbf{x}, t) &> 0, \quad \mathbf{x} \in \tilde{\Omega} \\ \Phi(\mathbf{x}, t) &\leq 0, \quad \mathbf{x} \notin \tilde{\Omega}, \end{aligned} \tag{3.2}$$

where $\tilde{\Omega} \subset \Omega$ is an open region bounded by Γ. The motion of the interface is driven by a velocity field $\mathbf{u}(\mathbf{x}, t)$ as:

$$\frac{\partial \Phi}{\partial t} + \mathbf{u} \cdot \nabla \Phi = 0 \text{ for } t > 0, \tag{3.3}$$

$$\Phi(\mathbf{x}, 0) = \Phi_0(\mathbf{x}). \tag{3.4}$$

The specific form of the velocity field depends on the problem under consideration.

The function Φ_0 is usually chosen as the signed distance to the interface such that $|\nabla\Phi| = 1$. However, during its evolution, the level set function $\Phi(t)$ can lose the property of being the distance function [158]. Reinitialization schemes such as fast marching methods have been introduced [145, 146] in order to maintain this property. Usually the evolution of the level set function is computed using grid-based methods and the spatial derivatives to determine the surface normal and curvature (cf. Eq. (3.6) and (3.7)) are calculated by finite difference schemes [147].

Alternatively the level set equation can be expressed in a Lagrangian framework using the material derivative $\frac{D}{Dt} = \frac{\partial}{\partial t} + \mathbf{u} \cdot \nabla$ as

$$\begin{aligned} \frac{D\Phi}{Dt} &= 0 \\ \frac{d\mathbf{x}}{dt} &= \mathbf{u}, \end{aligned} \tag{3.5}$$

where $\mathbf{x}$ denotes the characteristics of the equation. The Lagrangian description of the level set equation is utilised in interface tracking methods. These methods encounter difficulties when singularities are formed during the evolution of the interface and need to be complemented with suitable regularisation procedures [145] in order to recover a desired weak solution. In this work this regularisation is performed by a remeshing procedure (see Section 2.4).

The unit normal and the curvature of the interface can be evaluated based on the level set function by

$$\mathbf{n} = \frac{\nabla\Phi}{|\nabla\Phi|} \tag{3.6}$$

$$\kappa = \nabla \cdot \frac{\nabla\Phi}{|\nabla\Phi|} \tag{3.7}$$

The calculation of interface normals and curvature involves the computation of gradients of the level set function which is achieved on the mesh in grid based methods, whereas in the present particle methods they are calculated by taking suitable derivatives of the mollification kernels (see Section 2.3).

3.3 Particle Representation of Level Sets

We consider two descriptions of level set functions, namely: the signed distance function and the color function.

The *signed distance function* (SDF) is defined by Eq. (3.2) along with the constraint that

$$|\nabla\Phi(\mathbf{x},t)| = 1. \tag{3.8}$$

The absolute value of the SDF measures the distance to the interface and the sign of the function changes when crossing the interface.

The *color function* (CF) [113] is defined by a different characteristic constant on each subdomain separated by the interface. The CF used in this work is :

$$\Phi(\mathbf{x},t) = 1, \quad \mathbf{x} \in \tilde{\Omega} \tag{3.9}$$

$$\Phi(\mathbf{x},t) = 0, \quad \mathbf{x} \notin \tilde{\Omega}, \tag{3.10}$$

where $\tilde{\Omega} \subset \Omega$ is an open region bounded by the interface Γ.

In level set methods the SDF approach can be used for computing interface quantities such as surface tension. However, in cases where the distance information is not necessary, use of the CF can result in significant computational savings.

3.4 Lagrangian Particle Level Set

In the proposed Lagrangian method the evolution of the LS function Φ amounts to evolving the particles on which it is discretised. The particle position $\mathbf{x}_p$, volume v_p and level set value Φ_p, evolve by the following system of ordinary differential equations derived from Eq. (3.3):

$$\begin{aligned}\frac{d\Phi_p}{dt} &= 0\\ \frac{dv_p}{dt} &= \langle\nabla\cdot\mathbf{u}\rangle_p\, v_p\\ \frac{d\mathbf{x}_p}{dt} &= \mathbf{u}_p\end{aligned} \tag{3.11}$$

where $\langle \diamond \rangle_p$ denotes the derivative approximation on a particle p (cf. Eq. (2.20)).

An immediate implication of the Lagrangian description is that simulation of solid body rotation almost exact, but for the introduction of errors introduced by the particle initialisation and by the accuracy of the time integration. The spatial derivatives used in Eq. (3.6) and Eq. (3.7) are computed according to the superposition over all particles of Eq. (2.20)).

3.4.1 Reinitialization for Particle Level Sets

During its evolution, the level set function usually ceases to be the signed distance function. Techniques such as fast marching methods [34, 148, 145] and re-distancing algorithms [157] have been introduced in order to maintain this property by reinitializing the level set function.

A prerequisite for applying techniques such as the fast marching method is the regularity of the computational elements. This reinitialisation is straighforward when using an Eulerian description of the level set methods but , in general, it is not possible for an arbitrary particle distribution. Remeshing, however, offers the benefit that it distributes the particles on a cartesian mesh and it allows the implementation of the fast marching method.

Remeshing also enables the use of the level set redistancing algorithm introduced by Sussman [157] by solving the following equation on the regularized particle locations:

$$\Phi_t = sign(\Phi_0)\left(1 - |\nabla\Phi|\right), \Phi(\mathbf{x}, 0) = \Phi_0(\mathbf{x}) \quad \text{for} \quad t \to \infty. \tag{3.12}$$

The computational effort of this scheme can be significantly high when the time integration of Eq. (3.12) requires a small time step to ensure the convergence of the solution.

An alternative scheme, suited for particle methods as it does not require that particles are distributed on a regular mesh, was developed based on the first order approximation of the derivative $\frac{\partial\Phi(\mathbf{x},t)}{\partial\mathbf{x}} = \frac{\Phi(\mathbf{x},t)-\Phi(\mathbf{x}_0,t)}{\mathbf{x}-\mathbf{x}_0}$ where $\mathbf{x}_0$ is the position on the interface that minimizes $|\mathbf{x}-\mathbf{x}_0|$. Reformulation of this first-order accurate equation leads to a first order approximation of the distance to the interface that can be used in turn for reinitialization:

$$\Phi_{new}(\mathbf{x},t) = |\mathbf{x}-\mathbf{x}_0| = \left| \frac{\Phi_{old}(\mathbf{x},t) - \Phi_{old}(\mathbf{x}_0,t)}{\frac{\partial \Phi_{old}(\mathbf{x},t)}{\partial \mathbf{x}}} \right| = \left| \frac{\Phi_{old}(\mathbf{x},t)}{\frac{\partial \Phi_{old}(\mathbf{x},t)}{\partial \mathbf{x}}} \right|. \tag{3.13}$$

The approximation of the gradient of the level set that can be obtained on the particle locations using Eq. (2.20). We wish to mention here that the scheme was first proposed independently by Cottet [43] while we have also been recently aware of a similar work in [56].

We compare these schemes on the evolution of a one dimensional signed distance function. This function evolves with a non-uniform velocity field leading to loss of its signed distance property (Fig. 3.1). The Taylor series re-distancing of Eq. 3.13 is computationally efficient and it maintains the signed distance property very close to the interface (within two particle spacing) but fails in regions further away from the interface as high order derivatives become important. The redistancing scheme of [157], provides a good reconstruction within five iterations. The fast marching method yields the best results for the longer part of the domain and is overall the most efficient method in this test case. In the remaining part of this study we use the fast marching method for the re-initialisation.

3.4.2 Fast Marching Method

We base our implementation of the fast marching method on the work of Sethian [146] and Adalsteinsson *et al.* [2].

The fast marching method provides the numerical solution of the Eikonal equation

$$|\nabla d(x,y,z)| = F(x,y,z). \tag{3.14}$$

With the assumption of $F = 1$ and $d = 0$ on an interface, the solution of Eikonal equation describes the signed distance to the interface. A convenient upwind finite difference approximation of Eq.(3.14) is given by

$$\left[(D_{m,x}d)^2 + (D_{m,y}d)^2 + (D_{m,z}d)^2\right]^{1/2} = F(x,y,z), \tag{3.15}$$

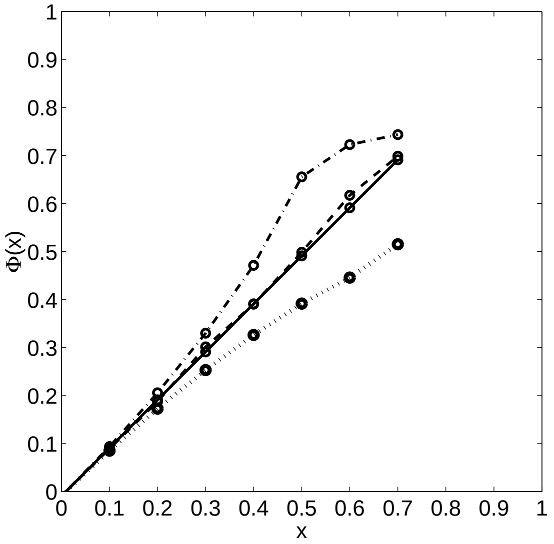


Figure 3.1: Comparison of the reinitialization methods in one dimension. A distorted signed-distance function (dotted line) is reinitialized by a Taylor series redistancing algorithm (Eq. (3.13)) (dash-dot line), the redistancing algorithm [157] (dashed line) and the fast marching method [147] (solid line)

where $D_{m,x}, D_{m,y}$ and $D_{m,z}$ denote differential operators defined by $D_{m,l}d = max\left(D^{-l}_{ijk}d - D^{+l}_{ijk}d, 0\right)$. The forward and backward operators D^{-l} and D^{+l} are finite difference approximations of first order accuracy.

Consider a two-dimensional version of the Eikonal equation (cf. Eq.(3.14)), where a set of boundary values is known at the upwind side (Fig. 3.2). The set of points neighboring the set of accepted values form the narrow band of trial values evaluated based on Eq.(3.15). The remaining set of points at the downwind side carry 'far away values' that are not defined. The algorithm of the fast marching method bases on the observation that the smallest value in the narrow band of trial values must be correct. The speed of the algorithm comes from a heapsort technique to efficiently locate the smallest element in the set '*Trial*'.

The fast marching algorithm can be described by: First, tag points with the initial conditions as '*Alive*', tag as '*Close*' all points one grid point away and tag as '*Far*' all other grid points. The loop sweeping the computational domain is as follows

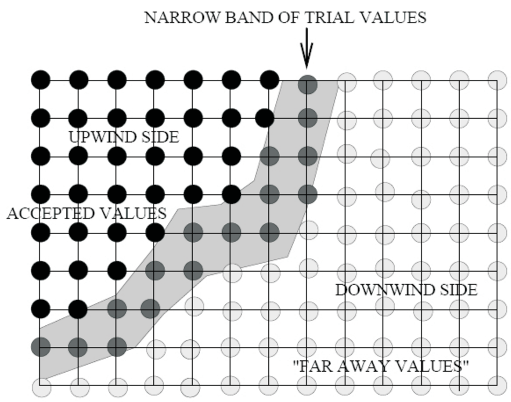


Figure 3.2: Typical setup for the upwind computation of accepted values in the fast marching method (in courtesy of Sethian [146])

1. Let *Trial* be the point in *Close* with the smallest value of d.
2. Tag as *Close* all neighbors of *Trial* that are not *Alive*: If the neighbor is in *Far*, remove it from that list and add it to the set *Close*.
3. Recompute the values of d at all *Close* neighbors of *Trial* by solving the piecewise quadratic equation according to Eq.(3.15).
4. Add the point *Trial* to *Alive*; remove it from *Close*.
5. Repeat loop until all points are in *Alive*.

Adalsteinsson *et al.* [2] illustrate the computation of an estimation for the initial set of level set values to start the algorithm with. We extend their technique [2] in the case a where only one of the neighboring points of the considered close point is on the other side of the zero level set (Fig. 3.3). In this case, we do not define the level set value as the distance to the intersection point on the line connecting the two grid points 1 and 2, but we rescale the distance according to the surface normal as estimated by a central finite difference scheme. This approach ensures that the difference in Φ of the grid points 1 and 2 in Fig. 3.3 is equal to the quotient h/n_y, where h is the grid spacing and n_y a projection of the local surface normal as shown in Fig. 3.3. Moreover, we do not implement the fast marching method separately for points inside and outside the front, but instead we use the method based on the initial set of points inside the front to both directions. This ensures that the gradient of Φ converges to one within the first order accuracy allowed by the fast marching method.

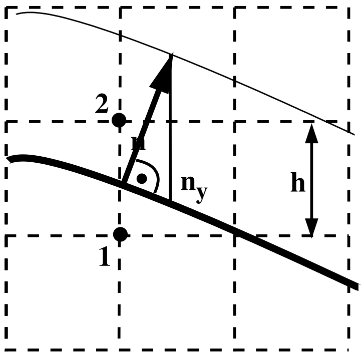


Figure 3.3: Extension to initial setting of Adalsteinsson [2]: the difference in the level set values of points 1 and 2 is not assumed to be equal to the grid spacing h but to the quotient h/n_y where n_y is a projection of the local interface normal n.

3.4.3 Implementation

In order to reduce the computational cost of the level set method the computational elements are limited to narrow bands around the interface [145]. This concept is readily implemented in the present method due to the local support of the underlying particle based functions. The remeshing provides a consistent process by which particles near the interface of the level set are being introduced while particles away from the interface are eliminated. The particles carry an indicator function to define the domain of the narrow band.

The equations for the particle locations and volumes (Eq. (3.3)) are integrated using a Runge-Kutta method of 4th order in all cases. To reduce the computational cost involved with the reconstruction of the level set function from the individual particles (Eq. (2.7)) we use Linked List [84] and Verlet Lists [172] (see also Section 6.5). The overall cost of the method scales linearly with the number of active particles. For 10^5 particles one time step of the method implemented in FORTRAN 90 requires 1.2 (in 2D) and 2.0 (in 3D) CPU seconds on an Apple Powerbook with a G4 processor of 1.25 GHz.

3.5 Results

We present results from the application of the method on a number of benchmark problems in two and three dimensions including pure convection and curvature induced motion. The method is compared quantitatively with existing grid based and hybrid grid-particle level set algorithms as reported in [57, 163, 35, 143]. The feasibility of the method as a tool for real time cutting procedures in virtual surgery environments is discussed.

3.5.1 Zalesak's Disk and Zalesak's Sphere

We demonstrate first the advantages of the proposed method on the rigid body rotation of the Zalesak's disk in a constant vorticity field [188]. Initially, a slotted disk is centered at (0.5, 0.75) with a radius of 0.15, a width of 0.05, and a slot length of 0.25. The velocity field is given by

$$\mathbf{v}(\mathbf{x}) = \frac{\pi}{314}\begin{bmatrix} 0.5 - y \\ x - 0.5 \end{bmatrix} \tag{3.16}$$

The disk completes one revolution in the unit domain every 628 time units. This test problem can identify diffusion errors of an interface-capturing method. In the proposed method, since the interface is driven by a rigid body rotation there is no particle distortion and hence no need for remeshing.

We present the solution after one revolution for the CF and the SDF description of the level sets, cf. Fig. 3.4. For the the CF approach, the present method requires only 591 particles uniformly distributed within the disk whereas the narrow band of the SDF initialisation involves 802 particles. The numerical errors result from the time integration scheme which is of 4th order, and the interpolation kernel used in the initialisation, which is of 2nd order. Fig. 3.5 shows that the error of area computation indeed converges with

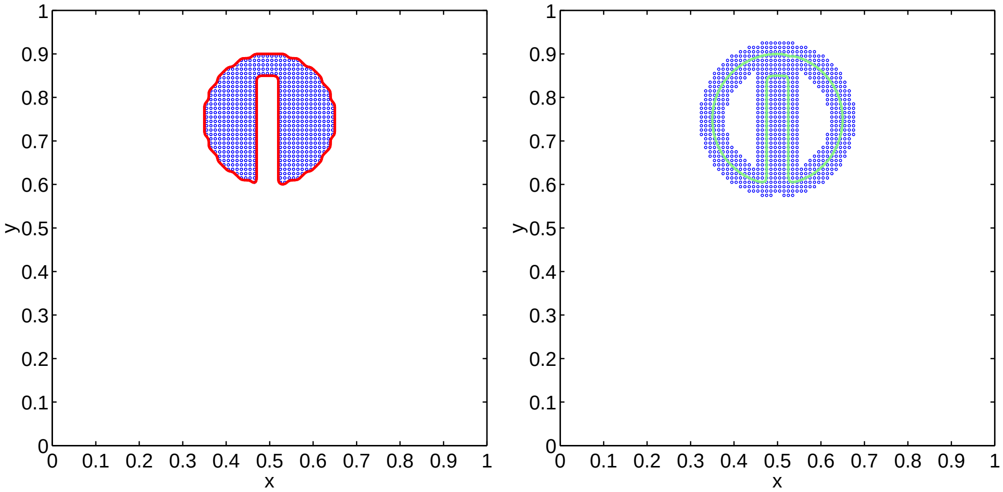


Figure 3.4: The Zalesak's disk after one revolution. The comparison of level set solutions using color function with particle positions (left, 591 particles) and signed distance function with a narrow band particles (right, 802 particles).

2nd order. The number of active particles are listed in Table 3.1. Note that the numbers for the particles being used along with the mesh in the hybrid methods are not reported [57] and those reported herein represent estimates with an average of 16 particles per cell. The results indicate that the present method requires on the average about five times less computational elements to achieve the same accuracy as the hybrid particle-level set approach of Enright *et al.* [57] when comparing the ratio between numerical error and number of computational elements for each resolution.

As a three dimensional test case we consider a slotted sphere, corresponding to the two dimensional Zalesak disk problem. The sphere has a radius of 0.15 and placed at (0.5, 0.75, 0.5) in a unit domain. The slot has a width of 0.05 and and length of 0.125. It rotates in the z=0.5 plane around the point (0.5, 0.5, 0.5).

Table 3.1: Absolute relative error of the area (exact 0.05822) in the slotted disk case using different Level Set Methods (LSM)

Spacing	Particle LSM		Hybrid Particle LSM [57]		LSM [57]
	Error	Particles	Error	Aux. Particles*	Error
Color function					
1/50	4.3%	153	14.9%	3328	100%
1/100	1.3%	591	0.3%	12864	5.3%
1/300	0.2%	5252	-	-	-
1/500	0.1%	14574	-	-	-
Signed distance function					
1/50	1.1%	208	14.9%	3328	100%
1/100	0.3%	804	0.3%	12864	5.3%
1/300	0.03%	7312	-	-	-
1/500	0.01%	20332	-	-	-

*estimated under the assumption of 16 auxiliary particles per narrow band cell

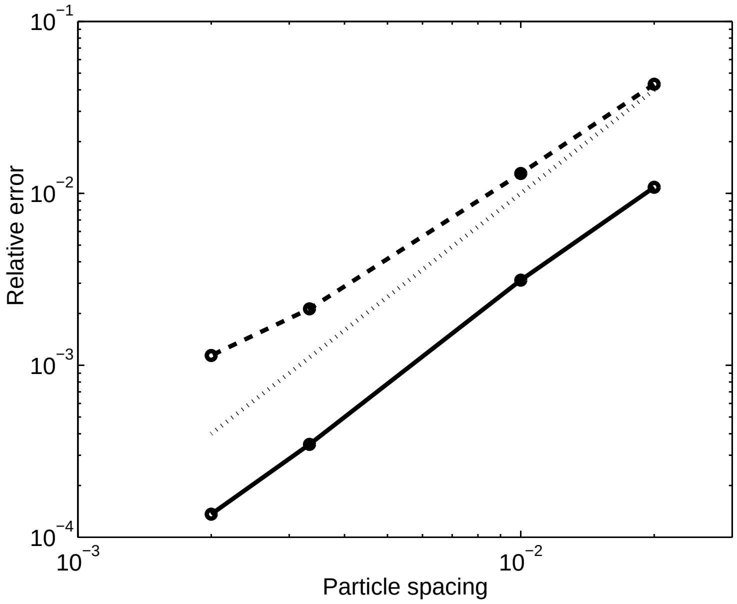


Figure 3.5: Comparison of the relative error of the area using color function (dashed line) and level sets (solid line) after one revolution to 2nd convergence (dotted line). The error matches with the initial error.

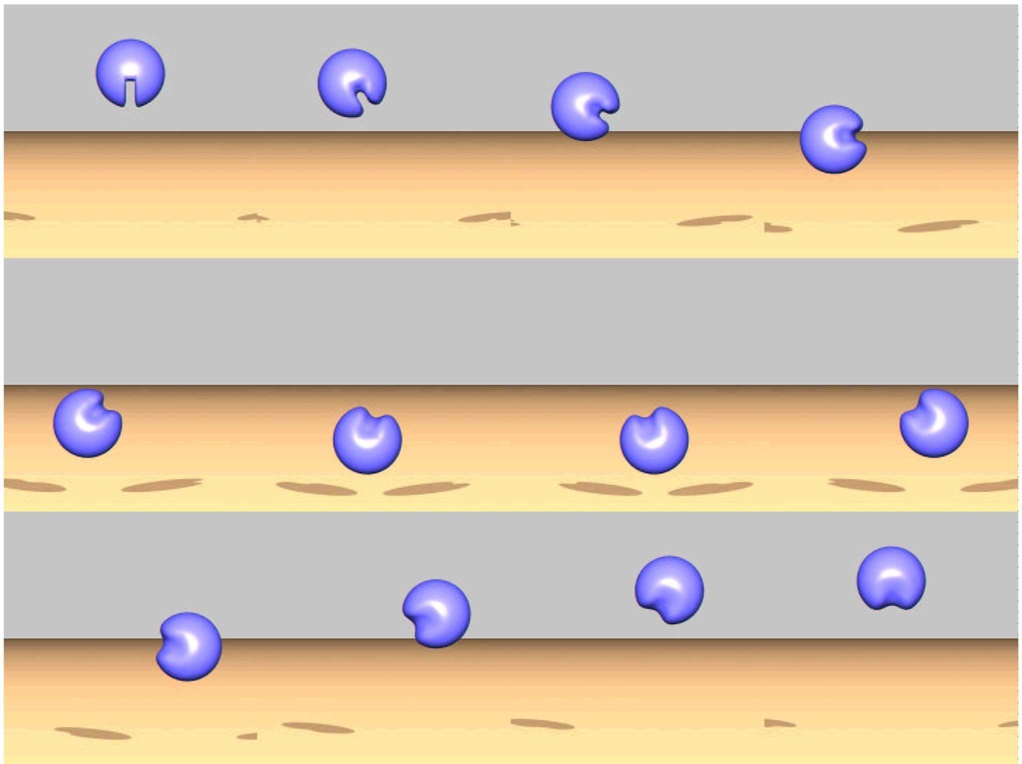


Figure 3.6: Zalesak's Sphere: level set solution using 100x100x100 cells (in courtesy of Enright et al. [57]).

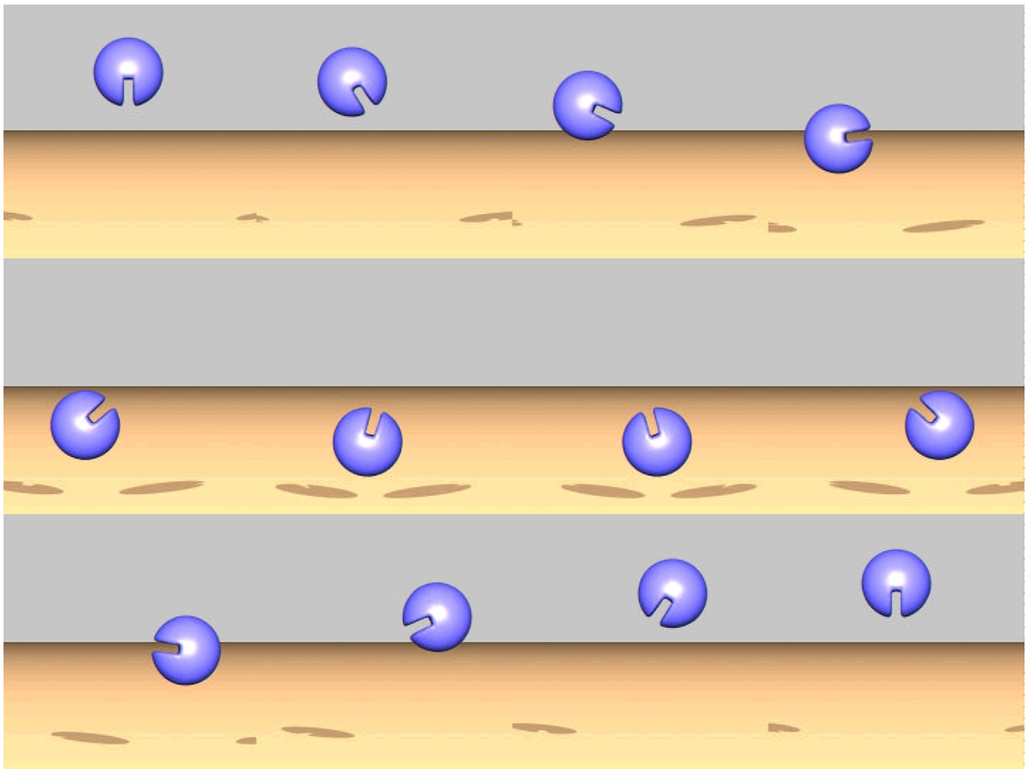


Figure 3.7: Zalesak's Sphere: particle level set solution of Enright et al. [57] using 100x100x100 cells and subscale particles.

The velocity field describes a rigid body rotation evolving over 628 time units per revolution

$$\mathbf{v}(\mathbf{x}) = \frac{\pi}{314} \begin{bmatrix} 0.5 - y \\ x - 0.5 \\ 0 \end{bmatrix} \tag{3.17}$$

As reported in [57] the level set solution with 100x100x100 cells (Fig. 3.6) suffers from numerical diffusion which can be alleviated by the hybrid particle level set method introduced in [57] as shown in Fig. 3.7.

In the present method the slotted sphere maintains its sharp features (Fig. 3.8) as the particles follow the rigid body rotation, without any numerical diffusion effects, associated with the advection of the level sets. Fig. 3.8 shows that the Lagrangian particle level set method performs very well on this problem. Since the particle level set function remains a SDF there is no need for reinitialization.

3.5.2 Single Vortex Flow

To test the scheme on resolving thin filaments as they occur in stretching and tearing flows we consider the single vortex flow as introduced by Bell *et al.* [16]. The initial interface, a circle placed at (0.5, 0.75) with radius 0.15, is shown in Fig. 3.9 together with the velocity field:

$$\mathbf{v}(\mathbf{x}) = 2 \begin{bmatrix} -\sin^2(\pi x)\sin(\pi y)\cos(\pi y) \\ \sin^2(\pi y)\sin(\pi x)\cos(\pi x) \end{bmatrix} \tag{3.18}$$

The velocity field stretches the circle into a long filamentary structure which wraps itself around the center of the unit domain. In this case the particles get very distorted by the flow and the remeshing procedure becomes an essential part of the scheme. The threshold for remeshing in our simulations is set to $\Delta H = 10^{-7}$. As a result, the remeshing scheme is applied every timestep. Fig. 3.10 shows the final interface at t=3 after 90 timesteps when 16384 particles are used to capture the interface by a CF. When the particle are not remeshed, they soon cease to overlap and the smoothness of the interface gets completely

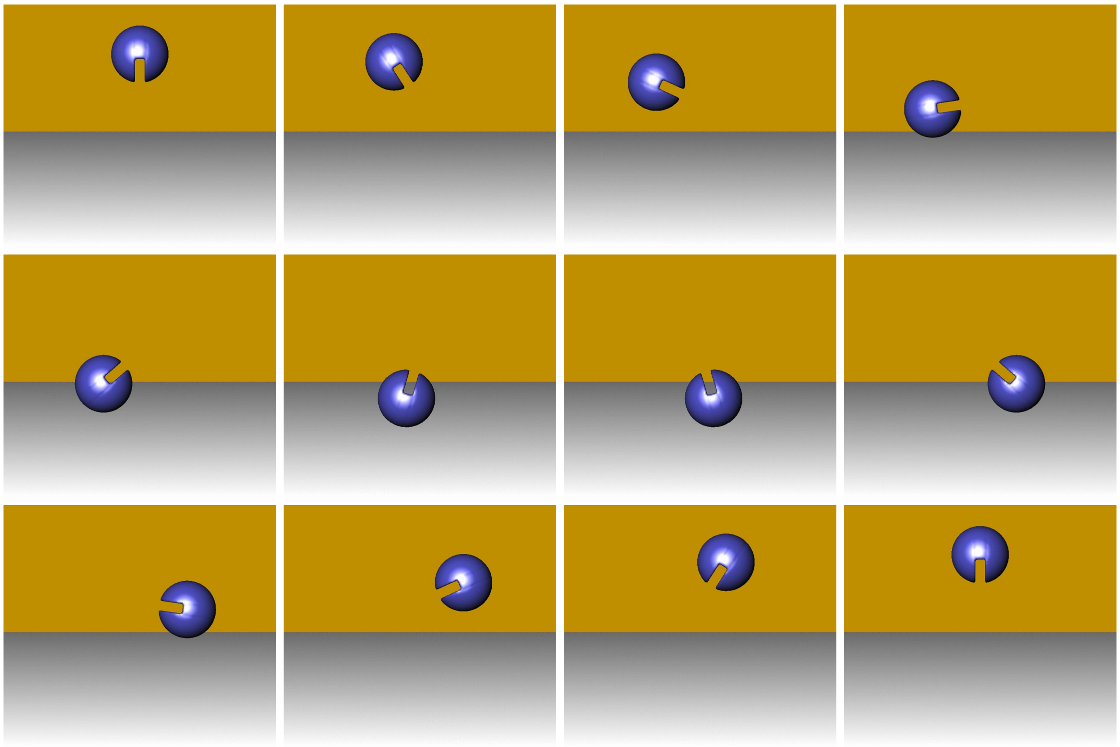


Figure 3.8: Zalesak's sphere during one revolution solved by Lagrangian particle level sets (24351 particles).

lost and the interface breaks apart as seen in Fig. 3.10 (left). Remeshing ensures that the particle overlap and the interface maintains its smooth features (cf. Fig. 3.10 (right)). Nevertheless in underresolved regions, particles do not describe accurately the level sets and the interface vanishes.

Using the SDF approach for initialization has similar results (Fig. 3.11). The remeshing scheme recovers the thin filament very well until the tail becomes underresolved. In both cases, the tail evolution is captured for much longer times when compared to the level set solution obtained by an Eulerian scheme (Fig. 3.13). Reinitialization is applied every 20 remeshing steps. Our results indicate that the frequency of reinitialisation can affect the interface when using the SDF description when applied too often (e.g. at every time step in conjuction with remeshing). When reinitialization is used less often the captured interface is better recovered but the signed distance property of the level set function is

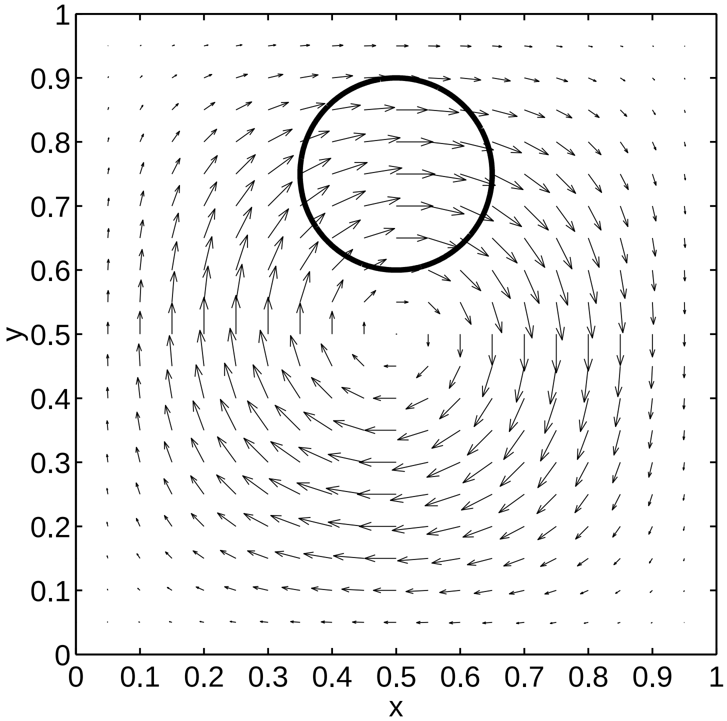


Figure 3.9: Single Vortex Flow : Initial interface with velocity field.

not guaranteed any more. Fig. 3.12 show results when the particle spacing is equal to 10^{-6} and demonstrate that the CF and the SDF approach converge to the same interface reconstruction. The interface matches nicely with the high resolution front track method as shown in Fig. 3.13. The quality of the interface is also comparable with the hybrid particle level set method of Enright *et al.* [57]. In their work they used 128x128 cells along with a number of marker particles seeding randomly the area around the interface. The marker particles were used to track the interface and perform an error correction by providing subgrid resolution. The present Lagrangian particle method is similar to the hybrid particle method as particles are being implemented to adaptively enhance the evolution of the interface. However, the methods differ as in the present work particles are used also to describe the Lagrangian formulation of the level sets. In addition, in the present method, remeshing is utilised to introduce, in a consistent manner, particles in areas where the particle distribution gets non-uniform, in order to maintain the accuracy of the particle description and to regularize the evolution of the particles.

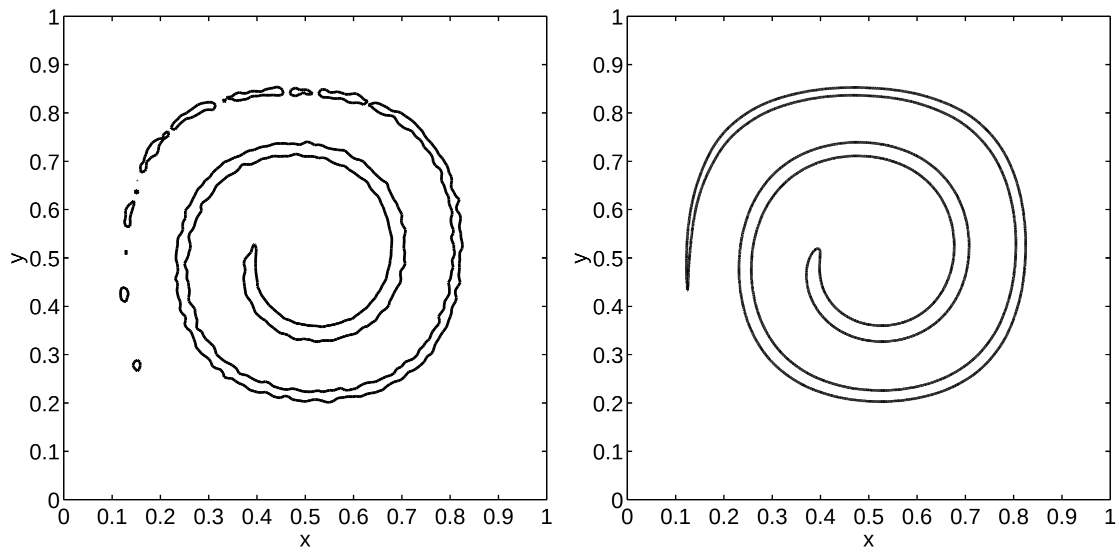


Figure 3.10: Zero level set of the single vortex problem at t=3 using Lagrangian particle level sets with a color function having remeshing suppressed (left, 1160 particles) and frequently remeshed (right, 3131 particles).

In order to quantify the error of the method the velocity field is reversed by multiplying its value by $\cos(\pi t/T)$ where T is the time of one period. For T=8 the maximal stretched interface is similar to the one in Fig. 3.12. The final interface at T=8 should match with the initial state. Table 3.2 shows the errors of the method in reconstructing the area as compared to the analytical solution. The errors indicate close to second order convergence when using the SDF without reinitialization and approximately first order convergence with reinitialization and when using the CF approach (Fig. 3.14). This implies that the error convergence behavior of the SDF approach with reinitialization results from the reinitialization scheme that shows first order accuracy. In general, the SDF approach delivers more accurate results than the CF approach. Figure 3.15 shows a selected number of the final interfaces of this problem. Note that the interface reconstruction severely suffers from the reinitialization in simulations using relatively low resolution (i.e. when h is larger than $1/128$).

The number of particle varies over time as remeshing creates a new set of particles each time it is applied. The number of particles for the time-reversed single vortex problem

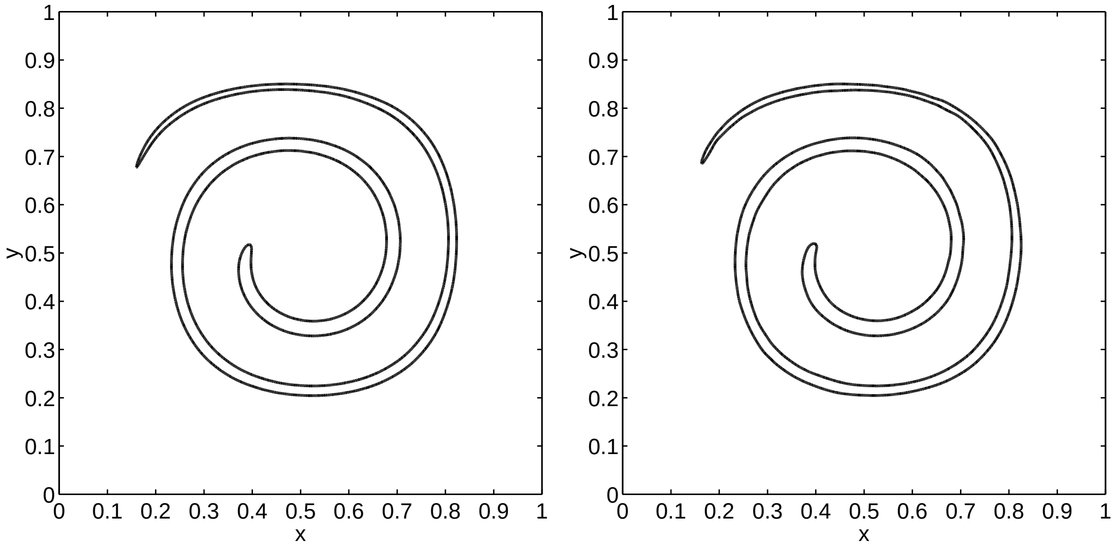


Figure 3.11: Zero level set of the single vortex problemat t=3 using Lagrangian particle level sets initialized by a signed distance function using remeshing only (left, 1860 Particles) and remeshing and reinitialization of the signed distance function (right, 4161 particles).

typically increases in the first half of the time period as the interface stretches out and decreases in the last third of the time period, cf. Fig. 3.16.

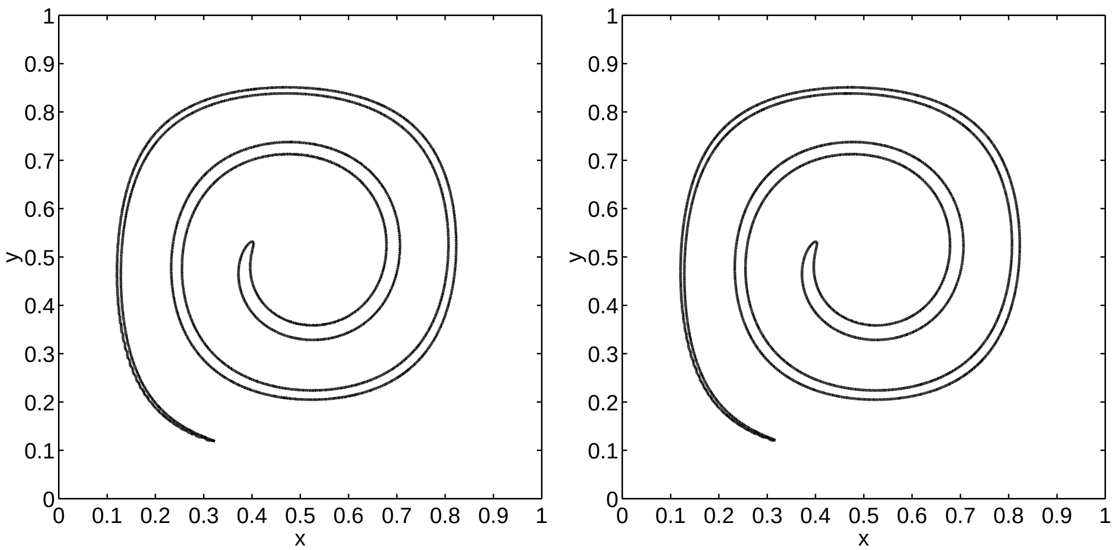


Figure 3.12: Zero level set of the single vortex problem at t=3 using Lagrangian particle level sets initialized by a color function (left, 70946 particles) and a signed distance function (right, 55914 particles).

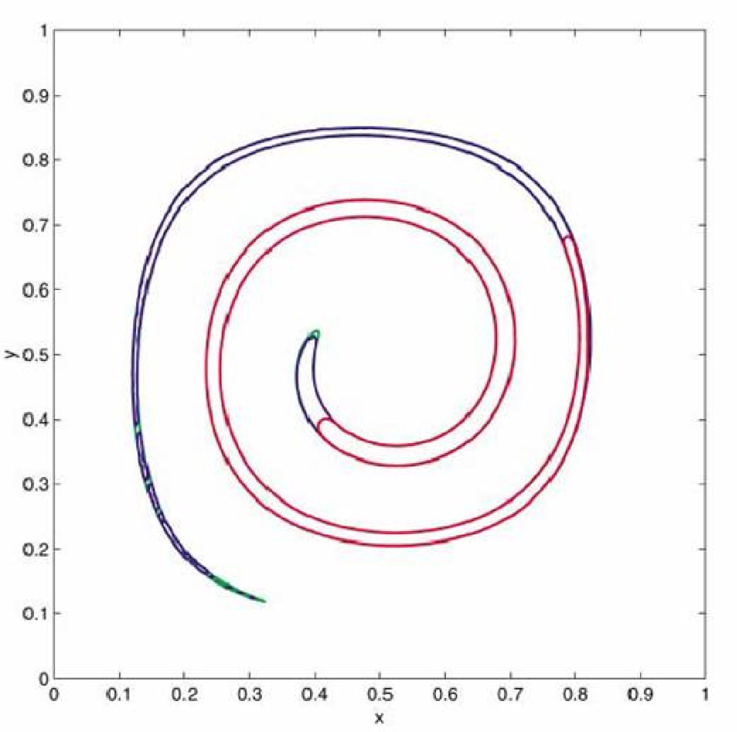


Figure 3.13: Zero level set at t=3 from Enright et al. [57]: Particle level set method 128x128 cells with subscale particles (blue), Level set method (red), high resolution front track method (green).

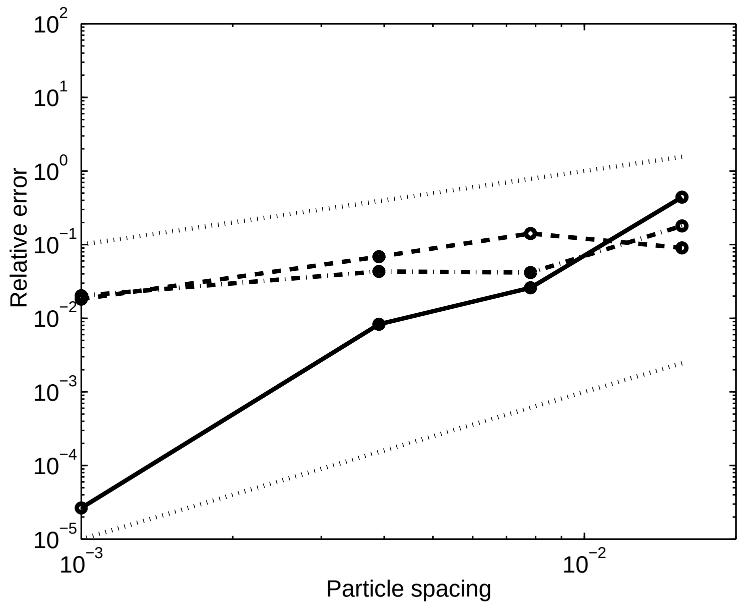


Figure 3.14: Single Vortex Flow : Comparison of the relative error of the area at $t = 8$ using CF(dashed line), and SDF level sets without (solid line) and with (dash-dot line) reinitialization. Dotted Lines showing first- and second-order scaling.

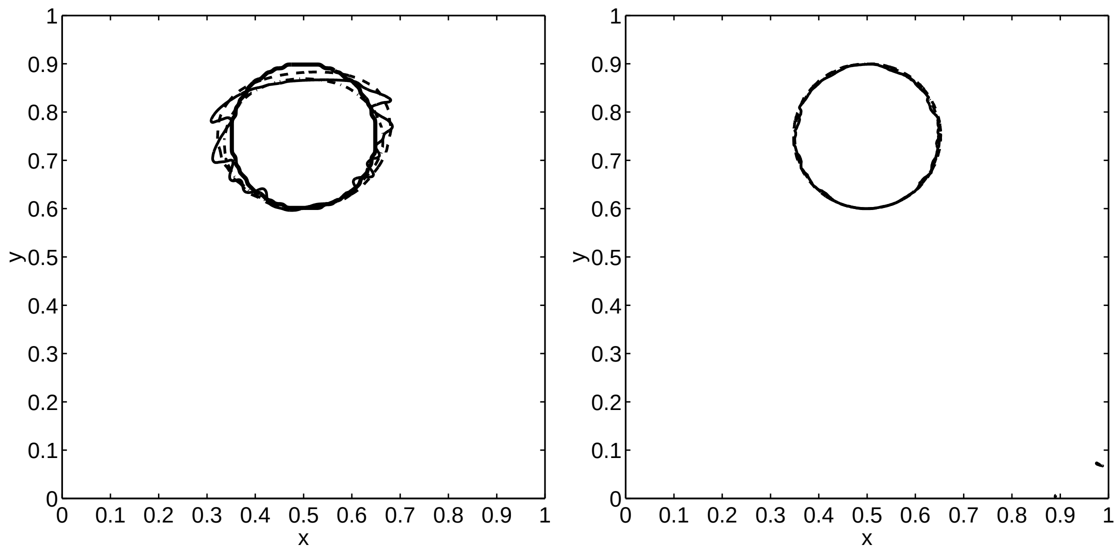


Figure 3.15: Time reversed single vortex problem. Zero level set at t=8 using Lagrangian particle level sets with a particle spacing h=1/128 (left) and h=1/1000 (right) using a color function having remeshing suppressed (thick solid line) and remeshed (dashed line) and using a signed distance function without (dash-dot line) and with reinitialization (thin solid line).

Table 3.2: Absolute relative error of the area (exact 0.07069) in the single vortex problem (240 time steps) using different Level Set Methods (LSM)

Spacing	Particle LSM		Hybrid Particle LSM [57]		LSM [57]
	Error	Particles ($t = 0$)	Error	Aux. Particles* ($t = 0$)	Error
Color function					
1/64	9.0%	284	1.8%	3776	100%
1/128	14.2%	1160	0.7%	15040	39.8%
1/256	6.8%	4628	0.4%	59072	10.3%
1/1000	1.8%	70688	-	-	-
Signed distance function without reinitialization					
1/64	44.1%	236	-	-	-
1/128	2.6%	940	-	-	-
1/256	0.8%	3692	-	-	-
1/1000	0.003%	55270	-	-	-
Signed distance function with reinitialization					
1/64	17.9%	236	1.8%	3776	100%
1/128	4.2%	940	0.7%	15040	39.8%
1/256	2.1%	3692	0.4%	59072	10.3%
1/1000	2.0%	56536	-	-	-

*estimated under the assumption of 16 auxiliary particles per narrow band cell

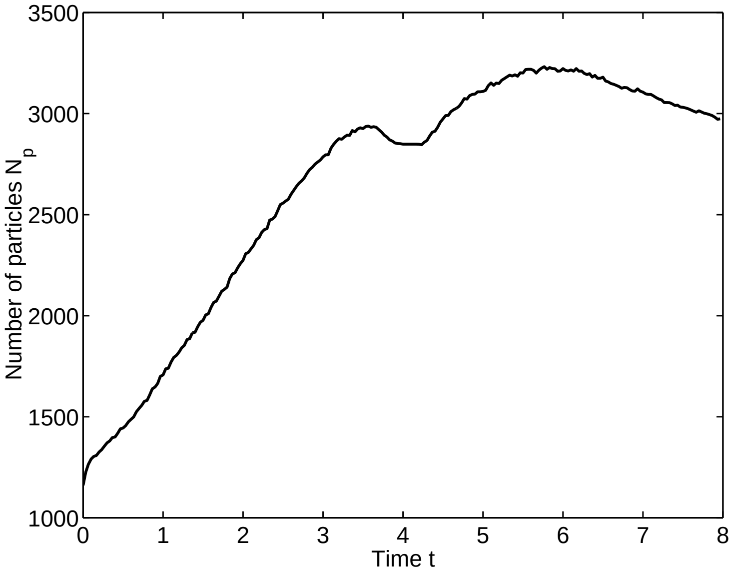


Figure 3.16: Time-reversed single vortex problem. The Number of particle versus time when using a color function and a particle spacing of h=1/128.

3.5.3 Deformation Test Case in Three Dimensions

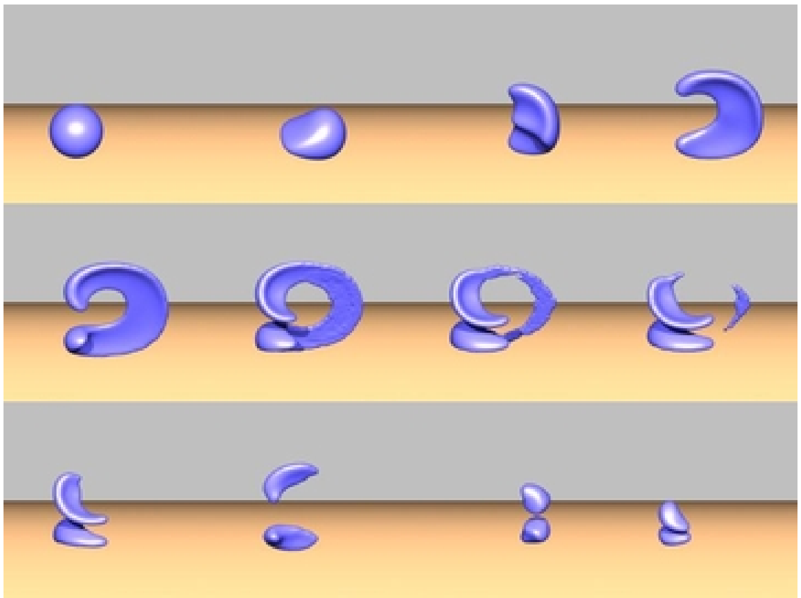


Figure 3.17: Deformation test case: level set solution using 100x100x100 cells (from Enright et al. [57].

To demonstrate the ability of the method to capture three dimensional deformations we consider a divergence-free velocity profile proposed by LeVeque [101].

$$\mathbf{v}(\mathbf{x}) = \begin{bmatrix} 2\sin^2(\pi x)\sin(2\pi y)\sin(2\pi z) \\ -\sin(2\pi x)\sin^2(\pi y)\sin(2\pi z) \\ -\sin(2\pi x)\sin(2\pi y)\sin^2(\pi z)\,. \end{bmatrix} \tag{3.19}$$

A sphere of radius 0.15 is placed within a unit computational domain at (0.35, 0.35, 0.35). At t=2 (after 75 time steps) the velocity field is reversed. Enright *et al.* [57] presented a level set solution and a hybrid particle level set solution to this problem, for parts of the interface that were underresolved as shown in Fig. 3.17 and Fig.3.18. The cell size of these solutions ($\Delta x = 0.01$) corresponds to the particle spacing of one of our simulations. For the same grid size and particle spacing the hybrid method and the present Lagrangian particle method, respectively, do not to resolve the thin interface. Note however that the present method involves a fraction of computational elements when

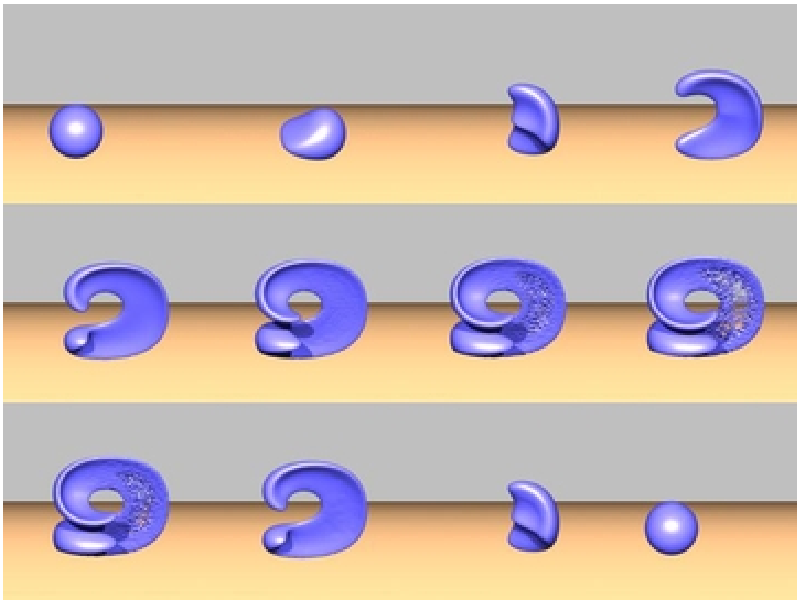


Figure 3.18: Deformation test case: particle level set solution of Enright et al. [57] using 100x100x100 cells and subscale particles.

compared to the hybrid method. Both methods give far better results than the purely Eulerian level set method which seems to fail severely on this problem 3.17. Figs. 3.19 and 3.20 show the Lagrangian particle level set solution with and without remeshing. Initially, particles are initialised using a CF. Particles carrying a color value of less than a small threshold (2.5e-2) are eliminated in further steps. Fig. 3.19 shows the simulation of the sphere when the particles are undergoing pure advectionwithout been remeshed. In this case the surface quickly becomes very rough and it is ruptured very early. Since the particles are only advected, the final surface after evolving for one period returns perfectly to the initial condition. Remeshing of the particles every second timestep improves the smoothness of the surface and delays the rupture as shown in Fig. 3.20. However, in underresolved regions the interface disappears. The Lagrangian particle level set method is able to recover from this rupture and provides a final smooth surface that shows small spurious features in the $y = 0.5$ -plane but overall recovers well the initial condition. The hybrid particle level set method (Fig.3.18) results in a better final result as additional subscale particles are utilised, but it still cannot resolve the thin interface at the maximal stretching. Simulations using larger numbers of particles were performed in order to find the resolution that avoids the disappearance of the interface at its maximum stretching. As it is shown in Fig. 3.21 this is achieved using about 170,000 particles with a spacing that would correspond to an Eulerian mesh of 200x200x200 for a total of 8,000,000 grid points.

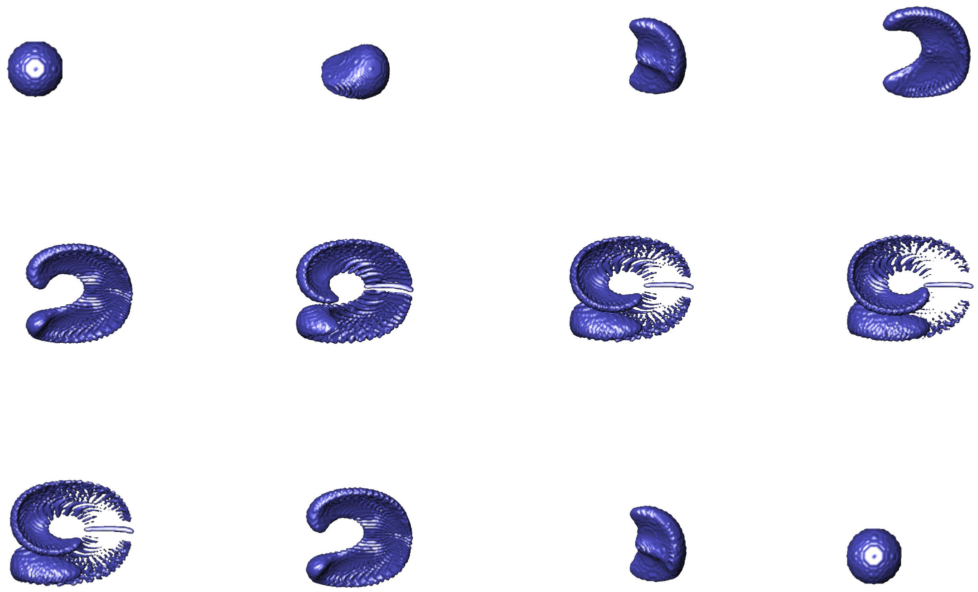


Figure 3.19: Deformation test case: particle level set solution without remeshing based on a color function (14054 particles). Lagrangian particle level set solution without remeshing at t=0, 15 Δs, 30 Δs, 40 Δs, 50 Δs, 60 Δs, 70 Δs, 75 Δs, 80 Δs, 100 Δs, 120 Δs 150 Δs, where $\Delta s = 2/150$.

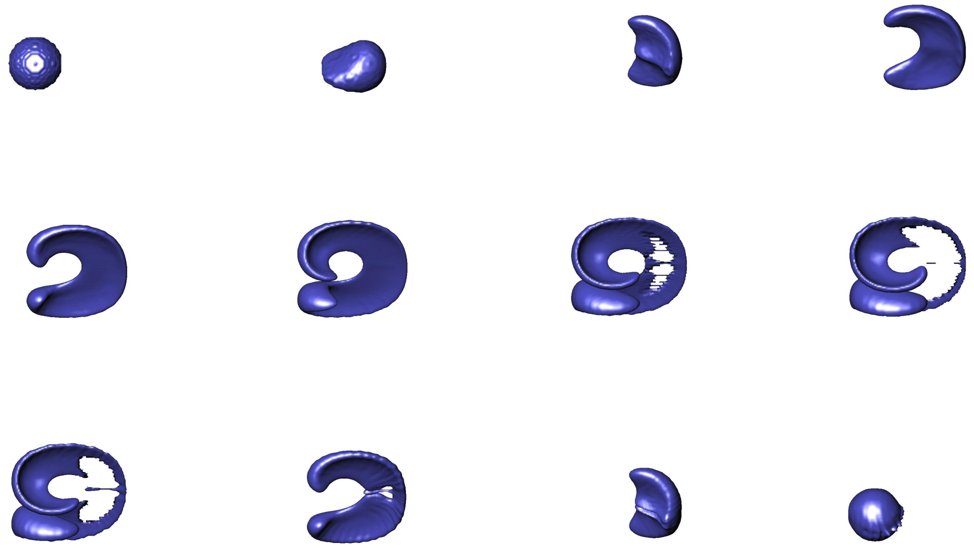


Figure 3.20: Deformation test case: Lagrangian particle level set solution based on a color function (minimum 14054 particles, maximum 30739 particles). Lagrangian particle level set solution with remeshing at t=0, 15 Δs, 30 Δs, 40 Δs, 50 Δs, 60 Δs, 70 Δs, 75 Δs, 80 Δs, 100 Δs, 120 Δs 150 Δs, where $\Delta s = 2/150$.

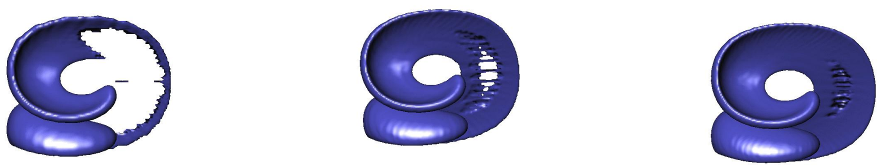


Figure 3.21: Effect of the resolution on the surface at t=1: h = 1/100, 29073 particles (left), h=1/160, 94033 particles (middle), and h=1/200, 169500 particles (right)

3.5.4 Flow under Mean Curvature

The particle description of the Level Set can be extended to advection determined by mean curvature of the interface as:

$$\frac{\partial \Phi}{\partial t} - \kappa \left| \nabla \Phi \right| = 0, \tag{3.20}$$

where κ is the curvature as defined by Eq. (3.7). The computation of spatial derivatives on particle locations makes use of the approximations described in Eq. (2.20). We determine the accuracy of the curvature evaluation by computing the curvature of an ellisoid with main axes' length of $a = 0.1, b = 0.2, c = 0.2$ and comparing it to the analytical solution. The L_∞ norm of error is shown in Fig. 3.22. The error of the SDF description converges with the 2nd order of the interpolation kernel while the CF description is less accurate.

To investigate the accuracy of our method in resolving interface normals using Eq. (3.6), we present the L_∞-error of the normal at a circular interface with radius R in terms of R/ϵ where ϵ is the characteristic length of the kernel (Fig. 3.23). We observe close to second order convergence for the SDF description while sublinear convergence is observed for the CF description.

In order to further validate our method on the computation of normals, We implement the SDF description in order to track an anisotropic evolution of an interface as proposed by Sethian *et al.* [144]. The anisotropic speed function u_A of this test case is fourfold

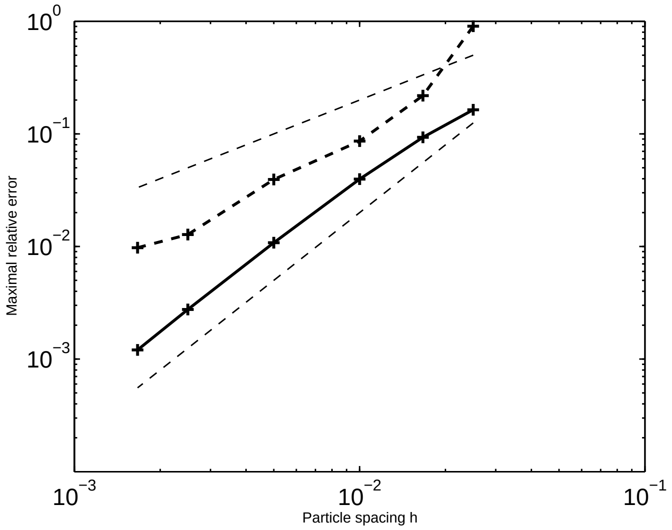


Figure 3.22: Convergence of the mean curvature of on the surface of ellipsoid based on the color function (dashed line) and signed distance function (bold line).

($k_A = 4$) and defined by

$$u_A = 1 - \epsilon_C \left(1 - \cos\left(k_A \left(\theta + \theta_0\right)\right)\right) \kappa. \tag{3.21}$$

where κ is the curvature and ϵ_C is set to 0.25. Fig. 3.24 shows the interface evolution for the phase angle $\theta_0 = 0$ (left) and $\theta_0 = \pi/4$ (right) using about 3000 particles. In both cases, the speed function transforms the circle into a square. In the first case, the preferred modes of growth are along the x- and y-axes, whereas in the second case the preferred modes of growth are along the diagonals. The corners turn out sharper in the first case. These results compare well with the results of Sethian *et al.* [144] using finite difference schemes.

We simulate the front evolution of an H-shape contour using Runge Kutta 4th order with a timestep of 10^{-4} and a particle spacing of 10^{-2}. Grayson [68] showed that all simple close curves flowing under curvature shrink to a point. The H-shaped contour is is progressively becoming a shrinking ellipsoid. We compare with the results of an Adaptive Mesh Redistibution (AMR) method [163] in Fig. 3.25. The effective resolution of the AMR is comparable to the resolution of the particle simulation. The two solutions

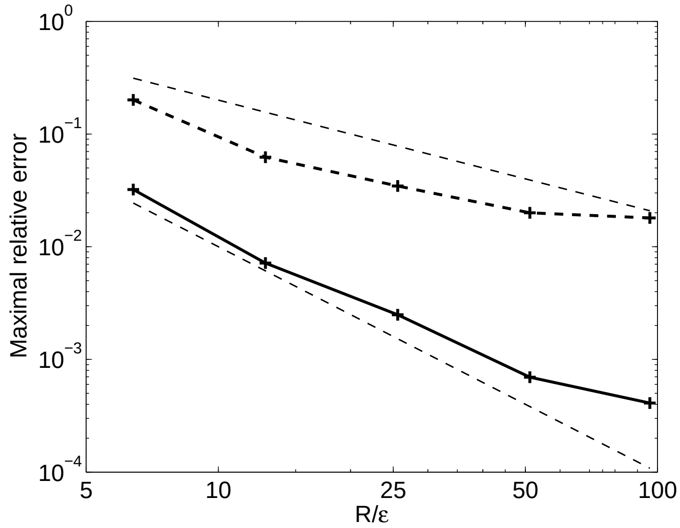


Figure 3.23: Convergence of the interface normals computed on a circle for the SDF (solid line with markers) and CF description (dashed line with markers).

match very well in the first 50 time steps and up to $t = 5 \cdot 10^{-3}$. The interfaces differ slightly at later time steps after the interface velocity has decayed significantly. As the analytical solution for this problem is not available we cannot judge which solution is more accurate.

To demonstrate the performance in three dimensions we simulate the collapse of a dumbbell that is a well known curvature flow example [35, 143] as it exposes a singularity. The mean curvature flow pinches off the handle that separates into two pieces, which continue to shrink and finally vanish. Grayson [69] used this example to show that non-convex shapes in three dimensions may in fact not shrink to one sphere. The dumbbell is made up of two spheres, each of radius 0.3, and connected by a cylindrical handle of radius 0.15. The x-axis is the axis of symmetry. We choose a particle spacing of $0.09\overline{7}$ and a time step of $2 \cdot 10^{-5}$. The particles are reinitialized every 10th time step. Fig. 3.26 shows the surface as it appears initially, after shrinkage, when reaching the singularity and after the break up. The quality of the results is comparable with the finite difference solution of Sethian [35, 143] as seen in Fig. 3.27. The resolution of domain and the size of the time step are equivalent in both simulations. The interface is plotted every 100 time

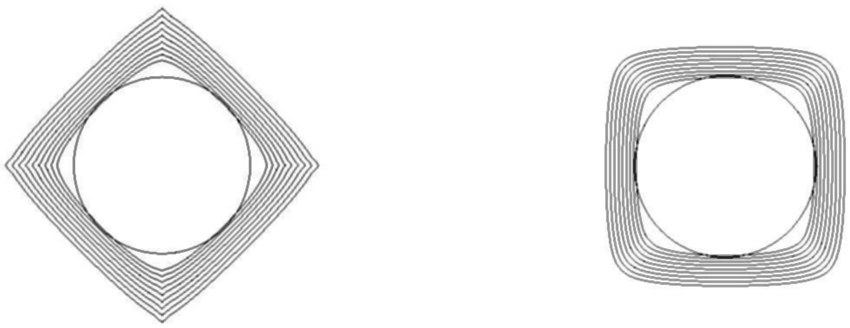


Figure 3.24: Interface evolution under a fourfold anisotropic speed function with phase angle $\theta_0 = 0$ (left) and $\theta_0 = \pi/4$ (right) using about 3000 particles.

steps, later, when approaching the singularity, it is plotted every 10 time steps.

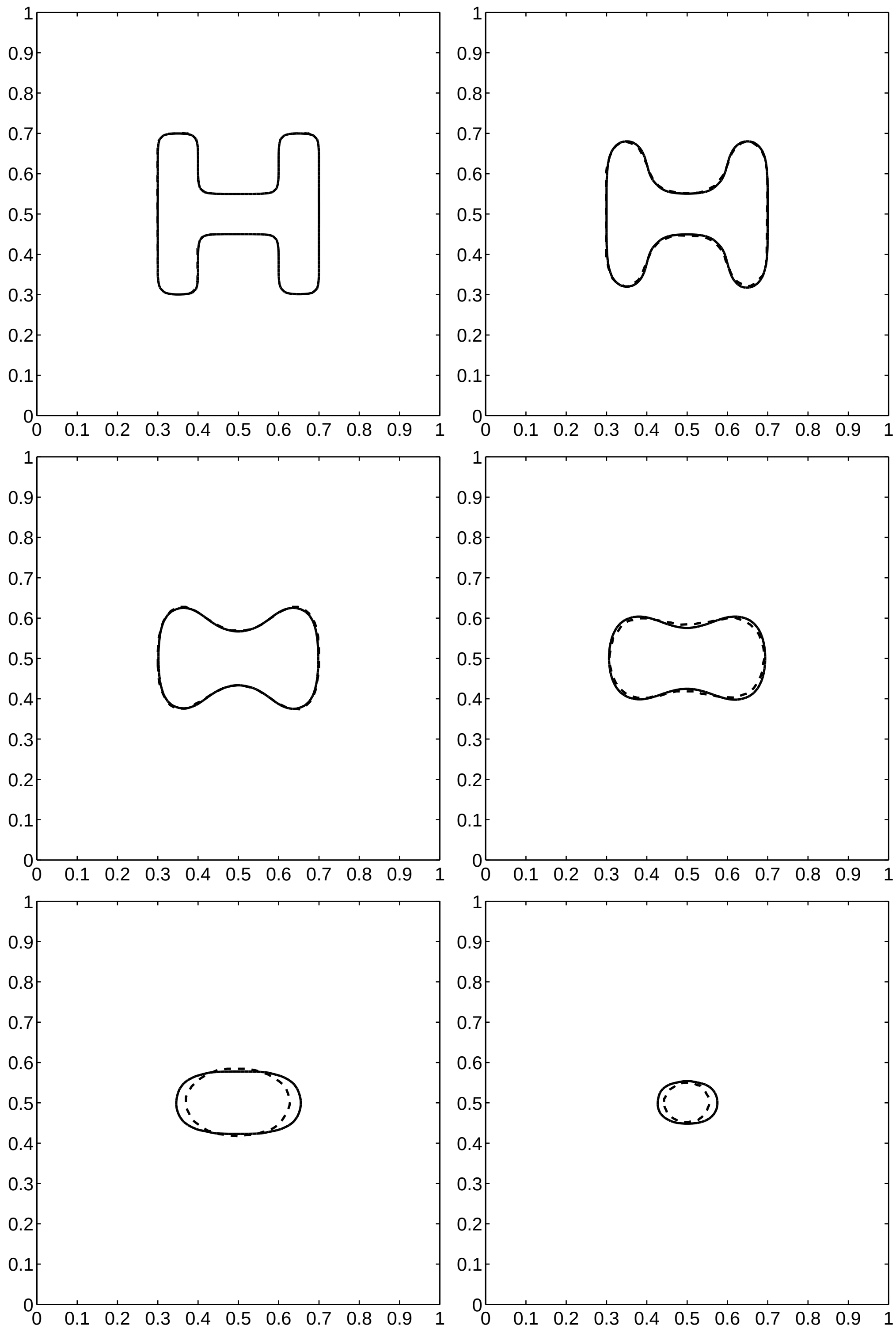


Figure 3.25: Curvature flow of an H-shaped contour at $t = 0, 10^{-3}, 3{\cdot}10^{-3}, 5{\cdot}10^{-3}, 10{\cdot}10^{-3}, 14{\cdot}10^{-3}$ *with initially 1701 particles (solid line) in comparison to Tang [163](dashed line).*

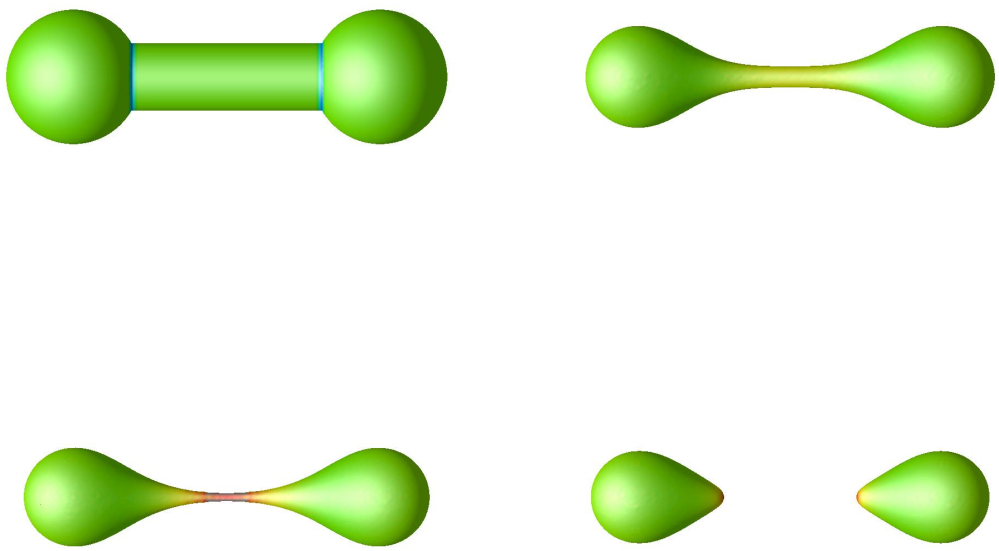


Figure 3.26: Evolution of a dumbbell shaped surface under mean curvature flow at $t = 0, 10^{-3}, 1.1 \cdot 10^{-3}, 1.26 \cdot 10^{-3}$ *(initially 334616 particles).*

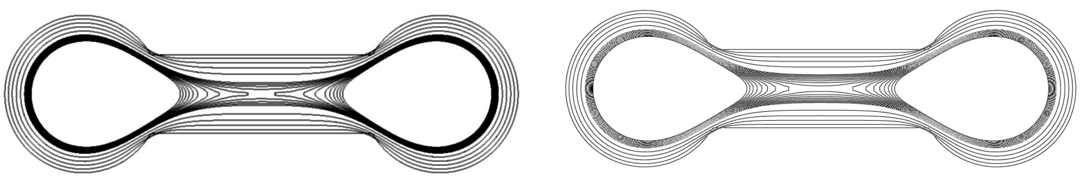


Figure 3.27: Evolving dumbbell (left) in comparison to Chopp and Sethian [35] (right).

3.6 Simulations of Processes in Microchip Fabrication

In this section, we demonstrate the performance of Particle Level Set Method for the simulation of the etching and deposition process in the fabrication of microchips (Fig 3.6).

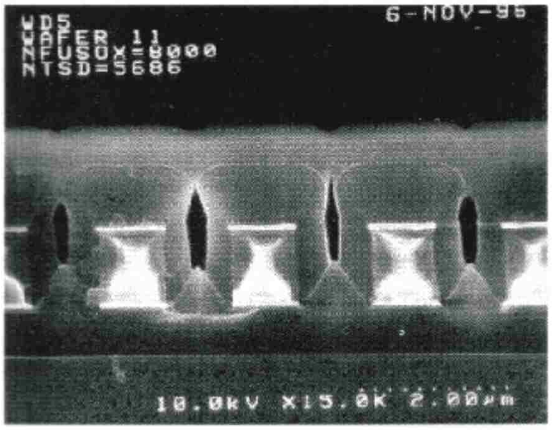


Figure 3.28: Microchip fabrication. Plasma-enhanced chemical vapor deposition.

During the fabrication of microchips a thin layer of crystialline silicon is oxidized, a *photoresist* is applied and the wafer is then covered by a pattern mask [147]. The mask shields part of the light-sensitive photoresist and contains the layout of the underlying circuit. Light causes the exposed photoresist to polymerize, leaving an unexposed material that is etched away. Silicon dioxide is then deposited at reduced pressure in a plasma discharge. Finally, thin films such as aluminum are deposited and contacts to the electrical components and components interconnections are established. These processes produce considerable changes in the surface profile as it undergoes various effects of etching and deposition. This problem is known as the ”surface topography problem” in microfabrication. The process is controlled by a large number of physical factors including the visibility of the etching/deposition source from each point of the profile, surface diffusion, and masking profiles.

The evolution of the surface profile [3, 4] can be described by a level set evolution equation similar to Eq.(3.3):

$$\frac{\partial \Phi}{\partial t} + \mathbf{F} \cdot \nabla \Phi = 0 \text{ for } t > 0, \qquad \Phi(\mathbf{x}, 0) = \Phi_0(\mathbf{x}), \tag{3.22}$$

where **F** is the local profile velocity that the depend on the considered process.

We consider the following processes

- Isotropic Etching ($\mathbf{F}_{Iso}^{Etch}$): Uniform etching, also known as chemical or wet etching
- Directional Etching ($\mathbf{F}_{Dir}^{Etch}$): Etching from an external source including visibility effects
- Isotropic Deposition ($\mathbf{F}_{Iso}^{Dep}$): Uniform deposition, also known as chemical or wet deposition
- Directional Deposition ($\mathbf{F}_{Dir}^{Dep}$): Deposition from an external source including visibility effects

To demonstrate the performance of the particle level set method we simulate the processes in a unit cube containing about 33000 particles carrying the level set function. The time step is 0.0035 using Runge Kutta 2nd order. To obtain the best results the particles are remeshed and reinitialized every time step.

3.6.1 Isotropic Etching and Deposition

In isotropic etching/deposition, the speed function F of the interface in its normal direction is independent of the orientation of the interface. In this case, inward pointing sharp corners are rounded when they etch inwards, and are outward pointing corners remain sharp when etched (the opposite situation takes place under deposition). This corresponds to the selection of the correct weak solution to the equations of motion [147].

Under isotropic etching ($\mathbf{F}_{Iso}^{Etch} = -1$) a cavity expands while the border surrounding the cavity is moving towards the bottom, , i.e., material is removed from the entire scene

(Fig.3.29). The results compare very well with the simulations of Adalsteinsson and Sethian [4]. In particular, the smoothing effects at the edges match very well.

Figure 3.29: Surface evolution during isotropic etching (left) in comparison with Adalsteinsson and Sethian [4] (right)

Under isotropic deposition ($\mathbf{F}_{Iso}^{Dep} = 1$) a cavity closes off while the border surrounding the cavity is moving upwards, , i.e., material is added to the entire scene (Fig.3.30). The results compare well visually to the simulations of Adalsteinsson and Sethian [4] as shown in Fig. 3.31.

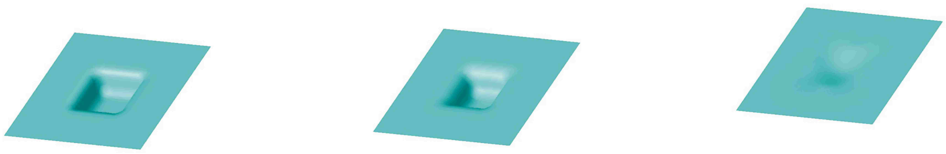


Figure 3.30: Surface evolution during isotropic deposition

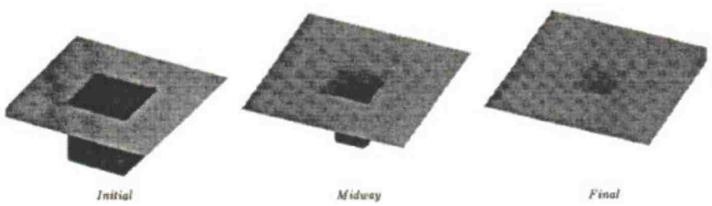


Figure 3.31: Surface evolution during isotropic deposition presented by Adalsteinsson and Sethian [4]

3.6.2 Directional Etching and Deposition

In this section, we analyze the outcome of deposition/etching including the effects of visibility. Let $Y(\theta)$ be the yield function, that is the effectiveness of the etching/deposition process. It is described by a function $Y(\theta)$ where θ is the angle between the surface normal and the direction of the light beam. In the case of pure isotropic deposition, we have $\mathbf{F}(\theta) = 1$, in the case of etching, $\mathbf{F}(\theta) = -1$. We choose $\mathbf{F}(\theta) = Y(\theta) = \cos\theta$, i.e. the beam causes the strongest effect when the beam is normal to the surface. Only regions that are visible from the source are affected.

We consider a cavity under directional etching and deposition where the light beam direction is $(1, -1, 1)$. During the directional etching process (Fig. 3.32 left) the material is only removed in lighted regions, whereas shadowed regions remain its shape because they are unreachable from the light source. The same effect applies to the directional deposition process (Fig. 3.32 right). However, the lighted part of the cavity is lifted.

The particle simulations are unconditionally stable for the advection as the computational elements move according to the velocity of the surface. Open issues involve the consideration of redeposition and reemission. During redeposition the material particles are expelled during the etching process and attach to the profile at other locations. Reemission describes the process where particle deposited during directional deposition may to stick to the profile and are re-emitted in the domain.

Figure 3.32: Surface of a cavity under directional etching (left) and deposition (right) in side and top view.

3.7 Virtual Cutting using Lagrangian Particle Level Sets

The simplicity and efficiency of the proposed method enables simulations associated with virtual cutting of soft biological tissue. We consider a liver topology that was segmented from image data of the Visible Human Project [114] into a triangular mesh. The graphical interactivity is provided by the OpenInventor Toolkit during the simulation. The collision detection is performed by a collision detection library for deformable objects provided by Heidelberger *et al.* [79].

3.7.1 OpenInventor Toolkit for interactive 3D Graphics

Open Inventor is an object-oriented toolkit for developing interactive, 3D graphics applications. It also defines a standard file format for exchanging 3D data among applications which serves as basis for the Virtual Reality Modeling Language (VRML) standard.

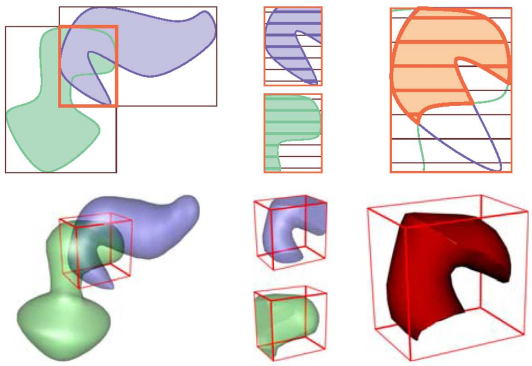
At the programming level, we use the following set of tools for graphics application development offered by the toolkit:

- A 3D scene database that includes shape, property, group, engine, and sensor objects, used to create a hierarchical 3D scene.
- A set of manipulators, including handle box and trackball, that allow the user to interact directly with the scene.
- An Component Library for Xt, including a render area, material editors, viewers and utility functions used to provide some high-level interactive tasks.

3.7.2 Collision Detection for Deformable Objects

The efficient detection of collision between geometric objects is an essential component in interactive simulations. Usually, the objects are triangle meshes with hundreds of thousands of elements. Thus, to accelerate the computation, spatial data structures like bounding-boxes and distance fields are commonly used. This structures are generated in a preprocessing step, thus removing the computational burden from the simulation. Although this approach performs very well for rigid objects, it is difficult to employ in cases where the geometry of the object changes over time. As, for instance, it is the case in a surgery simulation. Heidelberger *et al.* [79] present an algorithm based of Layered Depth Images (LDI) to accelerate collision detection of rigid and deformable objects.

The collision detection algorithm requires a closed triangular mesh of a 3D closed object as an input and returns a discrete representation of the intersection volume. The intersection volume provides a Vertex-in-volume test that is used for the collision response algorithm.



(a) Stage 1. (b) Stage 2. (c) Stage 3a.

Figure 3.33: Overview in 2D and in 3D. (a) AABB intersection. (b) LDI generation within the Volume of Interest. (c) Computation of the intersection volume (in courtesy of Heidelberger et al. [79]).

The collision detection algorithm proceeds in three stages (Fig. 3.33):

Stage 1 uses Axis-Aligned Bounding Box (AABB) to check whether two objects could overlap or not. This test is highly efficient because of the alignment of the bounding box axes. If the intersection is empty, the objects do not collide and we can avoid testing the primitives. If it is not, stages 2 and 3 are applied to the intersection volume (Volume of Interest).

Stage 2 computes two LDIs, one for each object. The LDI data structure essentially stores multiple depth values per pixel. Thus, an LDI can be used to approximate the volume of an object. The LDI generation is restricted to the Volume of Interest. The depth values of the LDI can be seen as 3D scan lines entering or leaving an object, as described on Figure 3.34. The intersection points are then classified into *entry* and *leaving points*. Likewise, the LDI classifies the Volume of Interest into *inside* and *outside regions*.

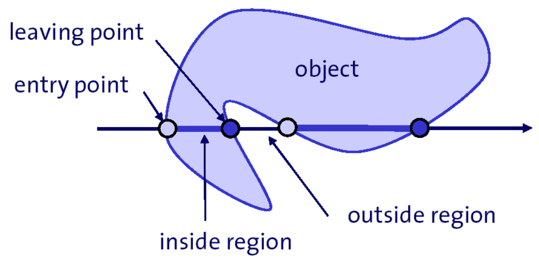


Figure 3.34: Inside and outside regions for a scan line.

Stage 3 performs the collision detection. Two kinds of operations can be realized with the resulting LDIs:

- The two LDIs can be combined to compute an intersection volume. The intersection volume is computed by a pixelwise intersection of the inside regions of both LDIs. If the resulting intersection is non-empty, a collision is detected. The sum of all intersections regions forms a discrete representation of the intersection volume.
- Individual vertices can be tested against an LDI. First, the vertices are transformed into the local coordinate system of the LDI. If the transformed vertex intersect with an inside region, a collision is detected.

3.7.3 Liver Reconstruction and Collision Response

Based on the a triangular mesh of the liver topology, particles are placed inside the liver surface and they are assigned values following a CF approach. Fig. 3.35 shows the surface reconstruction of the liver based on 3209 particles together with the medical devices visualized using the OpenInventor toolkit.

We apply the collision detection algorithm between the medical device and the liver at each time step. Whenever a medical device collides with one of the particles inside, the

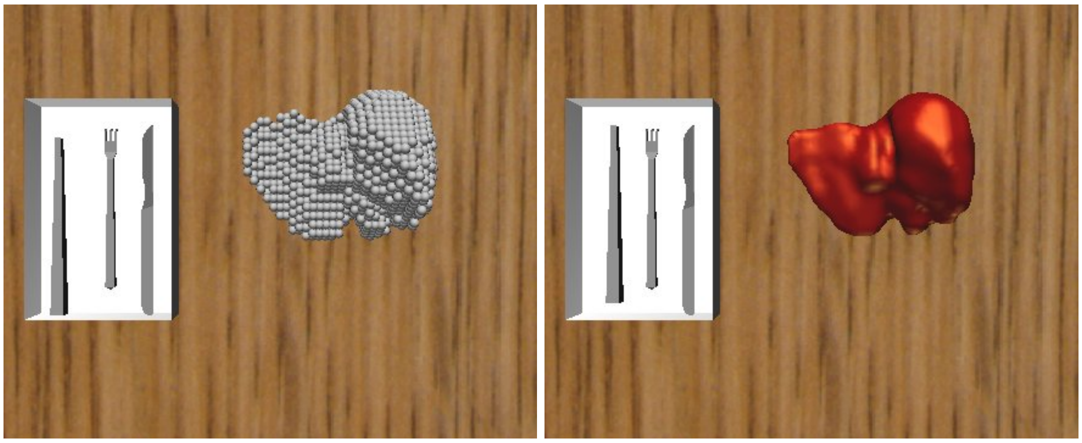


Figure 3.35: Liver topology reconstruction using 3209 particles.

contribution of this particle is removed from the superposition of Eq. (2.7) to simulate a cutting process. Hence reconstruction of the surface is computationally very inexpensive as the new surface is reconstructed according to:

$$\Phi(\mathbf{x})^{\text{new}} \;=\; \Phi(\mathbf{x})^{\text{old}} \;-\; \sum_{q}^{M} \upsilon_q \Phi_q \zeta_\epsilon(\mathbf{x} - \mathbf{x}_q) \tag{3.23}$$

where M denotes the (small) number of particles detected during the collision process. The triangular mesh of the surface is evaluated using the marching cube algorithm [103] and passed to the graphical renderer of OpenInventor.

3.7.4 Results

This algorithm shows high efficiency and enables interactive simulations (Fig. 3.36 and 3.37) when the devices moved into the liver are not thinner than the particle spacing. We obtain a realistic representation already with a small set of particle as shown in the sequence of Fig. 3.37. Adaptive insertion of particles having smaller core size is necessary in order to refine this process and to achieve a better mass conservation.

This approach, however, neglects the mechanical behavior of the liver and is, therefore, not capable of simulating deformations. In the appendix A, we present a physical approach to consider the effect of elasticity based on a simplified particle model for elastic solids.

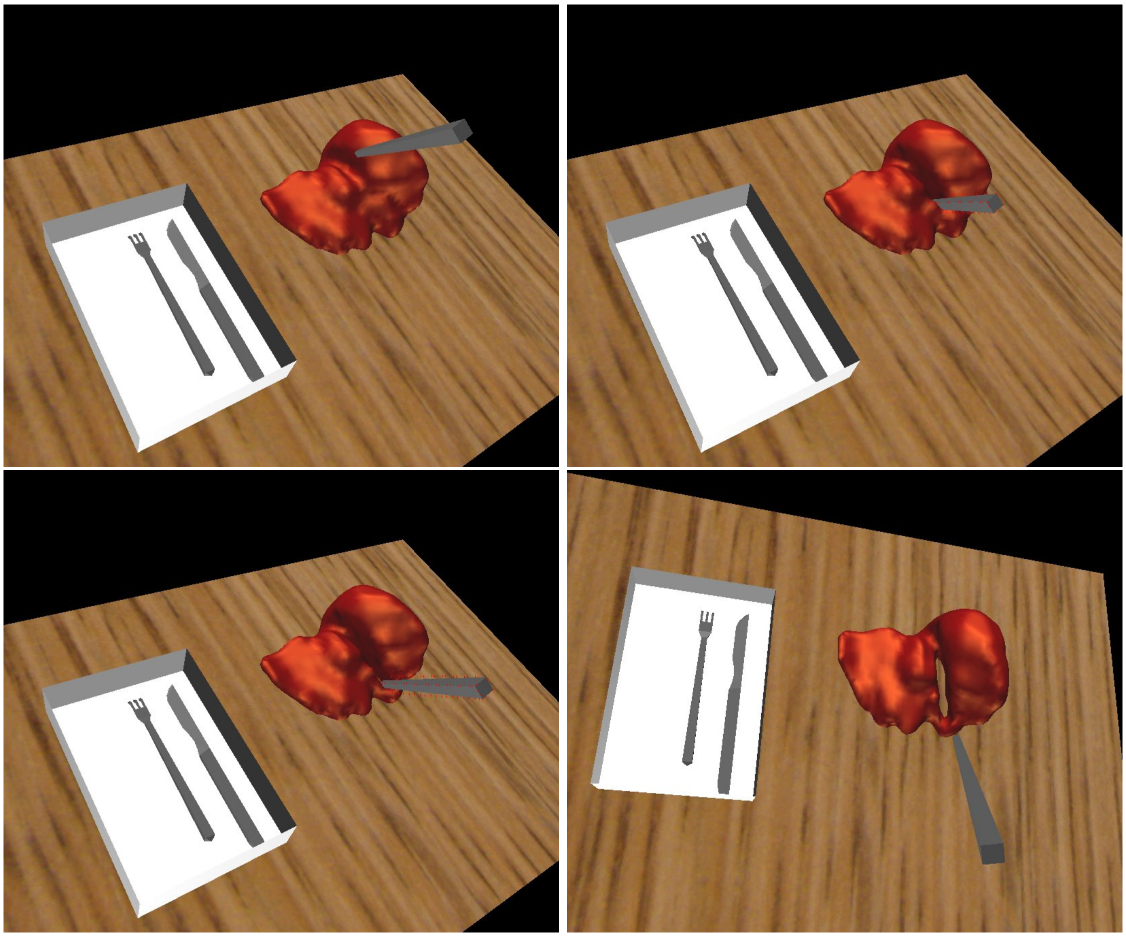


Figure 3.36: Particles assign with a color functions are removed from the superposition in real time when hit by an instrument.

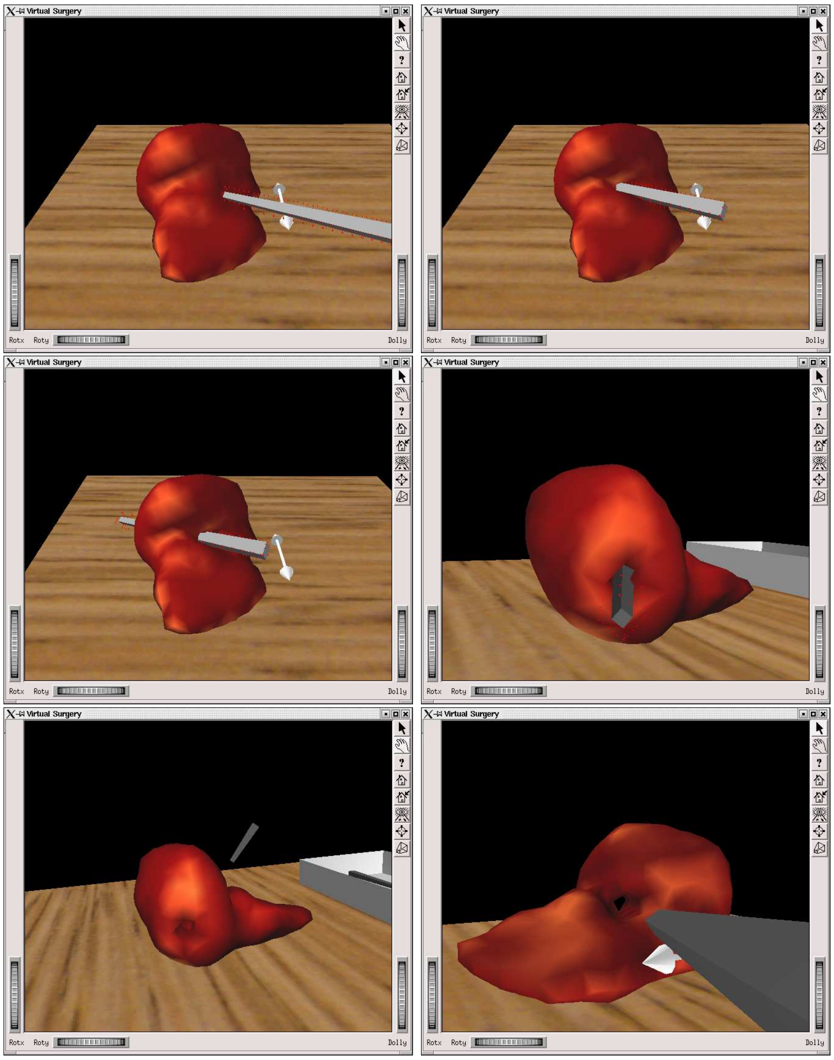


Figure 3.37: Perforation sequence using 950 particles.

Chapter 4

Particle Simulation of Fluids

4.1 Introduction

Fluid dynamics simulation is a classical research field where particle methods, such as vortex methods [41] and smoothed particle hydrodynamics (SPH) [109], are established numerical schemes.

SPH is a Lagrangian numerical method introduced by Gingold and Monaghan [65] to model and simulate problems in continuum physics while circumventing some of limitations of grid-based methods. This method enjoys the properties of Lagrangian schemes, such as automatic adaptivity and numerical stability. A key aspect is the approximations of function derivatives based on the particle superposition as presented in Section 2.3. Chaniotis *et al.* [29] presented the remeshed SPH (rSPH) method that combines the Lagrangian particle method with a regular remeshing scheme as described in Section 2.4 to simulate viscous and heat conducting flows.

This chapter presents the particle modeling and simulation of compressible fluid using rSPH for a compressible vortex ring demonstrating the stability and accuracy of the method. It furthermore, defines the characteristic numbers relevant for fluid dynamics.

4.2 Governing Equations

A system of differential equations govern the motion of a viscous, compressible medium. The fundamental system describes the conservation of mass and momentum. The conservation equations for a fluid are

$$\frac{D\rho^\star}{Dt} = -\rho^\star \nabla \cdot \mathbf{u}^\star \tag{4.1}$$

$$\rho^\star \frac{D\mathbf{u}^\star}{Dt} = -\nabla p^\star + \nabla . \tau^\star \tag{4.2}$$

$$\tau^\star_{ij} = \mu \left(\frac{\partial u^\star_i}{\partial x^\star_j} + \frac{\partial u^\star_j}{\partial x^\star_i} - \frac{2}{3}\delta_{ij} \frac{\partial u^\star_k}{\partial x^\star_k} \right) \tag{4.3}$$

where $\frac{D\diamond}{Dt} = \frac{\partial \diamond}{\partial t} + (u^\star \cdot \nabla)(\diamond)$ denotes the material derivative, $\rho^\star$ denotes the density, $u^\star$ the velocity, $p^\star$ the pressure, $\tau^\star$ the shear stress tensor with the elements $\tau^\star_{ij}$ and μ the viscosity. $x^\star_i$ are the components of the position, $u^\star_i$ the components of the velocity where Einstein's summation convention must be taken into account.

The system of differential equations Eq.(4.1)-(4.3) is closed with the equation of state for an ideal gas

$$p^\star = \rho R T^\star \tag{4.4}$$

where R is the specific gas constant and $T^\star$ the temperature. We assume the temperature $T^\star = T_0$ to be constant in space and time. The reference density is ρ_0.

The initial condition is described by a density and a velocity field. The inflow boundary involves a prescribed inlet velocity and a homogeneous Neumann boundary condition for the pressure. At the outlet, we consider a prescribed outlet pressure and a homogeneous Neumann boundary condition for the velocity.

4.3 Definition of Non-dimensional Numbers Characterizing the Flow

The Reynolds number of the flow is defined as

$$Re = \frac{\rho_0 U_0 d}{\mu}, \tag{4.5}$$

where ρ_0 is the characteristic density of fluid, U_0 the characteristic velocity, μ the dynamic viscosity. The characteristic length d is equal to the channel width or the cylinder/sphere diameter depending on the considered problem.

The Mach number M is the ratio of the characteristic velocity U_0 to the speed of sound c_0

$$M = \frac{U_0}{c_0} = \frac{U_0}{\sqrt{RT_0}} \tag{4.6}$$

The drag coefficient is an important characteristic for the energy loss of a flow that is commonly used to validate flow simulations. It is defined as

$$C_d = \frac{F_D}{0.5\rho U_0^2 A}, \tag{4.7}$$

where F_D, the drag force, is force acting on the body parallel to the main stream direction and A the reference area. Similar to the drag coefficient the lift coefficient is defined as

$$C_L = \frac{F_L}{0.5\rho U_0^2 A}, \tag{4.8}$$

where F_L, the lift force, is the force acting perpendicular to the main stream.

The Strouhal number is defined as the dimensionless frequency of the shedding vortices

$$St = \frac{fd}{U_0}, \tag{4.9}$$

where f is the vortex shedding frequency. This frequency can be obtained using the Fast Fourier Transform of the lift coefficient.

4.4 Nondimensional Governing Equations

The Navier-Stokes equations (cf. Section 4.2) can be expressed in non-dimensional Lagrangian form using the characteristic flow numbers presented in Section 4.3.

$$\frac{D\rho}{Dt} = -\rho\nabla \cdot u, \tag{4.10}$$

and

$$\rho\frac{Du}{Dt} = -\frac{1}{\mathrm{M}^2\gamma}\nabla p + \frac{1}{\mathrm{Re}}\nabla \cdot \tau, \tag{4.11}$$

where

$$p = \rho, \tag{4.12}$$

and the components of the stress tensor τ

$$\tau_{ij} = \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\delta_{ij}\frac{\partial u_k}{\partial x_k}. \tag{4.13}$$

$\rho, \mathbf{u}, p$ and τ are non-dimensional variables normalized by their characteristic values.

4.5 Particle Equations

The governing equations (4.10) and (4.11) are discretized using the rSPH approach [29].

In the present implementation, the summations of the rSPH approximation are resorted to form less summations and the kernel evaluations are replaced by pre-computed look-up tables in order to obtain a formulation that includes three simple look-up tables Λ_1, Λ_2, and Λ_3, thus

$$\begin{aligned}
\frac{dm_p}{dt} &= 0\\
\frac{dv_p}{dt} &= v_p \sum_q \Delta\mathbf{x}_{pq}\Delta\mathbf{u}_{pq}\Lambda_1 v_q\\
\frac{d\mathbf{u}_p}{dt} &= \frac{1}{\mathrm{M}^2\gamma}\sum_q \Delta\mathbf{x}_{pq}\Delta p_{pq}\Lambda_1 v_q + \frac{1}{3\mathrm{Re}}\sum_q \left[\Lambda_3 + \left(\Delta\mathbf{x}_{pq}\Delta\mathbf{x}_{pq}^T\right)\Lambda_2\right]\Delta\mathbf{u}_{pq}v_q.
\end{aligned} \tag{4.14}$$

The look-up tables Λ_i are sampled at the distance $||\Delta\mathbf{x}_{pq}||$ between particles p and q, where $\Delta\mathbf{x}_{pq}$ denotes the vector of the Cartesian distance between the particles, $\Delta\mathbf{u}_{pq}$ the vector of the velocity difference, and Δp_{pq} the pressure difference. The look-up table values are pre-computed as

$$\Lambda_1 = \frac{1}{||\Delta\mathbf{x}_{pq}||h}\left.\frac{dW(r,h)}{dr}\right|_{r=||\Delta\mathbf{x}_{pq}||}, \tag{4.15}$$

$$\Lambda_2 = -\frac{1}{||\Delta\mathbf{x}_{pq}||^3 h}\left.\frac{dW(r,h)}{dr}\right|_{r=||\Delta\mathbf{x}_{pq}||} - \frac{1}{||\Delta\mathbf{x}_{pq}||^3 h}\left.\frac{d^2W(r,h)}{dr^2}\right|_{r=||\Delta\mathbf{x}_{pq}||} \tag{4.16}$$

$$\Lambda_3 = 10\Lambda_1 + ||\Delta\mathbf{x}_{pq}||^2\Lambda_2, \tag{4.17}$$

where $W(r,h)$ is chosen to be the quartic spline kernel M_5 [29]. This novel formulation of rSPH has several advantages compared to the explicit calculation of the individual components. Firstly, the evaluation of the right-hand side is reduced from originally five to two summations. Secondly, the kernel evaluations in the summations are avoided by using the look-up tables.

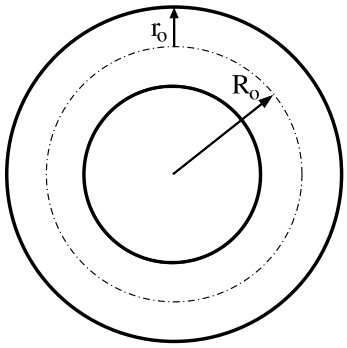


Figure 4.1: Initial vortex ring. The ring torus has a mean radius of R_0 and a tube radius of r_0.

4.6 Compressible Vortex Ring

For the compressible vortex ring, we use $\mathrm{M} = 0.5$, Re$= \Gamma\rho_0/\mu = 3000$, and a computational domain of size $2 \times 1 \times 1$. The initial vortex ring (cf. Fig. 4.1) is assumed to have a Gaussian distribution of vorticity $\omega = \Gamma/(\pi r_0)\exp(-r^2/r_0^2)$, where $\Gamma = 0.3$, r is the distance to the core of the tube, and $r_0 = 0.025$ the tube radius. The ring radius R is perturbed around a mean value of $R_0 = 0.125$ by a truncated Fourier series of amplitude $9 \cdot 10^{-4}$ [150]

$$R = R_0 + \sigma g(\theta) \tag{4.18}$$

$$g(\theta) = \sum_{k=1}^{32} \sin(k(\theta + 2\pi r v_1)) + \cos(k(\theta + 2\pi r v_2)) \tag{4.19}$$

whew θ is the angular position, $\sigma = 0.0009$, and rv_1 and rv_2 are random variables in $[0, 1[$.

For the initialization of the velocity field, we assume that the flow is incompressible with an initial unit density field. The domain is periodic in all dimensions.

The particles are reinitialized (remeshed) after each time step using the M_4' kernel function [106]. Time integration is done with a second-order Runge-Kutta scheme.

4.7 Results

The result of the vortex ring simulation using 33 million particles distributed onto 16 AMD Opteron processors is shown in Figs. 4.2 and 4.4. We use a constant time step of $5 \cdot 10^{-4}$, corresponding to a maximum CFL number of 0.5. The velocity profile of the vortex ring creates a density profile that has its minimum at the core of the ring. The density field evolves to create an accumulation of mass around the ring, resulting in pressure waves that travel through the system, re-entering through the periodic boundary downstream of the vortex ring. Interferences with the vortex ring create additional pressure waves that decay over time (Fig. 4.2). Iso-surfaces of vorticity at corresponding times are shown in Fig. 4.4 and the velocity distribution in Fig. 4.3.

The propagation speed of the compressible vortex ring is 0.48, which is within 4% of the analytical solution [137], including corrections for compressibility [112].

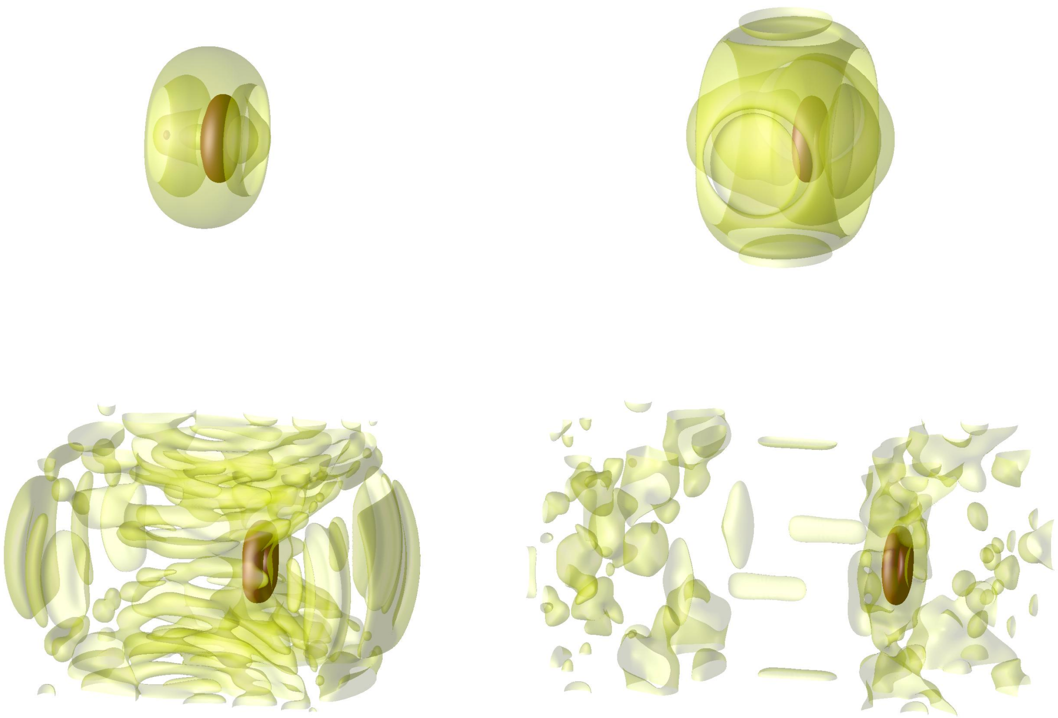


Figure 4.2: Simulation of a compressible vortex ring. Iso-surfaces of the density field for ρ=0.900, 0.990, and 1.015 at times t=0.15, 0.25, 0.50, and 1.00. The field is discretized using 33 million particles. Acoustic pressure waves propagating from the ring can be seen as lightly shaded surfaces. The darkest iso-surface of the density field indicates the position of the vortex ring.

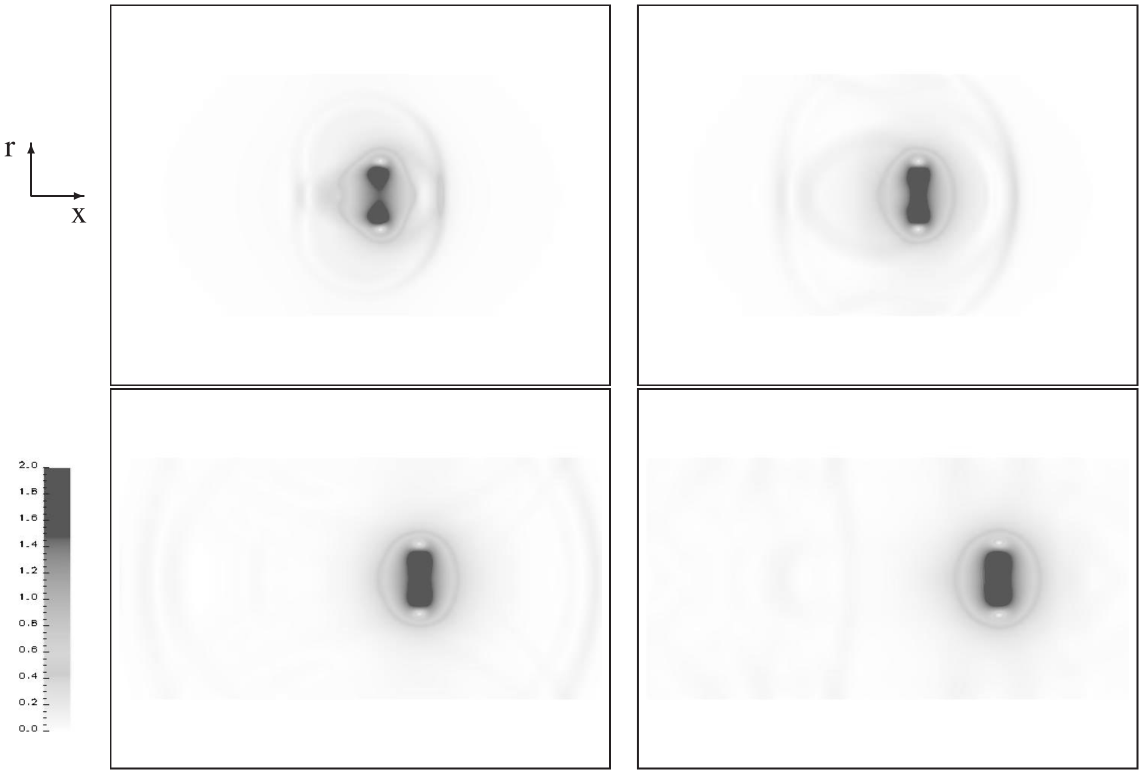


Figure 4.3: Simulation of a compressible vortex ring. Middle cross-sections of the velocity magnitude at t=0.15, 0.25, 0.50, and 1.00.

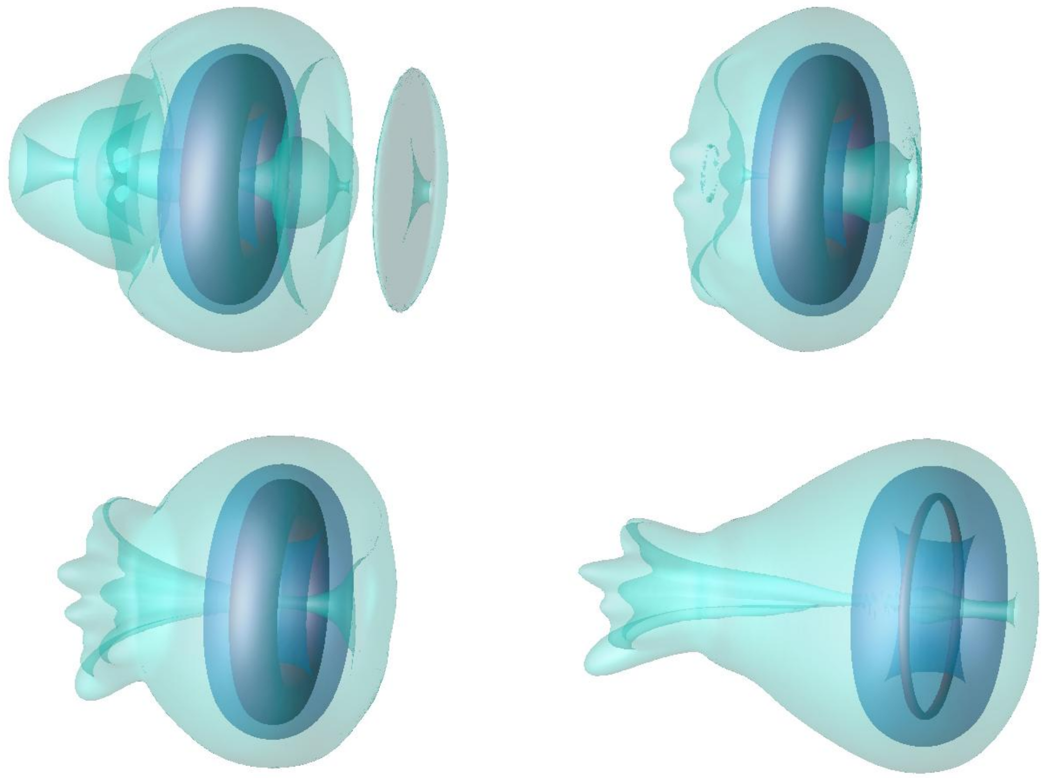


Figure 4.4: Simulation of a compressible vortex ring. Iso-surfaces of vorticity for $|\omega|$ = 40, 10, and 0.2 at t=0.15, 0.25, 0.50, and 1.00.

4.8 Note on the Error Analysis Performed by Chaniotis *et al.* [29]

In this section, we comment on the error evaluation presented in the journal article "Remeshed Smoothed Particle Hydrodynamics for the Simulation of Viscous and Heat Conducting Flows" [29] of Chaniotis *et al.* . The error analysis is based on the simulation of the Taylor-Green vortex flow where the analytical solution is known. Chaniotis *et al.* describe to perform the error analysis using the relative L_∞ error

$$L_\infty = \max_{t\in[0,T_{max}]} \left(\left| \frac{u^t_{ex} - u^t_{SPH}}{u^t_{ex}} \right| \right), \tag{4.20}$$

where u^t_{ex} denotes the maximum velocity of the exact incompressible solution at time t and u^t_{SPH} the maximum velocity of the SPH simulation at time t.

We failed to reproduce the errors reported in Fig. 2 for the Taylor-Green flow at $Re = 1$ by using the described rSPH method. A re-implementation of the method based on the paper indicates that the relative errors are significantly larger. A refinement study shows close to second-order convergence as expected for the use of a second order kernel (Tab. 4.8). Discussions with the authors of the paper revealed that the error analysis (Eq.(4.20)) was not applied correctly in the author's implementation. The resulting rela-

Table 4.1: Numerical Error of rSPH

Particle Spacing	Chaniotis *et al.*	Present study	Rescaled by $u_{max}(0)$	Order
1/20	4.5%	28.5%	0.5%	
1/30	3.9%	12.8%	0.25%	1.97
1/60	-	3.6%	0.08%	1.83

tive errors (Tab. 4.8) appear to be unusually high on a first glance but they are reasonable when taking into account that the analytical solution approaches zero in this problem, where round off errors start to become important. The error corresponds to the maximal error of the maximal velocity within the time interval [0,0.05] where the analytical maximal velocity decays from 0.04 to 0.00077. The absolute error appears in the final time step in all cases and is for example for $h = 1/60$ equals to $3 \cdot 10^{-5}$. By scaling the error in by the initial velocity we obtain a more meaningful measurement of the performance.

4.9 Remeshed SPH versus Particle-Mesh Hydrodynamics

In the context of hybrid particle-mesh methods, Chatelain *et al.* [31] presented the Particle-Mesh Hydrodynamics (PMH), a computationally efficient particle-mesh method based previous work of Cottet [41, 97]. PMH is a limiting case of rSPH where remeshing is applied at every time step and has been tested on the Euler equation for a compressible, inviscid and adiabatic flow.

In rSPH the required frequency of remeshing depends on the level of particle distortion during a simulation. The frequency is chosen as low as possible because the remeshing scheme may lead to a smearing effects in areas of high quantity gradients. The rSPH method employs smooth kernels to evaluate the differential operators at the particle locations (Eq.(2.20)) which requires neighbor searches and summations.

In PMH, the particles are re-initialized every time step which can lead to numerical diffusion but enables the differentiation to be performed computationally more efficient by using finite differences because the particles reside on the positions of grid nodes after remeshing. In the PMH formulation, the particles solely handle the convective part of the governing equations. The particle quantities are then interpolated onto a mesh, where the pressure terms are computed. PMH, like SPH, is free of the convection CFL condition while at the same time it is more efficient as derivatives are computed on a mesh rather than particle-particle interactions.

Chapter 5

Particle Simulation of Elastic Solids

5.1 Introduction

The simulation of soft tissue is one of the key components of virtual reality systems with applications ranging from video games to virtual surgery environments [161, 160]. An important aspect of these simulations is the physics-based modeling of linear and non-linear elastic solids undergoing large deformations. A number of computational techniques have been employed in the past in order to address this problem including finite element [48, 132, 105], finite differences [164], mass-spring models [165, 167] and particle methods [130, 131, 108, 67, 72]. Grid based methods such as finite elements have been shown to be efficient and robust in simulations of systems undergoing small or medium deformations [48], whereas meshless/particle methods are advocated for the simulations of solids undergoing excessive deformation and mechanical splitting [130]. The adaptivity and grid-free character of meshless methods have enabled a number of unprecedented simulation is biomechanics [52]. We note however that, even though meshless methods are considered capable of handling large deformations, one often introduces empirical techniques in order to maintain the accuracy of the method. This problem may be further aggravated when convection terms may further distort the computational elements, as in simulations of fluid mechanics (a problem with extreme deformations) problems. Smooth Particle Hydrodynamics (SPH) has been proposed as a meshless method capable of resolving problems with large deformations for both solids and fluids. In SPH the continuum properties are discretized on smooth particles, the stress-strain governing equations are formulated in a Lagrangian frame and the derivatives are computed by taking the derivatives of the particle kernels. The simulations of solids using SPH [130] suffer however from the well known tensile instability. The problem of tensile instability arises when the distance of particles is small under positive pressure and was attributed to the shape of the second derivative of the interpolating kernel [159]. The forces become attractive due to the shape of the derivative approximation resulting in large numerical errors. Methods presented in literature to resolve this problem are the use of an alternative smoothing kernel [89], the introduction of stress points [131] or artificial stress [108, 67]. This pathology of SPH is related to the fact that smooth particle approximations are inaccurate when the particle distribution is excessively distorted [29, 41].

Linear models are well suited to mimic small deformation of elastic material because the stress-strain relationship can be approximated by a linear function. Several works [130, 67, 108] have employed particle models for linear elasticity. Several materials in nature exhibit a nonlinear behavior for stretch levels exceeding 10%. Nonlinear elasticity problems have been simulated in an Eulerian framework using the the finite element method (FEM)[132]. They are based on the formulation of the deformation gradient in terms of the current material position with respect to its reference position. The extension of this approach to Lagrangian particle methods is hindered by the fact that when the material properties are tracked in a moving framework the reference position becomes irrelevant. In order to circumvent this difficulty we replace the direct evaluation of the deformation gradient by its temporal evolution. The resulting formulation enables a straightforward and efficient implementation using particle methods. To the best of our knowledge, particle models for nonlinear elasticity have not been presented in the literature.

In this chapter, we propose a particle method that relies on remeshing to ensure the accuracy of the simulations. Particles are convected in a Lagrangian framework, followed by a regularization of their locations and the corresponding projection of the particle properties. This approach enables accurate simulations of solids undergoing large deformations and eliminates spurious effects such as the tensile instability. In addition, this remeshing procedure accommodates the generalisation of stress-strain relations in a Lagrangian framework and the extension of particle methods to problems of non-linear elasticity. We compare the present particle methods with the finite element solver (ABAQUS 6.4/EXPLICIT) in benchmark problems involving linear and nonlinear elastic solids. We examine the accuracy of the method in a three-dimensional test problem with periodic boundary conditions and demonstrate its capability to simulate large deformations in a plane strain compression test. Finally, we validate our particle solver by simulating liver tissue exposed to an aspiration test [115, 116].

5.2 Governing Equations

We consider two different approaches to model elastic material: a linear and nonlinear model. Linear models are well suited to mimic small deformation of elastic material because the stress-strain relationship can be approximated by a linear function. Exceeding a nominal stretch of 10% most material in nature reveal a nonlinear behavior. Several researchers ([130],[67],[108]) have studied the particle model for linear elasticity. Particle models for nonlinear elasticity has not been presented in literature to the best of our knowledge.

5.2.1 Linear Elastic Model

We consider the conservation of mass and momentum in Lagrangian form:

$$\frac{D\rho}{Dt} = -\rho\nabla\cdot\mathbf{u}, \tag{5.1}$$

$$\rho\frac{D\mathbf{u}}{Dt} = \nabla.\sigma = \nabla.\left(-p\mathbf{I}+\mathbf{S}\right), \tag{5.2}$$

where ρ is the density, $\mathbf{u}$ the velocity of the material and $\frac{D}{Dt}$ is the material derivative. The stress tensor σ can be splitted into a pressure part $-p\mathbf{I}$ and a deviatoric part $\mathbf{S}$. Assuming

Hooke's law the deviatoric part $\mathbf{S}$ evolves componentwise as follows [67]

$$\frac{DS_{ij}}{Dt} = 2\mu_s\left(\dot{\epsilon}_{ij} - \frac{1}{3}\delta_{ij}\dot{\epsilon}_{ij}\right) + S_{ij}\Omega_{jk} + \Omega_{ik}S_{kj}, \tag{5.3}$$

$$\dot{\epsilon}_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right), \tag{5.4}$$

$$\Omega_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} - \frac{\partial u_j}{\partial x_i}\right), \tag{5.5}$$

where μ_s is a shear modulus, ϵ the strain rate, and Ω the rotation rate. The pressure p is determined by an equation of state

$$p = c_0^2(\rho - \rho_0), \tag{5.6}$$

where c_0 is the speed of sound and ρ_0 is the reference density. We obtain the speed of sound c_0 and the shear modulus μ_s from the Young's modulus E and the Poisson ratio ν by

$$c_0 = \sqrt{\frac{E}{3(1-2\nu)\rho_0}}, \tag{5.7}$$

$$\mu_s = \frac{E}{2(1+\nu)}. \tag{5.8}$$

5.2.2 Nonlinear Elastic Model

In this context, we consider a novel particle model for nonlinear elasticity. The continuity and momentum equation have the same form as in the linear case (Eq. 5.1 and 5.2). The stress tensor, however, is defined by a nonlinear relationship. Without loss of generality we focus on the constitutive law of isotropic compressible hyperelastic material in this study. The hyperelastic solid model is an established nonlinear model to describe soft biological tissue [62, 115, 116] originally developed for rubber-like material [85].

Hyperelastic material is characterized by of a strain-energy function $U(\mathbf{F})$ that is only a function of the deformation gradient $\mathbf{F}$ [85]. The deformation gradient $\mathbf{F}$ is defined by

$$\mathbf{F}(\mathbf{X}, t) = \frac{\partial \mathbf{x}(\mathbf{X}, t)}{\partial \mathbf{X}} \tag{5.9}$$

where $\mathbf{x}$ is the current position and $\mathbf{X}$ the reference position of the material. When the material resides in the reference position it is undeformed and stress-free ($\mathbf{x} = \mathbf{X} \Rightarrow \mathbf{F} = \mathbf{I}$). The deformation gradient can be considered as the Jacobian matrix of the mapping $\mathbf{X} \mapsto \mathbf{x}(\mathbf{X})$. Its determinant $J = |\mathbf{F}|$ corresponds to the local volume change with respect to the reference position. Numerical schemes, such as FEM, evaluate the deformation gradient directly using Eq.(5.9) in the reference frame.

In the Lagrangian framework, however, we consider the evolution of the deformation gradient in the following form

$$\frac{D\mathbf{F}}{Dt} = \frac{D\frac{\partial \mathbf{x}}{\partial \mathbf{X}}}{Dt} = \frac{\partial \frac{D\mathbf{x}}{Dt}}{\partial \mathbf{X}} = \frac{\partial u}{\partial \mathbf{X}} = \frac{\partial u}{\partial \mathbf{x}}\frac{\partial \mathbf{x}}{\partial \mathbf{X}} \tag{5.10}$$

$$\Rightarrow \frac{D\mathbf{F}}{Dt} = \frac{\partial \mathbf{u}}{\partial \mathbf{x}}\mathbf{F} \tag{5.11}$$

A key aspect of Eq.(5.11) is that the reference position $\mathbf{X}$ is absent and its knowledge is unnecessary during the evolution.

We consider the strain-energy function as reduced polynomial of order N as presented by Nava *et al.* [115] to characterize soft biological tissue

$$U(\bar{I}_1, J) = \sum_{n=1}^{N} C_{n0}\left(\bar{I}_1 - 3\right)^n + \frac{1}{D}\left(J-1\right)^2 \tag{5.12}$$

where $J = |F|$ is the volume change and $\bar{I}_1 = trace\left(\bar{\mathbf{B}}\right)$ is the normalized first strain invariant of the normalized left Cauchy Green strain tensor $\bar{\mathbf{B}}(\mathbf{F})$, C_{n0} denotes the polynomial coefficient and D the volume coefficient. The normalized left Cauchy Green strain tensor $\bar{\mathbf{B}}$ is expressed by

$$\bar{\mathbf{B}} = J^{-\frac{1}{3}}\mathbf{F}\mathbf{F}^T \tag{5.13}$$

The symmetric Cauchy stress tensor used in Eq.(5.2) is derived from the strain-energy function U by the relationship [85]

$$\sigma = J^{-1}\frac{\partial U}{\partial \mathbf{F}}\mathbf{F}^T. \tag{5.14}$$

Thus, the pressure p and the deviatoric stress $\mathbf{S}$ are evaluated by the following equations

[1]

$$p = -\frac{\partial U}{\partial J} = -\frac{2}{D}(J-1) \tag{5.15}$$

$$\mathbf{S} = \frac{2}{J} DEV\left[\frac{\partial U}{\partial \bar{I}_1}\bar{\mathbf{B}}\right] = \frac{2}{J} DEV\left[\sum_{n=1}^{N} nC_{n0}\left(\bar{I}_1 - 3\right)^{n-1}\bar{\mathbf{B}}\right] \tag{5.16}$$

where $DEV[\diamond]$ represents the deviatoric part of $\diamond$.

5.2.3 Initial and Boundary Conditions

The initial condition consists of a density and a velocity field distribution. The initial deviatoric part of the stress tensor is assumed to be $\mathbf{S}(\mathbf{x}, 0) = \mathbf{S}_0(\mathbf{x})$ in the linear case, the initial deformation gradient $\mathbf{F}(\mathbf{x}, 0) = \mathbf{F}_0(\mathbf{x})$ in the nonlinear case.

We consider two kinds of boundary conditions

- Free surface or stress-free boundary. The stress tensor σ at the boundary is such that the surface is traction free

  $$\sigma \cdot \mathbf{n} = 0, \tag{5.17}$$

  where $\mathbf{n}$ is the surface normal.

- Fixed or no-slip boundary. The velocity at the boundary is prescribed in advance, when for example the material is attached to a wall.

5.3 Particle Equations

The particle position $\mathbf{x}_p$, mass m_p, volume v_p, and velocity $\mathbf{u}_p$ evolve by the following system of ordinary differential equations:

$$\begin{aligned}
\frac{d\mathbf{x}_p}{dt} &= \mathbf{u}_p, \\
\frac{dm_p}{dt} &= 0, \\
\frac{dv_p}{dt} &= \langle \nabla \cdot \mathbf{u} \rangle_p v_p, \\
\frac{d\mathbf{u}_p}{dt} &= \frac{v_p}{m_p}\left(-\langle \nabla p \rangle_p + \langle \nabla . \mathbf{S} \rangle_p\right),
\end{aligned} \tag{5.18}$$

where $\langle \diamond \rangle_p$ represents the mollified derivative approximation of $\diamond$ based on Eq.(2.20). The evaluation of the pressure p and the deviatoric stress $\mathbf{S}$ depends on the constitutive model of the elastic material.

The surface of the elastic body is described using the Particle Level Set Method [82, 81] for visualization purpose and for the evaluation of the surface normal. The level set function represents the signed distance function to the interface. The particles carry the level set information as a scalar attribute Φ_p that remains constant during the time integration:

$$\frac{d\Phi_p}{dt} = 0 \tag{5.19}$$

We reinitialize the level set value every 20th time step to ensure the signed distance property.

5.3.1 Linear Elastic Model

The pressure p_p and the evolution of the deviatoric stress $\mathbf{S}_p$ on particle p for the linear elastic solid is expressed by

$$\begin{aligned} p_p &= c_0^2\left(\frac{m_p}{v_p} - \rho_0\right) \\ \frac{dS_{ij,p}}{dt} &= 2\mu\left(\dot{\epsilon}_{ij,p} - \frac{1}{3}\delta_{ij}\dot{\epsilon}_{ij,p}\right) + S_{ij,p}\Omega_{jk,p} + \Omega_{ik,p}S_{kj,p} \\ \dot{\epsilon}_{ij,p} &= \frac{1}{2}\left(\left\langle \frac{\partial u_i}{\partial x_j} \right\rangle_p + \left\langle \frac{\partial u_j}{\partial x_i} \right\rangle_p\right) \\ \Omega_{ij,p} &= \frac{1}{2}\left(\left\langle \frac{\partial u^i}{\partial x_j} \right\rangle_p - \left\langle \frac{\partial u^j}{\partial x_i} \right\rangle_p\right) \end{aligned} \tag{5.20}$$

where $\dot{\epsilon}_p$ is the strain rate and Ω_p the rotation rate of particle p.

5.3.2 Hyperelastic Model

The deformation gradient $\mathbf{F}_p$ on particle p evolves by

$$\frac{d\mathbf{F}_p}{dt} = \left\langle \frac{\partial \mathbf{u}}{\partial \mathbf{x}} \right\rangle_p \mathbf{F}_p. \tag{5.21}$$

The volume change J_p, the normalized left Cauchy Green strain tensor $\bar{\mathbf{B}}_p$, the normalized first invariant $\bar{I}_{1,p}$ of a particle p are evaluated by

$$
\begin{aligned}
J_p &= |\mathbf{F}_p|, \\
\bar{\mathbf{B}}_p &= J_p^{-\frac{1}{3}} \mathbf{F}_p \mathbf{F}_p^T, \\
\bar{I}_{1,p} &= trace(\bar{\mathbf{B}}_p).
\end{aligned}
$$

Finally, the pressure p_p and the deviatoric stress $\mathbf{S}_p$ can be expressed by

$$
\begin{aligned}
p_p &= -\frac{2}{D}\left(J_p - 1\right), \\
\mathbf{S}_p &= \frac{2}{J_p} DEV \left[\sum_{n=1}^{N} n C_{n0} \left(\bar{I}_{1,p} - 3 \right) \bar{\mathbf{B}}_p \right].
\end{aligned}
\tag{5.22}
$$

The terms are based on the constitutive model of a hyperelastic material in reduced polynomial form (Eq.(5.15) and (5.16)).

5.3.3 Boundary Conditions

The boundary conditions are imposed by use of image particles that have similar physical properties as the solid particles. The boundary particles interact with the solid particle such that the boundary conditions are satisfied. At a free surface, the image particles are adjusted to satisfy the stress-free boundary condition. The stress tensors of the image particles are projected such that the surface is traction free [131]. The surface normal used for the projection is evaluated based on the particle level set method [81]. At a fixed boundary, the image particles enforce a prescribed velocity of the boundary whereas the remaining attributes satisfy a homogeneous Neumann condition.

5.4 Accuracy

We consider a box of elastic material in 3D with periodic boundary conditions in all directions. The initial velocity is a three dimensional sinusoidal function $(u(x, 0) =$

$\sin(2\pi x)\sin(2\pi y)\sin(2\pi z))$ that leads to periodic oscillations in the material. To determine the accuracy of the particle method we measure the maximum error (L_∞) of the velocity profile by comparison with the analytic solution after one period. The Young's Modulus is $E = 100$ and the Poisson ratio is set to $\nu = 0.3$ in the linear case. The polynomial coefficient is $C_{10} = 10$ and the volume coefficient $D = 1$ in the nonlinear case. The Finite Element simulation are performed with ABAQUS 6.4/EXPLICIT using the explicit solver.

Fig. 5.1 shows the L_∞-error over the inverse of the particle distance h. The particle solution converges with second order in the L_∞ - error in both the linear and nonlinear case. The errors are comparable with the FEM solution including the same number of computational elements.

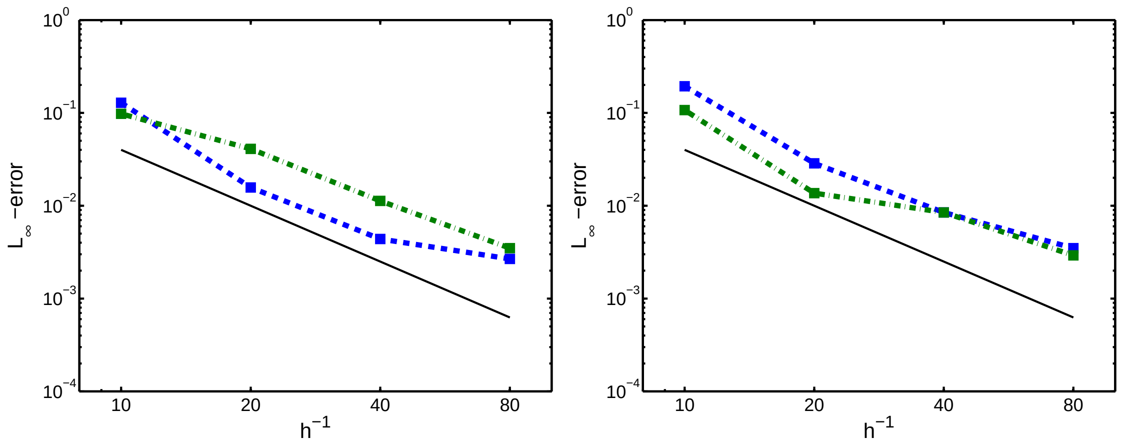


Figure 5.1: L_∞ - error of the particle solution (dashed line) and the FEM solution (dash-dotted line) in linear (left) and nonlinear (right) case compared to 2nd order scaling (solid line)

5.5 Plane Strain Compression Test

To show the performance of the solid particle solver we compare the results of plane strain compression test with an FEM solution provided by ABAQUS 6.4/EXPLICIT for linear and nonlinear elasticity. The elastic material is initially undeformed and has a rectangular shape of size 1×2 (Fig. 5.2). The horizontal faces move with a constant vertical velocity $u_{piston} = 25$ whereas the vertical faces represent free surfaces. The vertical compression of the material leads to a horizontal expansion. The thickness of the the piston is neglected. This problem exposes a singularity at the corners of the elastic material where the material is forced to rotate for $\approx 90°$.

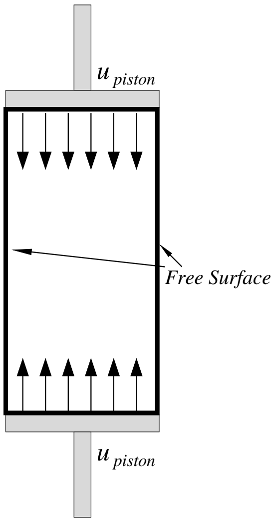


Figure 5.2: Plane strain compression test. Elastic material in rectangular shape is compressed by two imaginary pistons moving with a constant velocity u_{piston}.

The simulations are performed with a initial density that is equal the reference density $\rho_0 = 1$ in the presented cases. The time integration scheme is Runge Kutta 4th order with a constant time step of $\Delta\, t = 5 \cdot 10^{-5}$.

5.5.1 Linear Elastic Model

The elastic material has a Young's Modulus of $E = 100$ and a Poisson ratio of $\nu = 0.49$. This value of Poisson ratio results in to a behavior of a nearly incompressible solid.

Fig. 5.3 shows the evolution of the elastic material based on the presented particle solver compared to an FEM solution. The two solutions mainly differ in their behavior near the numerical singularity at the corner of the piston. The FEM solver reveals significant artifacts near the singularity resulting in nodes crossing the ambient piston whereas the particle solution is still physically plausible. Nodes of the FEM grid pass the border of the piston causing nonphysical structures in the vicinity of the singularity.

Even, refinement of the FEM grid cannot resolve this problem as shown in Fig. 5.4. In contrary, a refinement to a grid containing 8000 cells leads to an abort of the simulation at $t = 0.0263$ because the deformation speed becomes larger than the wave speed at nodes in the vicinity of the corner.

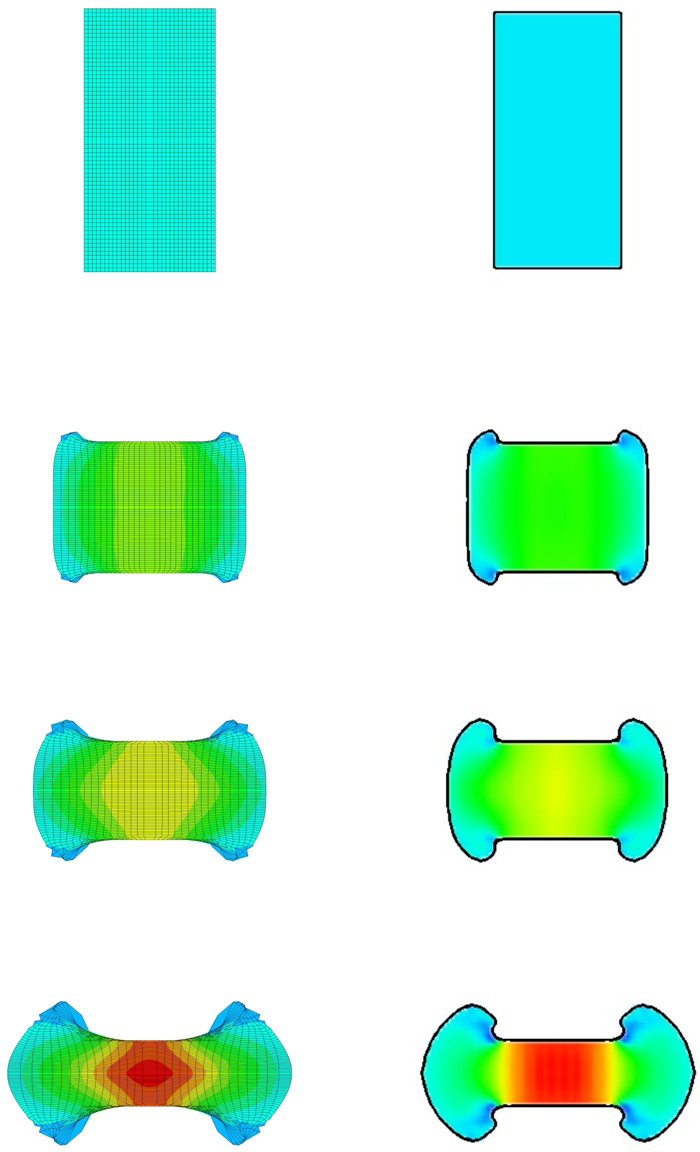


Figure 5.3: Plain strain compression test with linear elasticity. The Finite Element solution (left) is compared to the particle solution (right) at $t = 0.0, 0.02, 0.025, 0.03$ *using the same initial resolution (*≈ 2000 *computational elements). The color represents the pressure distribution.*

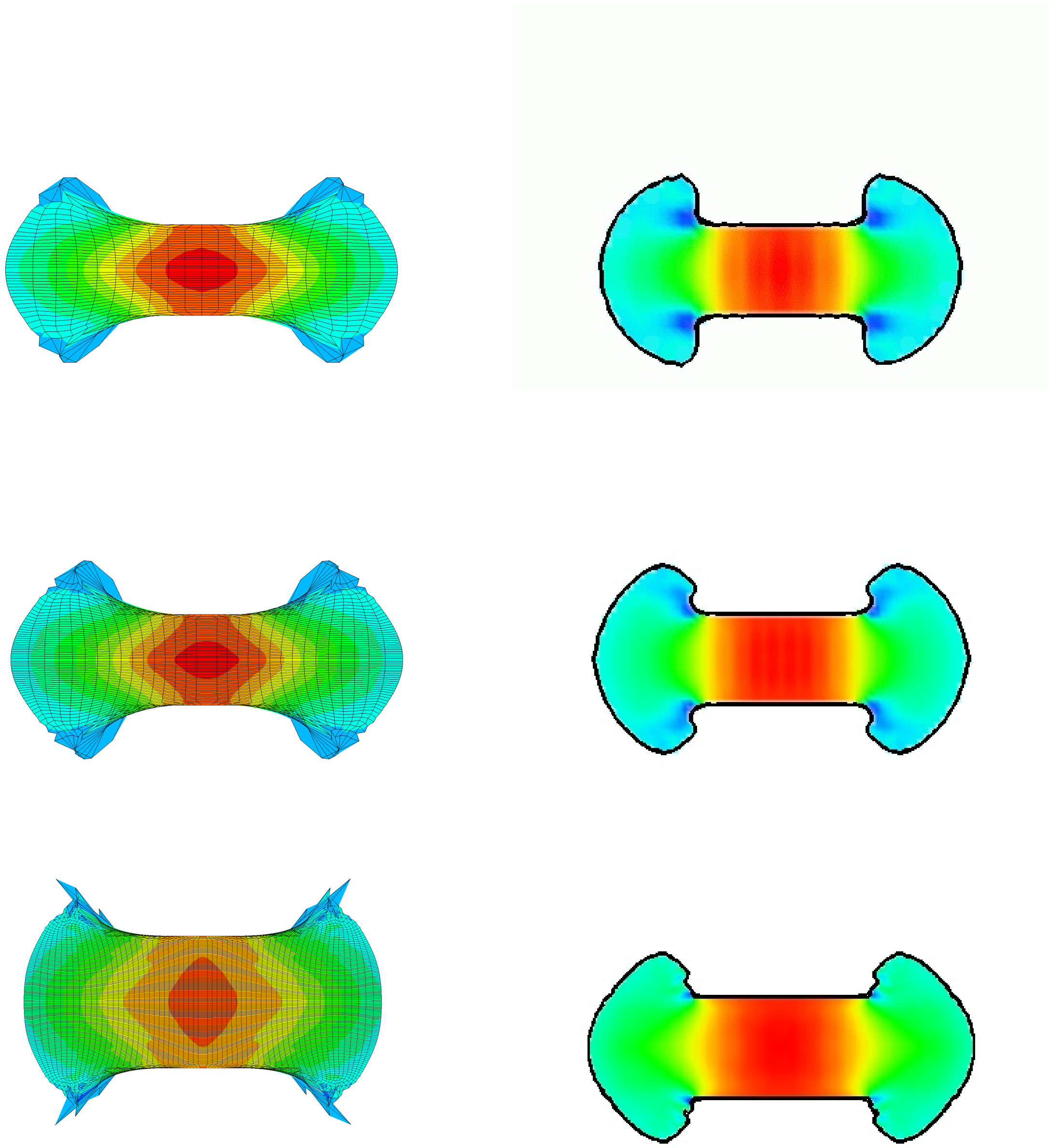


Figure 5.4: Plane strain compression test with linear elasticity. Effect of the resolution on the particle solution at $t = 0.03$ *using* $\approx 800, \approx 2000$ *and* ≈ 8000 *nodes/particles . The color represents the pressure distribution.*

Table 5.1: Maximum horizontal displacement at $t = 0.02$

Particles/Nodes ($t = 0$)	Particle Method	Order	FEM
512	0.138	-	0.230
2028	0.192	1.2	0.232
8192	0.228	2.6	0.232
32768	0.233	1.7	-
131072	0.235	1.8	-

Table 5.1 illustrate the effect of the resolution on the maximum horizontal displacement for both methods. The maximum horizontal displacement resides on the surface of the elastic solid. The particle solution converges more than linearly and agrees well with the FEM results. The FEM solution is nearly converged at a low resolution whereas the particle method requires a high resolution to reach convergence. This result might be explained by the diffusive effect of the remeshing and the level set reinitialization. Particularily, the first-order reinitialization scheme of the level set function can shift the surface significantly as shown by Hieber *et al.* [81].

5.5.2 Hyperelastic Model

To validate the particle solver for hyperelastic solid we consider the strain-energy relationship as an expansion series (Eq.(5.12)) with $N = 1$ and $C_{10} = 2.2$ and $D = 0.001$.

Fig. 5.5 shows the simulation results of the FEM and the particle solver at $t = 0.025$. The FEM solution remains stable for all resolutions and proves that the nonlinear model is suitable for large deformations. The solution, however, still reveals severe numerical artifacts in the vicinity of the singularity, even at a high resolution. The large rotation and compression of the material at the singularity causes large errors in FEM solution because the stresses are evaluated int the undeformed reference frame and mapped to the deformed Lagrangian frame. Contrarily, the Lagrangian particle solver can handle rotations easily because the computational elements follow the material making mapping of quantities unnecessary.

Table 5.2 shows the effect of the resolution on maximal horizontal displacement of both methods. The FEM solution and the particle solution converge to a displacement with a deviation of 2%. Similarly to the linear case (Table 5.1), the FEM solver reaches a converged solution at a lower resolution than the particle solver. Again, we see the origin of this discrepancy in the remeshing of the particles and reinitialization of the level set function.

Table 5.2: Maximal horizontal displacement at $t = 0.02$

Particles/Nodes ($t = 0$)	Particle Method	Order	FEM
512	0.143	-	0.249
2028	0.200	1.0	0.250
8192	0.237	1.6	0.250
32768	0.250	1.8	-
131072	0.254	1.8	-

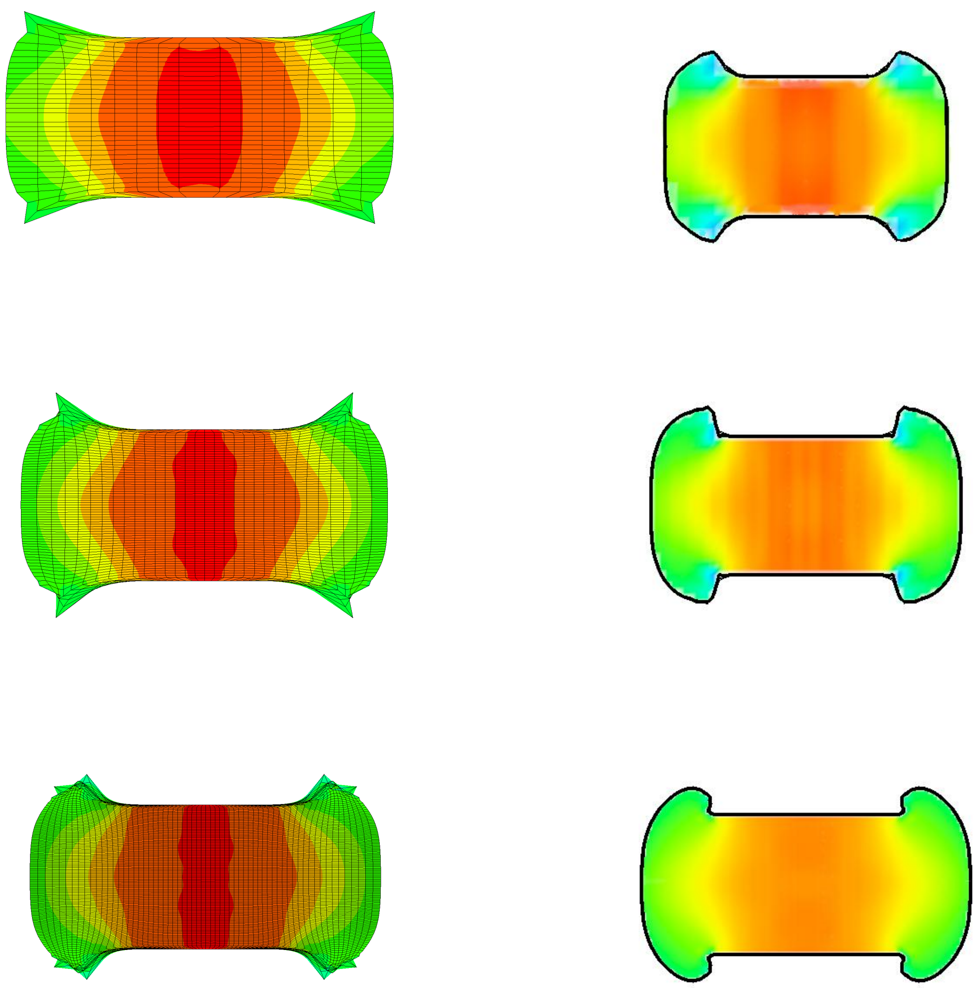


Figure 5.5: Plane Strain Compression test with nonlinear elasticity. Effect of the resolution on the particle solution at $t = 0.025$ using $\approx 500, \approx 2000$ and ≈ 8000 nodes/particles. The color represents the pressure distribution.

5.6 Simulation of an Aspiration Test on Liver Tissue

To demonstrate the performance of particle solver on a real world problem we try to reproduce the liver tissue behavior during an aspiration test [115, 116] using our particle solver. The aspiration test is performed by pressing a tube against the liver tissue and creating a vacuum inside the tube such that the tissue is sucked into the aspiration hole (Fig. 5.6 and 5.7). The material properties can be determined based on the tissue deformation by reverse engineering.

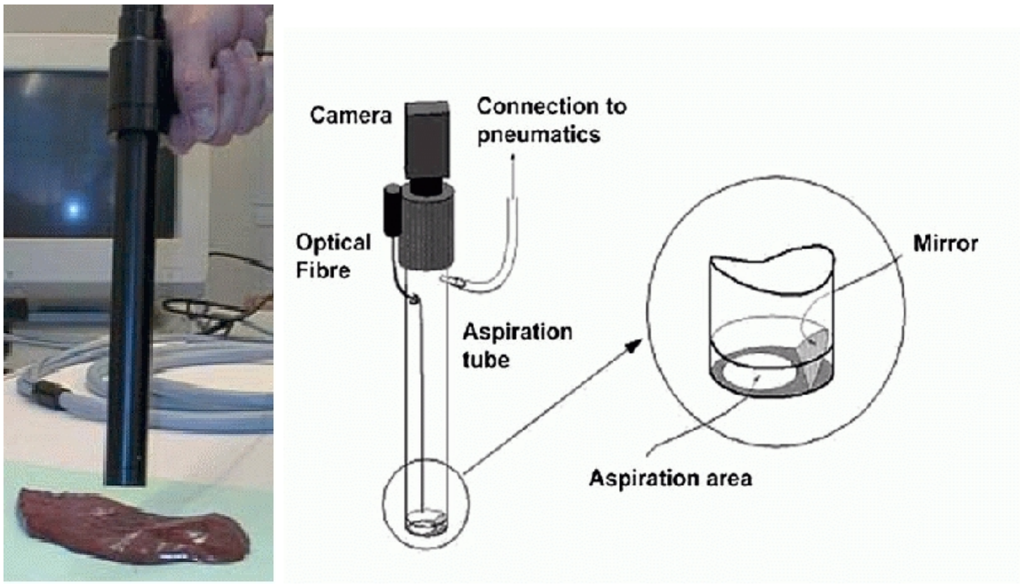


Figure 5.6: Aspiration test. The test device (left) and a schematic description (right) of the aspiration test (in courtesy of Nava et al. [116])

We choose the hyperelastic liver model of Nava *et al.* [115] that describes the strain-energy relationship in the reduced polynomial form of order $N = 5$ (Eq.(5.12)). The viscoelasticity of the tissue is modeled by time dependent relaxation coefficients in form of a Prony series of order $K = 4$ (cf. Appendix D, especially Eq.(D.2)).

We simplify the viscosity model by considering bulk viscosity only and restricting the Prony series to first order. The volume coefficient determines the compressibility of the material and is chosen to a very small value ($D = 5e^{-8}1/Pa$) to mimic incompressible behavior. We solve the viscosity model by integrating an additional ODE following the methodology of particle methods. The derivation of the ODE of first order from Eq.(5.15) is shown in the appendix D.

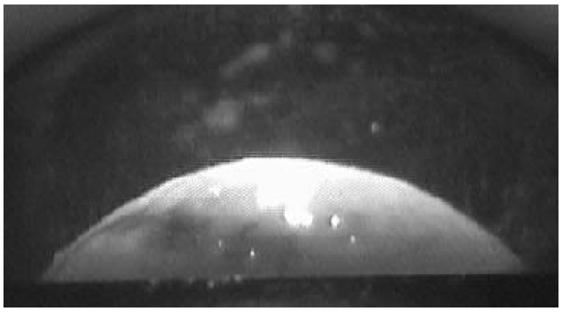


Figure 5.7: Aspiration test. Typical image showing the exposed tissue bubble during the aspiration test (in courtesy of Nava et al. [116])

The size of the domain is $3cm \times 3cm \times 3cm$ containing approximately 100000 particles. The tissue covers the domain up to a height of $2.5cm$. The aspiration hole with a diameter of $1cm$ is placed at the center of the top face. The time integrator is an explicit Runge Kutta scheme of 4th order with a time step of 0.0005. A no-slip boundary conditions is imposed to the lateral faces of the tissue and the area surrounding the aspiration hole. The pressure inside the aspiration tube $p_{tube} = p_\infty(1 - e^{-2t/s})$ decreases with time approaching a limit of $p_\infty = -300mbar$.

Fig. 5.8 shows a representative cross-section of the domain through the center of the bubble at time $t = 15$. It is colored according to the stretch distribution. The value of the stretch is small, generally less than 5%. The non-zero stretch is concentrated in the vicinity of the aspiration hole indicating that the domain size is sufficiently large. The stretch is positive where tissue is sucked into the tube and negative where the tissue is pressed against the aspiration device.

The time history of the bubble displacement is shown in Fig. 5.9. The bubble displacement is monitored on the tip of the bubble in both the simulation and the experiment. The comparison shows very good agreement between the particle solution and the experimental results.

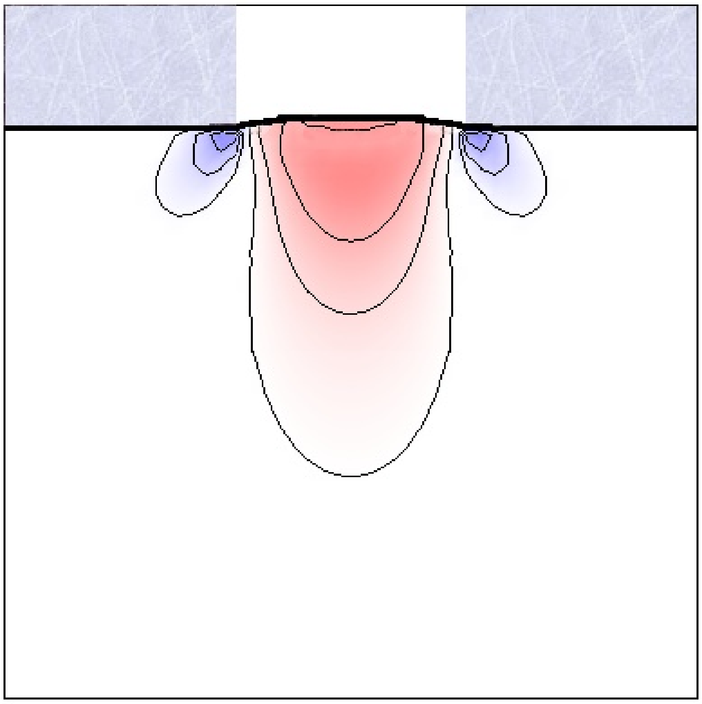


Figure 5.8: Aspiration test. Stretch distribution in the bubble crossection at $t = 15$ *with isolines at* $\pm1\%, \pm2\%, \pm3\%, \pm4\%$

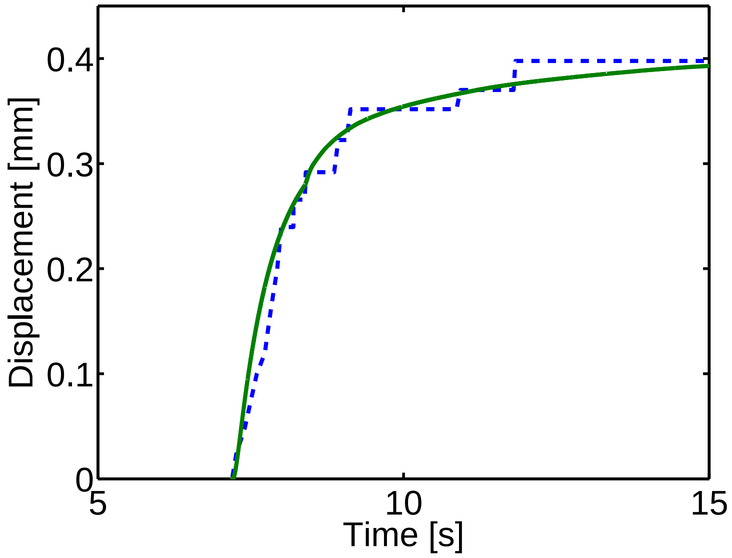


Figure 5.9: Aspiration test. Time history of the displacement of the bubble tip in the particle simulation(solid line) compared to the experimental measurement (dashed line) [116]

Chapter 6

Parallel Particle Simulations

6.1 Introduction

The dynamics of particle methods are governed by the interactions of the N computational particles resulting in an N-body problem with a computational cost that scales nominally as $\mathcal{O}(N^2)$. For short-ranged particle interactions, as in simulations of diffusion with the method of Particle Strength Exchange [49], the computational cost scales linearly with the number of particles. In the case of long-range interaction potentials such as the Coulomb potential in electrostatics, the gravitational potential in astrophysics, or the Biot-Savart law in VM, Fast Multipole Methods (FMM) [70] reduce the computational cost to $\mathcal{O}(N)$. Alternatively, long-range interactions can be described by equivalent field equations (such as the Poisson equation) that can be solved using meshes, resulting in hybrid Particle-Mesh (PM) algorithms [77, 84]. The computational cost of hybrid methods scales as $\mathcal{O}(M)$, where M denotes the number of mesh points used for resolving the field equations. The choice between FMM and PM techniques is dictated by the boundary conditions of the problem with FMM techniques allowing more flexibility on their specification, while PM schemes are well suited for periodic systems. An important factor, distinguishing FMM and PM techniques, is the parallelization efficiency of these methods, as the mesh regularity of the PM algorithm enables implementations that are typically one or two orders of magnitude faster than corresponding FMM [45, 178] implementations. FMM-based particle methods have limited scalability for shared memory systems [46], while their implementation in distributed memory systems is difficult due to the inherent global nature of the underlying tree data structure. It is important to observe, however, that even when FMM are used for the evaluation of the particle interactions, the need for hybrid PM algorithms is imperative in adaptive particle methods such as VM or SPH for the reinitialization of the distorted particle locations [97].

The parallel implementation of PM techniques is hindered by several factors:

- exploiting the symmetry of the particle interactions requires sending back of ghost contributions to the real particle,
- the simultaneous presence of particles and meshes prohibits a single optimal way of parallelization,
- complex-shaped computational domains and strong particle inhomogeneities require spatially adaptive domain decompositions,
- particle motion may invalidate the existing domain decomposition causing rising load imbalance, and complicates the implementation of multi-stage integration schemes,
- inter-particle relations constrain decompositions and data assignment.

State-of-the-art particle codes have successfully addressed some of the parallelization issues mentioned in Section 6.1 as for example the electromagnetic PIC code QUICKSILVER [127], demonstrating a parallel efficiency of 60% solving a scaled-size irregular case on 1024 processors, while achieving 90% efficiency in the ideal uniform load case on 3200 processors. For purely particle-based simulations a number of application-specific parallel software libraries is also available, such as PARTI for Monte-Carlo simulations [111], or the Parallel Utilities Library (PUL) [30]. Recent hybrid PM implementations include VORPAL [117] for plasma simulations and the Particle-in-Cell code PICARD [27].

While these codes provide a development platform for certain classes of particle methods, they do not allow generalizations to other classes of particle methods. For example

PICARD and VORPAL have been designed for PIC simulations and they can be used for load balancing and domain decomposition. Beyond these two aspects, however, they cannot be used for developing SPH, VM, or MD codes.

The Parallel Particle Mesh (PPM) library bridges the gap between general-purpose infrastructure libraries and application-specific simulation libraries, and provides a general-purpose parallel framework that can handle particles-only, mesh-only, as well as hybrid particle-mesh systems. The PPM library is portable through the use of standard languages (Fortran 90 and C) and libraries (MPI) and is applicable on single processor machines as well as on distributed memory, shared memory, and vector parallel processors. Computational efficiency is achieved by dynamic load balancing, dynamic particle re-distribution, explicit message passing, and the use of simple data structures. The library core provides several adaptive domain decomposition schemes, multiple processor assignment methods, load balance monitoring, dynamic load balancing, data mapping (sending, receiving), update of overlap regions, parallel file I/O, optimized inter-processor communication, neighbor lists (cell lists and Verlet lists [172]), routines for building trees, particle-to-mesh, and mesh-to-particle interpolation. This core infrastructure is supplemented with commonly used numerical methods such as mesh-based solvers, evaluation of differential operators on particles [54], FMM, parallel FFT, and multi-stage ODE integrators. Moreover, the PPM library provides bindings for the external libraries fftw, MathKeisan FFT (NEC, Inc.), and METIS (for graph partitioning for load assignment [91]).

This chapter describes the key concepts in parallel hybrid particle-mesh computations and shows the efficiency of the PPM library on benchmarks relevant for this thesis. Parallel timings, speedup, and efficiency are demonstrated for an rSPH simulation of an compressible vortex ring. This prototype code exemplifies the performance of the program codes used in this thesis.

6.2 Fundamentals

We consider simulation systems that are formulated in the framework of PM algorithms as outlined in the preceeding section. The field equations are solved using structured or uniform Cartesian meshes. As a result, the physical and computational domains are rectangular or cuboidal. Complex geometries are handled by immersed boundaries, through the use of source terms in the corresponding field equations, or through boundary element techniques. Adaptive meshing capabilities are possible using Adaptive Mesh Refinement (AMR) concepts as adapted to particle methods [18]. Moreover, the particle wavelet method [19] combine the natural adaptivity of wavelets for multiresolution problems with the robustness of the particle methods.

The simultaneous presence of particles and meshes requires different concurrent domain decompositions. These decompositions divide the computational domain into a minimum number of cuboidal *sub-domains* with sufficient granularity to provide adequate load balancing. The concurrent presence of different decompositions allows to perform each step of the computational algorithm in its optimal environment with respect to load balance and the computation-to-communication ratio. For the actual computations, the individual sub-domains are treated as independent problems and extended with *ghost mesh layers* and *ghost particles* to allow for communication between them.

Connections can be used to define relations between particles, e.g. particle pairs, triplets, quadruplets, etc. These relations may describe a physical interaction, such as chemical bonds in molecular systems, or a spatial coherence, such as a triangulation of an immersed boundary or an unstructured mesh.

Memory for internal lists and communication buffers is allocated by the PPM library. All other memory, such as simulation data (particles, fields) and index lists (cell lists, Verlet lists, etc.), is held by the client application. This ensures user-control over the data and allows multiple different sets of particles, connections, and fields to be used concurrently. The number of topologies, sub-domains, particle sets, fields, and meshes is only limited by the cumulative memory capacity of all processors.

6.3 Topologies

A *topology* is defined by the decomposition of space into sub-domains with the corresponding boundary conditions, and the assignment of these sub-domains onto processors. Multiple topologies may co-exist and library routines are provided to *map* particle and field data between them (cf. Section 6.4). Fields are defined on *meshes*, which in turn are associated with topologies. Every topology can hold several meshes. The only constraint is that sub-domain boundaries must align with mesh lines/planes.

As the domain decomposition may take several seconds to complete, a given topology is assumed to persist through longer periods of the simulation. For problems with free-space boundary conditions the extent of the computational domain is adjusted in order to enclose all particles at any time. An extra margin may be added to the computational domain to avoid repeated update of the topology. For problems in confined systems, subject to e.g. periodic boundary conditions, the extent of the computational domain is fixed and the decomposition is performed filling the entire space, disallowing void space(s). This assures that particles cannot leave the computational domain, which would require an immediate, potentially expensive, re-decomposition.

In order to achieve good *load balance*, both the load distribution and the computational cost of the topology creation are monitored throughout the simulation. The Stop at Rise (SAR) heuristic [111] is used in the PPM library to decide when problem re-decomposition is advised, i.e. when the cost of topology re-definition is amortized by the gain in load balance. Moreover, all topology definition routines can account for the true computational cost of each particle, for example defined by the actual number of its interactions. A routine is provided to compute this number based on the lengths of Verlet lists.

Domain decompositions

The PPM provides a number of different adaptive domain decomposition techniques for particles, meshes, and volumes, the latter defining geometric sub-domains with neither meshes nor particles present. These decompositions currently include: recursive orthogonal bisection, x-, y-, and z-pencils, xy-, xz-, and yz-slabs, cuboids, and a user-defined decomposition. Recursive orthogonal bisection is based on an adaptive binary tree (cf. Section 6.11), where subdivisions are allowed in all spatial directions. Pencil decompositions prohibit subdivisions in one direction, resulting in an adaptive decomposition where each sub-domain extends over the whole computational domain in at least one spatial dimension. Such decompositions are useful when performing fast Fourier transforms. In slabs, two directions are fixed. Cuboids are created using adaptive quad- and oct-trees in two and three dimensions, respectively, and the user-defined decomposition allows the client program to explicitly specify the sub-domains. After checking the validity of such a decomposition, the PPM library directly proceeds with assignment of the sub-domains to the processors.

In addition, a special null decomposition is provided, that does not perform any domain decomposition. It creates only one "sub-domain" which is the computational domain itself. This trivial "decomposition" is used to evenly distribute the particles among processors, irrespective of their spatial location. The resulting special topology is called the *ring topology* and the sub-domain is assigned to every processor. The ring topology supports full $\mathcal{O}(N^2)$ calculations, and also allows to distribute data of initially unknown processor affiliation (cf. Sections 6.4 and 6.5).

To assess the performance of the different domain decomposition schemes, we compare them on four test cases using 16 processors (Table 6.1). The quality of decomposition is quantified by the standard deviation of the number of particles across processors and by the total number of ghost particles needed to communicate the boundaries. The domain is decomposed using a non-adaptive binary tree, Recursive Orthogonal Bisection (ROB), and an adaptive oct-tree. The subdomains are assigned onto the processors in an optimal way, minimizing the total length of communication boundaries. This assignment is performed using the external library METIS[91]. One million particles are distributed in the unit cube in four ways: uniformly, on a diagonal from the point (0,0,0) to the point (1,1,1), on the surface of a sphere with radius 0.25, and on a spiral. The computational time needed to contruct the topologies is about 30 milliseconds per subdomain in all cases.

The adaptive oct-tree and ROB decomposition provide a topology where the particles are well distributed in all cases

Particle Distribution	Standard deviation of particles per processor		
	Non-adaptive tree	ROB	Adaptive oct-tree
Uniform	422	268	265
Sphere	62501	1865	2626
Spiral	73350	2336	6011
Diagonal line	108255	148	161
Particle Distribution	Average number of ghosts particles per processor		
	Non-adaptive tree	ROB	Adaptive oct-tree
Uniform	33847	33832	33750
Sphere	35268	36526	28187
Spiral	10584	31102	22297
Diagonal line	20050	28940	46232

Table 6.1: Comparison of different domain decomposition schemes on four test problems. For each scheme, the equidistribution of particles and the total communication overhead are reported. Details see main text.

Assignment of sub-domains onto processors

Load balancing in the PPM library comprises two main components: domain decomposition and assignment of sub-domains onto processors. While the former has to ensure sufficient granularity and partitioning of the computational cost, the latter has to ensure even distribution of computational load among processors, accounting for possible differences in processor speeds. The computational cost for each sub-domain, as determined by the number of particles, the number of mesh points, or the true computational cost, is known from the domain decomposition step. The individual processor speeds are measured internally by the PPM library, solving a small Lennard-Jones system [8] with an increasing number of particles until all processors report sufficient timing statistics..

Using this information, PPM provides several methods of assigning the sub-domains to the processors. The PPM-internal method assigns contiguous blocks of sub-domains to processors until the accumulated cost of a processor is greater or equal to the theoretical average cost under uniform load distribution. The average is weighted with the relative processor speeds. In addition, four different METIS-based [91] assignments, and a user-defined assignment are available.

Boundary conditions

At the external boundaries of the computational domain Neumann, Dirichlet, free space, symmetric, and periodic boundary conditions are supported. These conditions complement the particular mesh-based solver that is being employed. More involved boundary conditions and complex boundary shapes are represented inside the computational domain by defining connections among the particles, or using immersed interfaces.

6.4 Mapping

PPM topologies implicitly define a data-to-processor assignment. *Mapping* routines provide the functionality of sending particles and field blocks to the proper processor, that is the one that "owns" the corresponding sub-domain(s) of the computational space. Three different mapping types are provided for both particles and field data:

1. a *global* mapping which involves an all-to-all communication,

2. a *local* mapping for neighborhood communication, and

3. *ghost* mappings to update the ghost layers.

In addition, a special *ring shift* mapping is provided for particle data on the ring topology, and a *connection mapping* is provided for taking into account links between particles.

The global mapping is used to perform the initial data-to-processor assignment or to switch from one topology to another, whereas the local mapping is mainly used to account for particle motion during a simulation. Communication is scheduled by solving the minimal edge coloring problem using the efficient approximation algorithm by Vizing [175, 51, 53]. Ghost mappings are provided to receive ghost particles or ghost mesh points, or to send ghost contributions back to the corresponding real element, for example after a symmetric particle-particle interaction or a particle-to-mesh interpolation. The ring shift mapping sends data-sets around all processors, while each processor keeps a local copy of its original data. After every ring shift, each processor can perform its operations using the original local data-set as well as the current traveling set. During a complete cycle, all possible pair interactions will thus be considered. Finally, connection mappings are provided to distribute connections among processors according to an existing distribution of particles, and to update connection lists when particles have moved across processor boundaries.

All mapping types are organized as stacks. A mapping operation consists of 4 steps: (1) defining the mapping, (2) pushing data onto the send stack, (3) performing the actual send and receive operations, and (4) popping the data from the receive stack. This architecture allows data stored in different arrays to be sent together to minimize network latency, and mapping definitions to be re-used by repeatedly calling the push/send/pop sequence for the same persistent mapping definition.

Mappings of field data can be masked. An optional binary mask selects which mesh points are to be mapped and which ones are not. The values of non-mapped points remain unaffected by the mapping operation.

6.5 Particle-Particle Interactions

The evaluation of Particle-Particle (PP) interactions is a key component of PM algorithms. Sub-grid scale phenomena can require local particle-based corrections, differential operators can be evaluated on irregular locations [54], or the main dynamics of the system can be governed by particle interactions.

The PPM library implements PP computations using cell lists, Verlet lists, or the full $\mathcal{O}(N^2)$ direct method. Both symmetric and non-symmetric interactions are supported, the former to reduce the amount of duplicated work. In each method, the interaction potential or kernel can be specified either by a function pointer to a user function, by passing a look-up table of kernel values, or by choosing one of the predefined PPM-internal kernels.

The direct evaluation makes use of the PPM ring topology (cf. Section 6.3) and the ring shift mapping (cf. Section 6.4) to compute all N^2 pair interactions. Cell lists are provided for local (short range) interactions. Hereby, particles are sorted into equisized cuboidal cells, whose size reflects the interaction cutoff. In PPM, cell lists are defined per sub-domain and *ghost cells* are used around each sub-domain. Fig. 6.1 illustrates the cell-cell interactions in asymmetric and symmetric evaluations. To achieve complete symmetry, a novel interaction scheme involving diagonal interactions is introduced. This scheme reduces the amounts of memory overhead and communication for symmetrically evaluated particle interactions by 33% in two dimensions and 40% in three dimensions. Furthermore, it allows a constant communication overhead with increasing number of processors, amounting to a parallel shift in the speedup plot rather than a convex curve. Given the cells are numbered in ascending x, y, (z), starting from the center cell with number 0, the cell-cell interactions in PPM are: 0–0, 0–1, 0–3, 0–4, and 1–3 in two dimensions, and 0–0, 0–1, 0–3, 0–4, 0–9, 0–10, 0–12, 0–13, 1–3, 1–9, 1–12, 3–9, 3–10, and 4–9 in three dimensions. The difference between symmetric and non-symmetric PP interactions is measured using a PSE diffusion problem. The computational time per time step is found to decrease by a factor of 1.72 when going from asymmetric to symmetric interactions.

For spherically symmetric interactions, cell lists contain up to $27/(4\pi/3) = 81/(4\pi) \approx 6$ times more particles than actually needed. Verlet lists [172] are provided to reduce this overhead. For each particle they involve an explicit list of all other particles it has to interact with.

Besides PP interactions, PPM also supports interactions based on inter-particle connections. Neighbor lists are not required in this case since a connection is an explicit list of

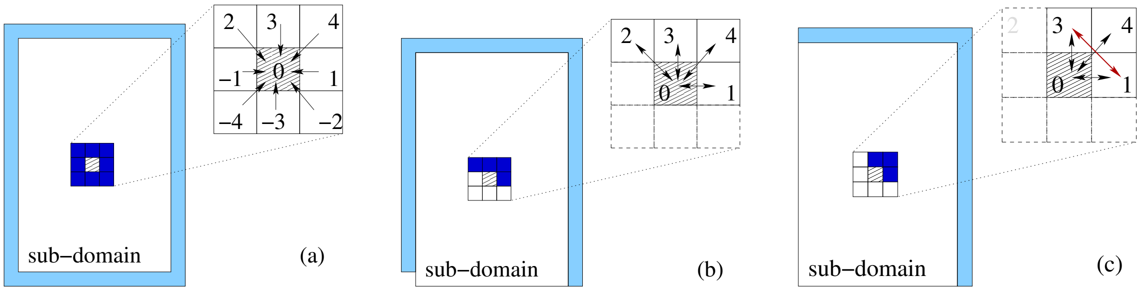


Figure 6.1: Cell-cell interactions and ghost-layer arrangement. (a) For non-symmetric particle-particle interactions, the ghost layer (light gray) extends all around the sub-domain. Interactions are one-sided. (b) In traditional symmetric cell list algorithms, ghost layers are required on all but one boundary of the domain. (c) In PPM*, diagonal interactions are introduced (1–3). Ghost layers are now symmetric and do not overlap with any other ghost layers of neighboring sub-domains. This results in less communication, better scaling in memory and simpler algorithms (e.g. when considering connected particles). The two-dimensional case is depicted. See text for interactions in the three-dimensional case.*

all its member particles. Connection interactions are supported by separate routines.

Alternatively, the client program can implement its own interaction routines. Template subroutines are provided for the use of cell lists, Verlet lists, direct interactions, and connection interactions.

In addition to the routines performing the actual computations, the PPM library also provides a routine to create look-up tables from either a function pointer or an internal kernel. Such tables can then be passed to any of the compute routines for the evaluation.

6.6 Particle-Mesh and Mesh-Particle Interpolations

All hybrid PM methods involve interpolation of irregularly distributed particle quantities from particle locations onto a regular mesh and interpolation of field quantities from the grid points onto particle locations.

These interpolations are utilized for two purposes, namely:

- the transformation of field quantities between the particle solver and the field solver, and
- the reinitialization of distorted particle locations.

While the first issue is a well-established notion in PM techniques, the *reinitialization* of particle locations and weights when particle locations get distorted by the flow map is a critical, albeit often overlooked, aspect of particle methods for the simulation of continuous systems [97]. Particle overlap is needed in order to ensure convergence of the method and this is achieved by periodically re-initializing particles onto a regular mesh ("remeshing"). This involves the interpolation of particle properties onto the mesh and replacing the current set of particles by new particles created at the locations of the mesh points.

The PPM library provides routines that perform these operations. The interpolation weights W can be pre-computed and stored to facilitate adjustments of the interpolation or interpolate several sets of quantities. If the weights are not pre-computed, they are determined during the actual interpolation. Currently implemented interpolants include first and second order B-Splines and the M_4' function [106].

The interpolation of mesh values onto particle locations readily vectorizes: the interpolation is performed by looping over the particles and receiving values from mesh points that lie in the support of the interpolation kernel. Therefore, the values of individual particles can be interpolated independently.

The interpolation of particle values onto mesh locations, however, leads to data dependencies as the interpolation is still performed by looping over particles, but a mesh point may receive values from more than one particle. To circumvent this problem, the PPM library implements the following technique [177]: when new particles are created in the course of remeshing, we assign colors to the particles such that no two particles within the support of the interpolation kernel have the same color. Particle-to-mesh interpolation then visits the particles ordered by color to achieve data independence. This coloring

	CPU time	vector operation ratio	vector length
colored	2.69 s	99%	230.6 words
classical	30.1s	0.36%	4.1 words

Table 6.2: Comparison of the vector performance of classical particle-to-mesh interpolation and the present coloring scheme.

scheme enables vectorization of particle-to-mesh interpolations as confirmed by a test on the NEC SX-5 vector computer (Table 6.2). Without this coloring scheme, interpolation in hybrid particle-mesh methods would be prohibitively expensive on vector architectures.

6.7 Parallel Fast Multipole Method

The Fast Multipole Method (FMM) can evaluate Dirichlet and free-space boundary conditions for the computationally more efficient Mulit-grid solver. The implementation of the parallel FMM module is based on the topology, tree, and mapping routines provided by the library core. Hereby, the FMM module creates multiple temporary topologies. The first topology comprises the sub-domains defined at the level of the tree that holds at least as many sub-domains as there are processors. Subsequent levels of the FMM tree structure are also declared as PPM topologies. By means of the user-defined assignment scheme, individual processors operate on disjoint sub-trees to minimize the amount of communication. The particles are then mapped according to the finest topology, which contains all the leaf boxes as sub-domains. The PPM tree directly provides index lists to the particles in each box. This allows straightforward computation of the expansion coefficients on the finest level, without requiring communication. The computed leaf coefficients are shifted to parent boxes by recursively traversing the tree toward its root, cf., e.g., [32]. Since the topologies are defined such that each processor holds a disjoint subtree, the expansions can be shifted without communication.

To evaluate the potential at the locations of a set of target particles, these particles are first mapped onto the finest-level PPM topology. A pre-traversal of the tree then decides which expansion coefficients and source particles are needed for each target point. This is done by traversing the tree from the root down to the leafs using a stack data structure. On each level, we check if the corresponding box is already far enough away from the target particle. This is done by comparing the distance (D) between the target particle and the center of the box to the diameter (d) of the box. If the ratio $D/d > \theta$, the expansion of that box is used and the traversal stops.

When evaluating the potential, expansion coefficients or particles from other processors may be needed. Before evaluating the potentials, the expansion coefficients from all processors are thus globally communicated. This can be done since the data volume of the coefficients is much smaller than the original particle data. Required particles are received on demand as additional ghosts using the regular PPM ghost mapping routines.

6.8 Mesh-Based Solvers

Meshes can be used to solve the field equations associated with long-range particle interactions [84] or to discretize the differential operators in the governing equations of the simulated physical system. These operators are often local and their computational cost scales linearly with the number of particles or mesh points.

A large class of pair potentials in particle methods can be described by the Poisson equation as it appears in MD of charged particles via electrostatics (Coulomb potential), fluid mechanics in stream-function vorticity formulation (Biot-Savart potential), and astrophysics (gravitational potential). The Poisson equation is expressed as:

$$\nabla^2\Phi = \rho(x, y, z). \tag{6.1}$$

The PPM library provides Poisson solvers based on FFTs and geometric MultiGrid (MG).

The FFT-based Poisson solver parallelizes a multi-dimensional FFT using a sequence of one- or two-dimensional FFTs performed on pencil and slab topologies (cf. Section 6.3). A single three-dimensional Fourier transform thus consists of mapping the data onto a temporary xy-slab topology, performing a two-dimensional FFT, mapping onto a temporary z-pencil topology, and performing a one-dimensional FFT. The actual serial one-dimensional or two-dimensional FFTs are performed using the external libraries fftw or MathKeisan (on NEC SX vector architectures).

The geometric MG method is implemented in PPM as a fast iterative method for solving the Poisson equation. The advantage of parallel MG solvers consists in restricting communication to the ghost layers whereas the corresponding FFTs require several global mappings. The PPM MG supports both the V and W cycle [169]. The Laplacian is discretized using five and seven point stencils in two and three dimensions, respectively. As residual smoother we employ the red-black successive over-relaxation scheme, which includes the Gauss-Seidel smoother as a special case. Furthermore, the full-weighting scheme [169] is used for the restriction of the residual, and bilinear (in two dimensions) or trilinear (in three dimensions) interpolation for the prolongation of the function corrections [169].

6.9 ODE Solvers

Simulations using particle methods entail the solution of systems of ODEs as outlined in Section 2.1. The characteristics of the Initial Value Problems (IVP) represented by these ODEs explicitly reflect the physics of the system that is being simulated.

The PPM library provides a set of explicit integration schemes to solve these IVPs. The ODE solver of PPM is designed as a "black-box" solver. The user selects the method to be used and provides as a function pointer a routine that computes the right-hand sides of the ODEs. Both allocation of storage (for the stages of multi-step schemes) and the actual computation of the stages is performed by the library. Second order ODEs are solved by transforming them into a system of first order problems and parallelism is achieved by mapping the integrator stages along with the other particle quantities (cf. Section 6.4). Thus, at the last stage of the integrator, the previous stages are available on the processor that currently hosts the particles, and the final particle update is completed without further communication. Low-storage schemes have the additional advantage of requiring little communication. The set of available integrators includes forward Euler with and without super time stepping [6], 2-stage and 4-stage standard Runge-Kutta schemes, Williamson's low-storage third order Runge-Kutta scheme [182], and 2-stage and 3-stage TVD Runge-Kutta schemes [151].

6.10 Parallel I/O

File I/O in distributed parallel environments exist in two different modes: *distributed* and *centralized*. By distributed we denote the situation where each processor writes its part of the data to its local file system. Centralized I/O on the other hand will produce a single file on one of the nodes, where the data contributions from all processors are stored. The latter is convenient for small or aggregated data, or for writing files that will later be read on a different number of processors, e.g. to continue an interrupted simulation.

The PPM library provides a parallel I/O module which supports both binary and ASCII read and write operations in both modes, distributed and centralized. The I/O mode is transparent to the client application. Write operations in the centralized mode can concatenate or reduce (sum, replace) the data from individual processors; read operations can transparently split the data in equal chunks among processors or send an identical copy to each one. The basic assumption behind the split mode is that no processor will be able to hold all the data in memory at any time. To improve performance of the centralized mode, network communication and file I/O are overlapped in time using non-blocking MPI calls.

6.11 Adaptive trees

The tree construction in the PPM library provides non-adaptive and adaptive binary trees, quad-trees, and oct-trees. At any stage of the tree, the space is subdivided into M boxes $\{B_k\}$. The indices i and j are used to denote coordinate directions. Adaptivity and subdivision behavior are guided by two *cost functions* ϕ_1, ϕ_2. Both cost functions are linear combinations of the three cost contributions: particle costs c_p (user-specified or unity per particle), mesh points (number of mesh points in the box $m_B = \prod m_{B,i}$), and geometry (volume of the box $|B| = \prod |B|_i$), with user-provided coefficients α, β, γ:

$$\phi_{\{1,2\}}(B_k) = \alpha_{\{1,2\}} \sum_{p \in B_k} c_p + \beta_{\{1,2\}} m_{B_k} + \gamma_{\{1,2\}} |B_k| . \tag{6.2}$$

The first cost function ϕ_1 guides adaptivity of the tree since the next subdivision will be applied to the box B_K of largest ϕ_1. The second cost function ϕ_2 defines the direction(s) of subdivision and position of the subdivision plane(s). Suppose B_K is to be subdivided next. In order to create the minimum ϕ_2-cut when subdividing a box, the tensor of intertia T is computed from the particle locations $x_p = (x_{p,i})$ and costs c_p as:

$$T_{ii} = \sum_{p \in B_K} \left(\sum_{j \neq i} x_{p,j}^2 \right) c_p \,, \qquad T_{ij} = \sum_{p \in B_K} x_{p,i} x_{p,j} c_p \,. \tag{6.3}$$

The eigenvectors v_r of T are scaled with the corresponding eigenvalue λ_r: $\nu_r = \lambda_r (v_r / |v_r|)$ and projected onto the unit coordinate vectors e_i. The number of mesh points in this direction $m_{B,i}$ and the length of the box in this direction $|B|_i$ are normalized and added to form a score value s for each coordinate direction

$$s(e_i) = \alpha_2 \sum_r e_i \cdot \nu_r + \beta_2 \left(\sum_j m_{B_K,j} \right)^{-1} m_{B_K,i} + \gamma_2 \left(\sum_j |B_K|_j \right)^{-1} |B_K|_i \,. \tag{6.4}$$

The subdivision directions (1, 2, or 3) are chosed in order of ascending score. The client program can however specifically disallow subdivisions in certain directions to enforce pencil-type or slab-type boxes. The actual position of a cut perpendicular to direction I is determined as the corresponding component of the center of mass of ϕ_2 within the box B_K

$$\phi_2(B_K)^{-1} \left[\alpha_2 \sum_{p \in B_K} x_{p,I} c_p + \mu_I(B_K) \left(\beta_2 m_{B_K} + \gamma_2 |B_K| \right) \right] , \tag{6.5}$$

subject to the constraint that a client-specified minimum box size is not under-run. Here, $\mu(B_k) = (\mu_i(B_k))$ denotes the geometric center of box B_k. To terminate the tree, multiple concurrent termination criteria can be prescribed.

6.12 Parallel Efficiency Benchmarks

The parallel efficiency of the library is measured based on the following tests in this thesis:

1. The potential evaluation using FMM,
2. The solution of the scalar Poisson equation using geometric MG,
3. The solution of the scalar Poisson equation using FFTs, and
4. Simulation of compressible viscous flow induced by a double shear layer using remeshed Smooth Particle Hydrodynamics (rSPH).

In addition, we illustrate the application of the library to problems in fluid dynamics with the simulation of a compressible vortex ring using rSPH.

Performance of the PPM library is tested for both a fixed-size and a scaled-size problem for all cases except the diffusion simulation. In the fixed-size problems, the number of mesh points and particles is kept constant, i.e. the work load per processor decreases with increasing number of processors. In the scaled problems, mesh point and particle numbers grow proportionally to the number of processors, resulting in a constant work load per processor. Timings and parallel efficiency figures are collected on the IBM p690 computer of the Swiss National Supercomputing Centre (CSCS). The machine consists of 8 Regatta nodes with 32 1.3 GHz Power4 processors per node. Within each node, it is configured in 4 groups with 8 processors sharing 12 GB of memory. Each processor has a peak performance of 5.2 GFlop/s, and the nodes are connected by a 3-way Colony switch system.

In each test, we measure the elapsed wall-clock time t_{ij} for each time step j on each processor $i = 1, \ldots, N_{proc}$. To account for synchronous communication steps, we report the maximum of these times over all processors. This maximum is averaged over 5 to 10 samples to compute speedup S and efficiency e:

$$S(N_{proc}) = \frac{t(1)}{\text{mean}_j \max_i t_{ij}(N_{proc})} \cdot \frac{N\left(N_{proc}\right)}{N\left(1\right)}\,, \tag{6.6}$$

$$e(N_{proc}) = \frac{S(N_{proc})}{N_{proc}}\,, \tag{6.7}$$

where $t(1)$ is the time on a single processor (linearly extrapolated if not measured), $t_{ij}(N_{proc})$ is the time on N_{proc} processors, $N(1)$ is the problem size on a single processor, and $N(N_{proc})$ is the problem size on N_{proc} processors. To account for the $\mathcal{O}(N \log N)$ scaling of the FFTs, the second factor of the speedup is accordingly adjusted in the benchmarks of the FFT-based Poisson solver.

Vectorization and parallel efficiency on vector machines are tested using the NEC SX-5 computer at CSCS. This is a shared memory machine with 16 NEC SX-5 vector processors. Each processor has a peak performance of 8 GFlop/s and 64 vector registers of a length of 256 words (2048 Bytes) each.

In addition to the benchmark tests, simulations are performed on a distributed memory cluster consisting of 16 2.2 GHz AMD Opteron 248 processors running under Linux. The nodes of this cluster are connected by a switched gigabit ethernet network.

6.12.1 The Fast Multipole Method

The test cases for the PPM FMM involve 10^5 source points with a uniformly random distribution in a cubic box. The potential induced by these points is computed at the locations of 10^5 target points, also uniformly randomly distributed in the same cube. Fig. 6.2(a) shows the wall-clock time as a function of the number of particles. The acceptance factor θ for the tree traversal is set to 1.5, and we vary the expansion order (l). The scaling of the FMM is compared to the $\mathcal{O}(N^2)$ scaling of the direct evaluation method and reveals the expected $\mathcal{O}(Nlog(N))$ behavior. Fig. 6.2(b) shows the parallel speedup of the PPM FMM on up to 16 processors of the Linux cluster. The wall-clock time is 452 seconds on 1 processor and 36.8 seconds on 16. The observed loss in efficiency is mainly caused by the global communication of the expansion coefficients.

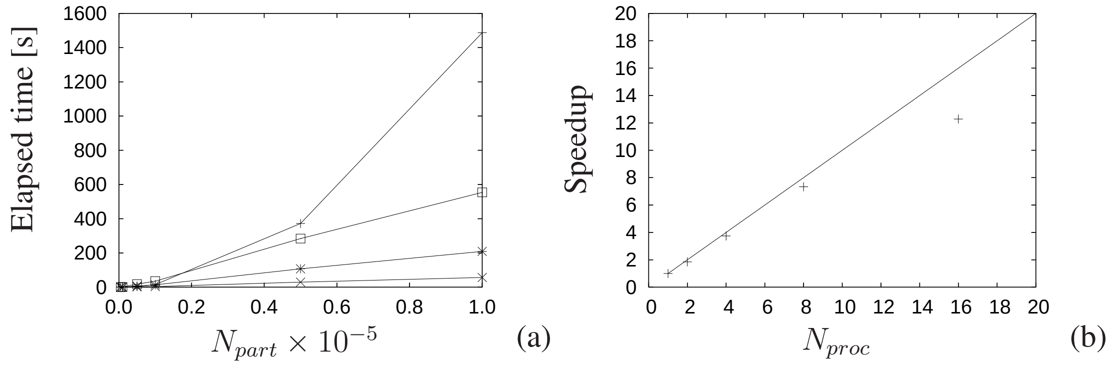


Figure 6.2: Performance of the Fast Multipole Method (FMM) implementation in the PPM *library. (a) Serial performance of the FMM as function of number of particles and the order (l) of the expansion: -+-: direct calculation; -□-: $l = 9$; -∗-: $l = 5$; -×-: $l = 1$. (b) Parallel speedup using 10^5 particles and $l = 5$ on up to 16 nodes of the 2.2 GHz Opteron Linux cluster. —: linear scaling; $+$: measurement.*

6.12.2 Parallel Multigrid Poisson solver

We test the performance of the PPM MG field solver by solving the scalar Poisson Eq. (6.1) with the right hand side

$$\rho(x, y, z) = \sin(2\pi x) \sin(2\pi y) \sin(2\pi z), \quad x, y, z \in [0, 1] \,, \tag{6.8}$$

subject to periodic boundary conditions. The initial value of Φ is zero everywhere and we use the V(2,1) cycle with one smoothing step at the finest level.

We conduct three tests. The first involves the fixed case with $256 \times 256 \times 256$ mesh points, while the two others are scaled cases, one starting from a $128 \times 128 \times 128$ mesh, the other one starting from $256 \times 256 \times 256$. Efficiency and speedup for the scaled cases are shown in Fig. 6.3 and for the fixed case in Fig. 6.4. We observe a strong decrease in the parallel efficiency up to 8 processors due to the congestion of the shared memory. This is removed when using only one processor per node in a pure distributed memory setup, cf. Figs. 6.3 and 6.4, and the efficiency improves to 90% on 16 processors for the scaled case. The effective efficiency based on the timing obtained on 8 processors is 92% for the large scaled case on 64 processors. The efficiency of the PPM MG solver for a

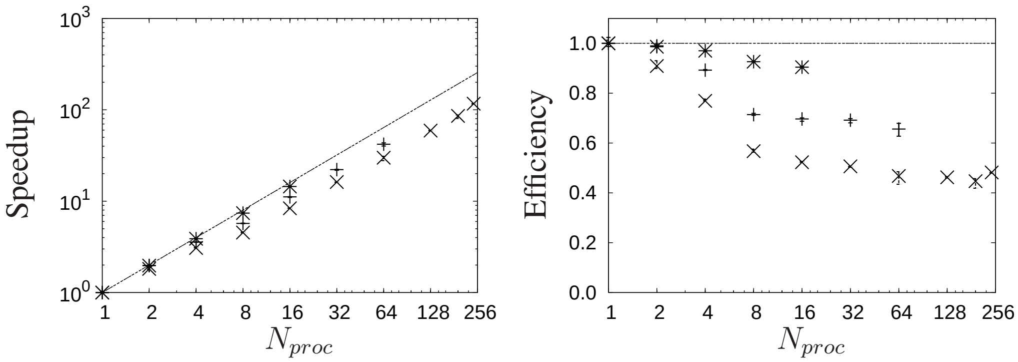


Figure 6.3: Parallel speedup and efficiency of the MG Poisson solver for the scaled-size problem. The initial mesh resolution on one processor is $256 \times 256 \times 256$ $(+)$ *and* $128 \times 128 \times 128$ $(\times)$*, respectively. Using only one processor per node in the large case, the bottleneck of the shared memory is removed* $(*)$*. Each point is averaged from 5 samples, error bars indicate min-max span. All timings are performed on the IBM p690.*

$1024 \times 1024 \times 1024$ system is 66% for the large scaled case on 64 processors. For this system, the elapsed time is 10.5 seconds per V-cycle, and thus 42 seconds for the 4 V-cycles needed to reduce the L_2 error to 10^{-4}. A system with half a billion unknowns is solved in the small scaled case on 242 processors at 48% efficiency in 1.7 seconds per V-cycle. This compares well with the 41% efficiency achieved by the Prometheus multigrid library [5] on 128 IBM Power3 processors for the same problem size.

The vectorization of the PPM MG solver is tested on the NEC SX-5 using up to 8 processors. The PPM MG sustains a performance of 2.4 GFlop/s per processor (30% of peak performance) with a vector operation ratio of 95% and a parallel efficiency of 96%. On this machine, a single V cycle on a $512 \times 512 \times 512$ system takes 1.21 seconds on 8 processors.

6.12.3 Parallel FFT-based Poisson solver

We test the parallel performance of the FFT-based Poisson solver and compare it to the MG solver by solving the same scalar Poisson Eq. (6.1) with the same right hand side

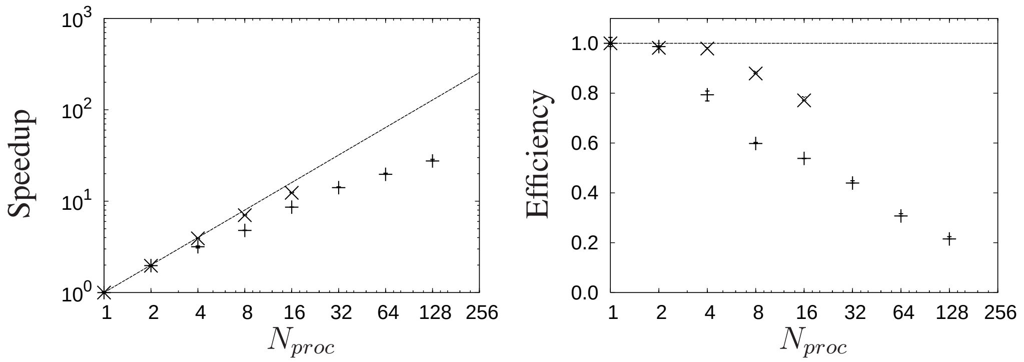


Figure 6.4: Parallel speedup and efficiency of the MG Poisson solver for the fixed-size problem with $256 \times 256 \times 256$ *mesh points on 2 to 128 processors* ($+$)*. Using only one processor per node in the large case, the bottleneck of the shared memory is removed* ($\times$)*. Each point is averaged from 5 samples, error bars indicate min-max span. All timings are performed on the IBM p690.*

Eq. (6.8), subject to periodic boundary conditions. All Fourier transforms are performed using the parallel FFT routines of the PPM library as described in Section 6.8.

The parallel speedup and efficiency for the scaled problem as shown in Fig. 6.5 exhibit two characteristic regions. The first one ranges from 1 to 8 processors, the second one from 8 and beyond. From 1 to 8 processors the efficiency drops significantly, due to conflicts and congestion in the shared memory architecture within each compute node. This is verified in a separate benchmark (Fig. 6.5: $\times$), in which only one processor per node is used. In this case, the congestion is removed and the efficiency significantly improves to 68% on 16 processors for the scaled case. Solving the Poisson equation to machine precision on a $128 \times 128 \times 128$ mesh takes 0.6 seconds on a single processor. The corresponding scaled system on 64 processors ($512 \times 512 \times 512$) requires 2.4 seconds. Speedup and efficiency for the fixed-size problem are shown in Fig. 6.6. Again, the scaling improves beyond 8 processors, similiar to the scaled case.

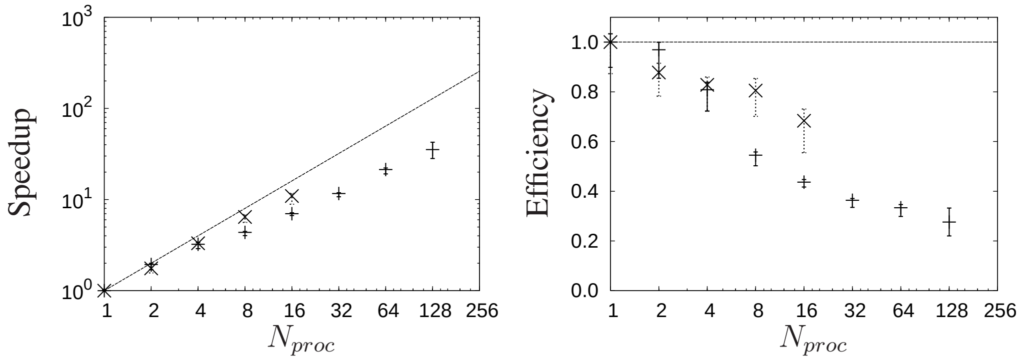


Figure 6.5: Parallel speedup and efficiency of the FFT-based Poisson solver for the scaled-size problem starting with $128\times128\times128$ *mesh points on one processor* ($+$)*. Using only one processor per node, the bottleneck of the shared memory is removed* ($\times$)*. Each point is averaged from 5 samples, error bars indicate min-max span. All timings are performed on the IBM p690.*

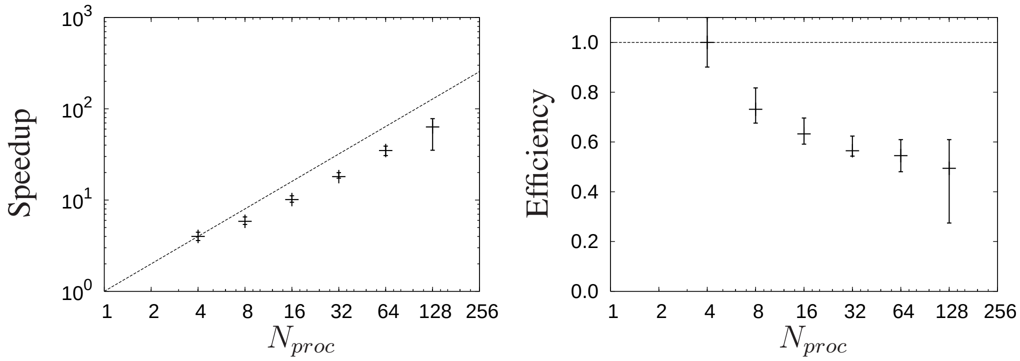


Figure 6.6: Parallel speedup and efficiency of the FFT-based Poisson solver for the fixed-size problem with $512 \times 512 \times 512$ *mesh points on 4 to 128 processors. Each point is averaged from 5 samples, error bars indicate min-max span. All timings are performed on the IBM p690.*

6.12.4 Three-Dimensional Remeshed Smooth Particle Hydrodynamics

We consider a client application based on a novel, computationally efficient formulation of the remeshed SPH (cf. Chapter 2). The rSPH client is applied to the simulation of a three-dimensional compressible double shear layer [63]. In order to measure the parallel performance, we consider a computational domain fully populated with particles so that the reported performance measures are independent of the particular flow problem. We furthermore present results from the application of the present rSPH methodology to the evolution of a compressible vortex ring (cf. Chapter 4), demonstrating the stability and accuracy of the method.

Parallel speedup, timing and efficiency

The speedup and parallel efficiency of the rSPH client are shown in Figs. 6.7 and 6.8 for the scaled and fixed-size problem, respectively. The largest simulation considered in this rSPH test case comprises 268 million particles and achieves a parallel efficiency of 91% on 128 processors. The efficiency on 32 processors using 67 million particles is also 91%, which compares well with the 85% efficiency of the GADGET SPH code by Springel *et al.* [155] on 32 processors of the same computer model (IBM p690). The efficiency in the fixed-size problem ranges between 100% and 84%. One time step of the simulation using 16.8 million particles takes 196.9 seconds on 4 processors and 7.3 seconds on 128 processors.

The communication overhead of the present rSPH client is assessed using 16 million particles. The fraction of time spent in communication is less than 13% of the total computational time in all cases (Table 6.3). Using 4 processors, only 5% of the total time is spent in communication. The communication effort increases by a factor of 2.5 when using 64 times more processors. This demonstrates the high efficiency of the mapping and communication routines in the PPM library.

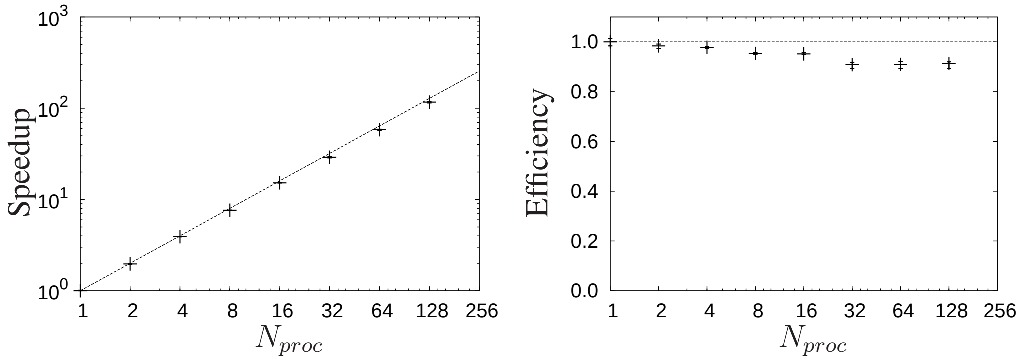


Figure 6.7: Parallel speedup and efficiency of the PPM *rSPH client for the scaled-size problem starting with 2 million particles on one processor. Each point is averaged from 5 samples, error bars indicate min-max span. All timings are performed on the IBM p690.*

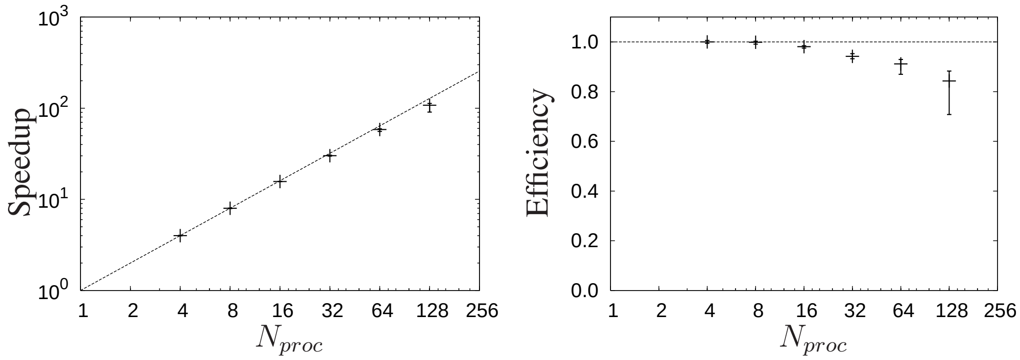


Figure 6.8: Parallel speedup and efficiency of the PPM *rSPH client for the fixed-size problem with 16.8 million particles on 4 to 128 processors. Each point is averaged from 5 samples, error bars indicate min-max span. All timings are performed on the IBM p690.*

N_{proc}	total time [s]	communication [s]	in percent
4	195	10	5%
16	50	4	8%
64	14	1.2	11%
128	7	0.8	12%

Table 6.3: Communication-to-computation ratio of the PPM *using the rSPH client with 16 million particles.*

Chapter 7

Particle Immersed Boundary Method

7.1 Introduction

Biofluid dynamics is characterized by the interaction of elastic incompressible tissue with viscous incompressible fluid. In some cases the elastic tissue is active, like muscle, which means that it can act a source of mechanical energy. The Immersed Boundary Method is both a mathematical formulation and a computational method for the biofluid dynamic problem. In the immersed boundary formulation, the equations of fluid dynamics are used in an unconventional way, to describe not only the fluid but also the immersed tissue with which it interacts. In the computational scheme motivated by this formulation, the fluid equations are solved on a fixed (Eulerian) cubic lattice, where elastic forces are computed from a Lagrangian representation of the immersed elastic tissue. The material points of the tissue move freely through the cubic lattice of the fluid computation. The two components of this Eulerian/Lagrangian scheme are linked by a smoothed version of the Dirac delta function, which is used to apply elastic forces to the fluid, and to interpolate the fluid velocity at the representative material points of the elastic tissue. A key aspect of this methodology is the simple handling of moving complex boundaries because it does not require an adaptive computationally expensive mesh generation. This methodology has been applied to the heart and its valves by Peskin [124] who introduced this method within this context.

Fadlun and Verzicco *et al.* [59] proposed an immersed boundary method for finite-difference methods where the velocity of fluid cells close the complex boundary is interpolated linearly between the boundary and the neighboring fluid cell. This method leads to second order accuracy and requires a accurate distance-information to the boundary at the all neighboring fluid cells. Kim [93] presented a similar approach for finite-volume methods combined with a mass source to increase the accuracy.

These approaches, however, are limited to Eulerian methods for incompressible flows. We present a novel particle Immersed Boundary method (pIBM) that is applicable to Lagrangian particle methods, such as smoothed particle hydrodynamics. The geometry of the body is described by Lagrangian points. A forcing term is evaluated on the boundary points such that the no-slip boundary condition on the body is fulfilled. The extrapolation of the forcing term onto the neighboring particles involves high-order B-Splines kernel.

The governing equations of the fluid along with general initial and boundary conditions are introduced in Chapter 4.

We demonstrate the performance of the pIBM on channel flow, flow past a circular cylinder and sphere. We characterize the flow for various Reynolds numbers with experimental and numerical results presented in the literature.

7.2 Particle Presentation of Immersed Boundaries

7.2.1 Particle Immersed Boundary Method (pIBM)

In pIBM (Fig.7.1), a forcing term f is added to the momentum equation (Eq.(4.2)) such that the no-slip condition is satisfied on the boundary.

$$\rho \frac{D\mathbf{u}}{Dt} = -\nabla p + \nabla . \tau + \mathbf{f} \tag{7.1}$$

We approximate the material derivative by a differential quotient:

$$\rho_i \frac{\mathbf{u}_{i+1} - \mathbf{u}_i}{\Delta t} = -\nabla p_i + \nabla . \tau_i + \mathbf{f}_i \tag{7.2}$$

Solving for f_i and assuming we reach the desired velocity within this time step ($\mathbf{u}_{i+1} =$

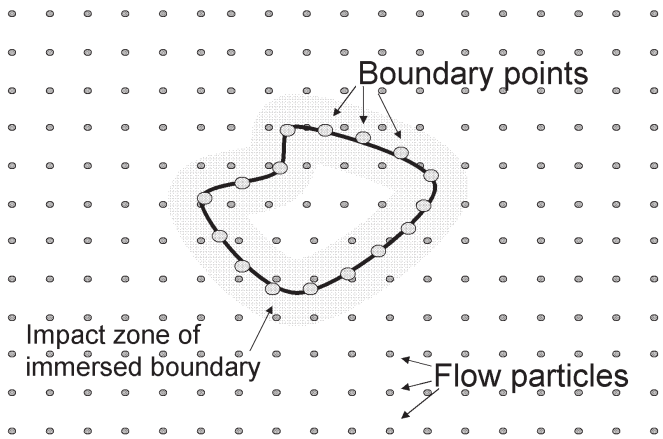


Figure 7.1: Particle Immersed Boundary Method. The immersed boundary is discretized using boundary points that can only impact flow particles within the kernel support.

$\mathbf{u}_{desired}$) yields

$$\mathbf{f}_i = \rho_i \frac{\mathbf{u}_{desired} - \mathbf{u}_i}{\Delta t} - (-\nabla p_i + \nabla.\tau_i) \tag{7.3}$$

Note that the forcing term $\mathbf{f}$ acts locally on the boundaries where a no-slip condition is imposed and the velocity $\mathbf{u}_{desired}$ is known. The boundary is described by boundary points associated with a surface area. We employ a Kernel based on B-Splines for a dirac delta approximation of the forcing term $\mathbf{f}$. We use particle-mesh, mesh-particle interpolation schemes for performance reasons (cf. Section 6.6).

Our implementation involves the separation of the forcing term into two parts, one part $\mathbf{f}_{ip}$ is evaluated on the particles , the other on the boundary points $\mathbf{f}_{ib}$:

$$\mathbf{f}_i = \rho_i(\mathbf{f}_{ip} + \mathbf{f}_{ib}) \tag{7.4}$$

$$\mathbf{f}_{ip} = \frac{-\mathbf{u}_i}{\Delta t} - \frac{1}{\rho_i}(-\nabla p_i + \nabla.\tau_i) \tag{7.5}$$

$$\mathbf{f}_{ib} = \frac{\mathbf{u}_{desired}}{\Delta t} \tag{7.6}$$

The pIBM consists of the following steps:

1. Evaluation of the first part of the forcing term $\mathbf{f}_{ip}$ on the particle
2. Interpolation of $\mathbf{f}_{ip}$ from the particles onto the boundary points via mesh.
3. Evaluate forcing term $\mathbf{f}_i$ on the boundary points by adding $\mathbf{f}_{ib}$
4. Interpolation of forcing term $\mathbf{f}$ from the boundary points to the particles via mesh.
5. Evolving particles according to the governing equations including forcing term $\mathbf{f}$

7.2.2 Particle Equations

The particle position $\mathbf{x}_p$, mass m_p, volume v_p, and velocity component $\mathbf{u}_{i,p}$ evolve by the following system of ordinary differential equations derived from Eq.(4.1),(4.2) and (4.3)

$$\begin{aligned}
\frac{d\mathbf{x}_p}{dt} &= \mathbf{u}_p \\
\frac{dm_p}{dt} &= 0 \\
\frac{dv_p}{dt} &= \langle \nabla \cdot \mathbf{u} \rangle_p v_p \\
\frac{d\mathbf{u}_{i,p}}{dt} &= \frac{v_p}{m_p}\left(-\langle \frac{\partial p}{\partial x_i} \rangle_p + \langle \frac{\partial \tau_{ij}}{\partial x_j} \rangle_p \right) + \mathbf{f}_{i,p}
\end{aligned} \tag{7.7}$$

where $\langle \diamond \rangle_p$ denotes the derivative approximation on a particle p (cf. Eq. (2.20)) and

$$\langle \frac{\partial \tau_{ij}}{\partial x_j} \rangle_p = \mu \left(\langle \frac{\partial^2 u_i}{\partial x_k^2} \rangle_p + \frac{1}{3} \langle \frac{\partial^2 u_l}{\partial x_i \partial x_l} \rangle_p \right) \tag{7.8}$$

$$p_p = \frac{m_p}{v_p} R T_0. \tag{7.9}$$

In the present study, the Laplacian approximation $\langle \frac{\partial^2 u_i}{\partial x_k^2} \rangle_p$ is evaluated using the particle strength exchange approach [49] to increase the stability of the simulations by suppressing spurious small scale structure in the range of the Nyquist frequency.

$$\langle \frac{\partial^2 u_i}{\partial x_k^2} \rangle_p = \sum_p v_p \left(\Phi_p - \Phi_q \right) \nabla^2 \zeta_\epsilon (\mathbf{x}_q - \mathbf{x}_p) \tag{7.10}$$

$$\zeta_\epsilon = \frac{15}{\epsilon^{-3}\pi} \frac{1}{|\mathbf{x}|^{10} + 1} \tag{7.11}$$

The second order kernel ζ_ϵ was successfully applied in diffusion simulations involving complex geometries [140].

The interface between the body and the fluid is captured using the Particle Level Set Method presented in Chapter 3. The level set function [81, 82] represents the signed distance function to the interface. The particles carry the level set information as a scalar attribute Φ_p that remains constant during the time integration:

$$\frac{d\Phi_p}{dt} = 0 \tag{7.12}$$

We reinitialize the level set value after every remeshing to maintain the signed distance property. The exact knowledge of the body shape allows the reinitialization of the level set function with its analytical value.

The inlet and outlet boundary conditions are imposed by using image particles that have the same attributes as the flow particles. The boundary particles interact with the flow particles such that the boundary conditions are satisfied. The no-slip boundary condition on the body surface is handled by the proposed particle Immersed Boundary Method.

7.3 Results

We demonstrate the performance of the presented Immersed Boundary Method on several test problems and compare with results presented in the literature. We consider the Poiseuille flow, flow past a cylinder and sphere and anguilliform swimming. The comparison is based on the drag/lift coefficients and the Strouhal numbers.

7.3.1 Poiseuille flow

A classic, and simple, problem in viscous, laminar flow involves the steady-state velocity and pressure distribution for a fluid moving laterally between two plates. The flow is driven by a pressure gradient in the direction of the flow, and is retarded by viscous drag along both plates, such that these forces are in balance. The simulation domain was considered to be a unit square with periodic boundary condition at the inlet/outlet boundary

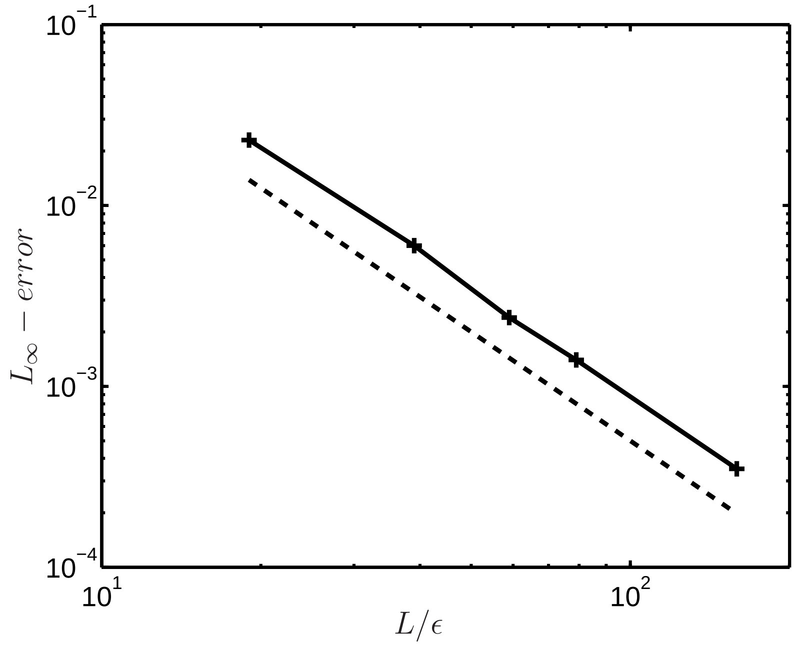


Figure 7.2: Poiseuille flow: $L_\infty-$ error velocity (solid line) of the particle Immersed Boundary Method (pIBM) compared to 2nd order scaling (dashed line)

(x=0, x=1) and no-slip conditions at the plates (y=0, y=1). We consider a fluid density with an initial density of $\rho = \rho_0 = 1$ at a Reynolds number of $Re = 100$ and a Mach number of $M = 0.5$. The fluid is initially at rest and accelerated by an constant pressure gradient of 0.001. The time integration scheme is a Runge Kutta scheme of 2nd order with a time step of $\delta t = 0.0005$ in all cases, i.e. the Courant-Friedrich-Lewy number $CFL = c_0\delta t/h < 0.2$ in all cases. For the error analysis the maximal difference in the velocity profile to the analytical solution is evaluated when the profile becomes stationary at time $T = 70$. The error normalized by the maximal velocity is shown in Fig. 7.2. The error analysis shows the flow simulations are second order accurate in space.

7.3.2 Flow past a cylinder

We present the simulation of flow past a cylinder for various Reynolds numbers to demonstrate the performance of the particle Immersed Boundary Method and compare with previous results presented in the literature. The flow past a circular cylinder is associated with various instabilities. These instabilities involve the wake, the separated shear layer and the boundary layer depending on the Reynolds number. Up to $Re \approx 47$, the flow is steady with two symmetric vortices on each side of the wake centre line. The first wake instability, manifestation of a Hopf bifurcation, occurs at $Re \approx 47$. For $Re > 47$, although remaining laminar, the flow becomes unsteady and asymmetric. Von Karman vortex shedding is observed for slightly larger Re. At $Re \approx 190$, three-dimensional instabilities, such as formation of vortex loops, deformation of primary vortices and stream wise and span wise vortices appear in wake. The wake flow undergoes a series of complex three-dimensional instabilities, making the flow eventually turbulent. Beyond a certain critical Re, the shear layer separating from the upper and lower surface of the cylinder starts becoming unstable via the Kelvin-Helmholtz mode of instability. The transition point, beyond which the separated layer becomes unstable, moves upstream with the increase of the Reynolds number. At $Re \approx 2 \cdot 10^5$, the boundary layer on the cylinder surface undergoes a transition from laminar to turbulent.

We use the Runge Kutta 4th order scheme for time integration with constant time step of $\Delta t = 0.005$. The domain size is set to $15d \times 30d$ where d is the diameter of the cylinder. The Mach number M is 0.05 The solution is remeshed after every time step. The fluid is initially at rest and accelerated by a small artificial force until the desired inlet velocity is reached to avoid the development of pressure wave at the boundary.

Fig. 7.3 and Fig. 7.4 shows the vorticity field at $Re = 100$ and $Re = 1000$ respectively. The instability is trigged by a pertubation of the inlet velocity in lateral direction as described by Plouhams [128]. The vorticity field match well with the Finite Element solutions presented by Singh *et al.* [153] in both cases. The simulations require a particle spacing of $h = 0.078d$ for $Re = 100$ and $h = 0.052d$ for $Re = 1000$.

Fig. 7.5 shows the variation of the drag coefficient with the Reynolds number. The simulation results of the pIBM are compared with experimental results and computations considering incompressible flow. It is observed that the values from present computations match well the experiments for $Re < 200$. In particular, Table 7.1 shows the excellent

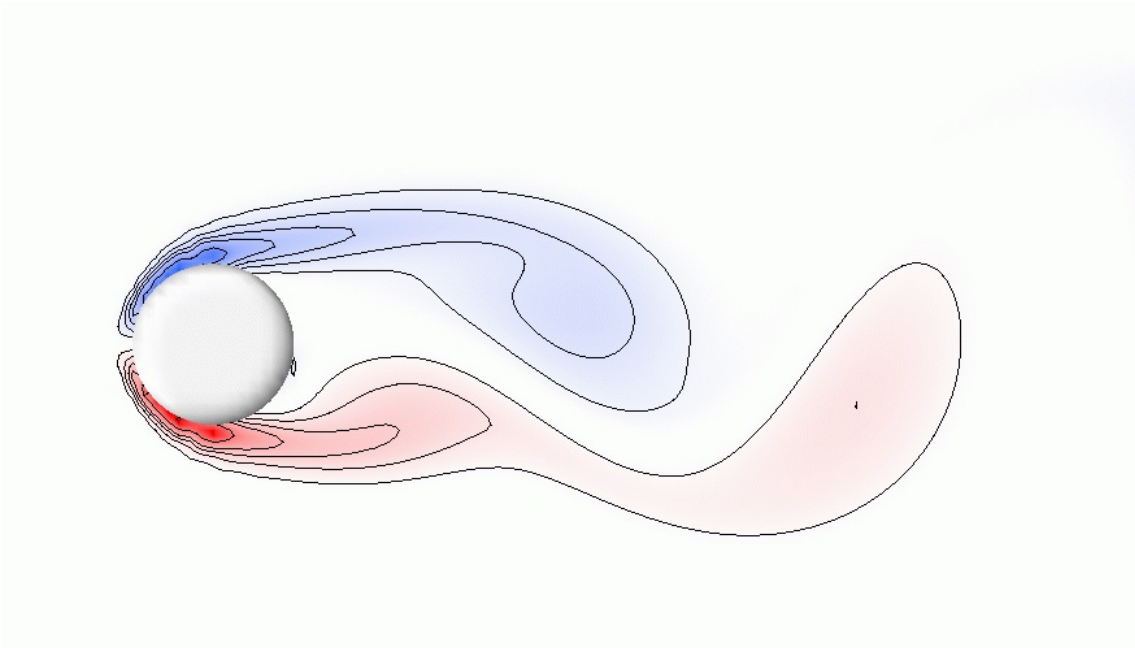


Figure 7.3: Flow past a cylinder at $Re = 100$*. Contour levels of the vorticity contours at* ($\pm 20, \pm 15, \pm 10, \pm 7.5, \pm 5, \pm 2.5$)

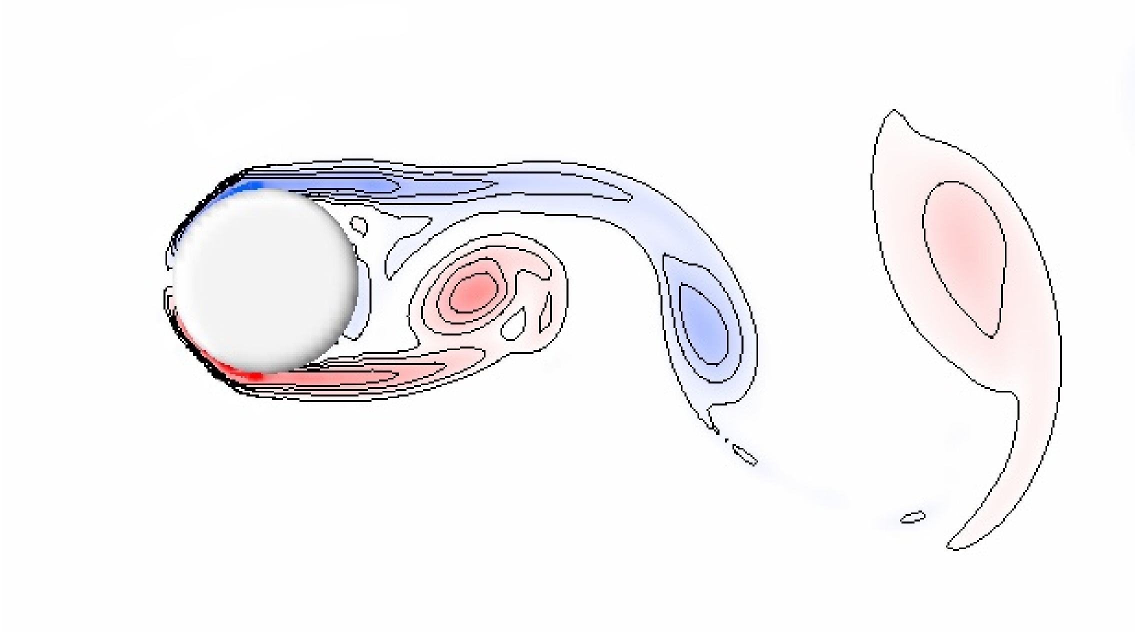


Figure 7.4: Flow past a cylinder at $Re = 1000$*. Contour levels of the vorticity contours at* ($\pm 20, \pm 15, \pm 10, \pm 5$)

agreement to experiments and previous simulation results with respects to drag coefficient and Strouhal number.

Beyond $Re = 180$ the wake flow undergoes three-dimensional transitional instabilities. Therefore, for $Re > 200$ the drag coefficient and the Strouhal number are overpredicted by two-dimensional computations.

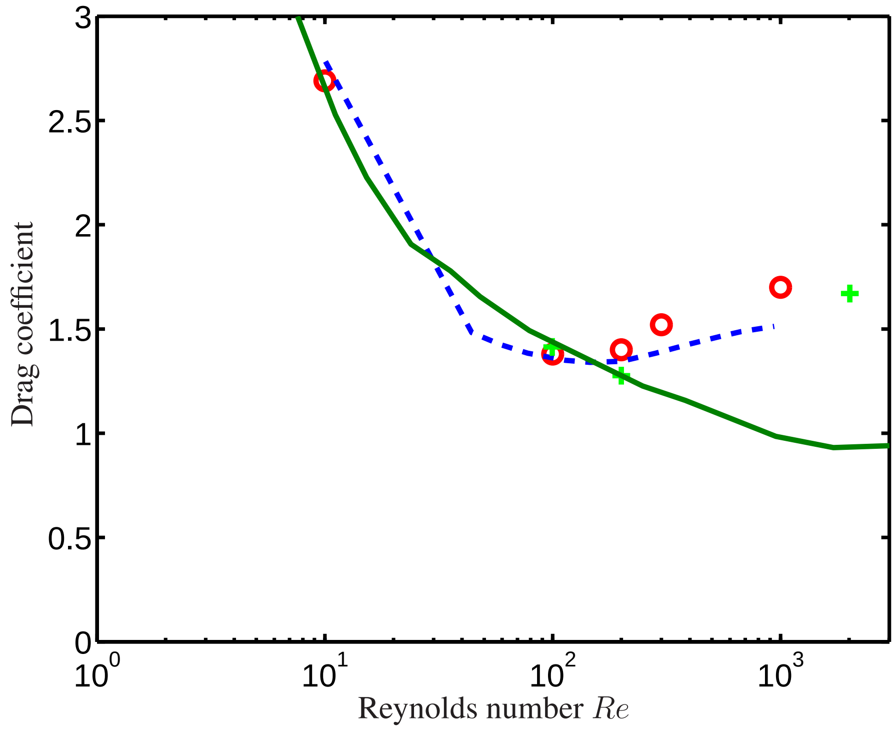


Figure 7.5: Flow past a cylinder: Time averaged drag coefficient of pIBM (circles) versus Reynolds number in comparison with experimental data (solid line, taken from [136]) , a Spectral Method (dashed line, taken from [153]) and a FEM solution (crosses) [153]

Fig. 7.6 shows the pressure coefficient of the time averaged flow along the cylinder surface for $Re = 100$ compared to the result of Park *et al.* [123]. The angle θ is measured from the stagnation point of the incoming flow towards the outlet. The pressure coefficient C_p agrees well with the results of Park *et al.* [123]. The pressure field tends to reveal noise at the angle $10° < \theta < 50°$ due to unresolved pressure waves in the compressible fluid.

Table 7.1: Flow past a cylinder: Comparison with previous simulations and experiments

$Re = 100, D = 0.2$	Drag coefficient	Strouhal number
pIBM	1.38	0.162
Henderson [80]	1.35	-
Park *et al.* [123]	1.33	0.165
Silva *et al.* IBM [152]	1.39	0.162
Singh *et al.* FEM [153]	1.41	0.164
Kim *et al.* FV IBM [93]	1.33	0.165
Wieselberger (Exp.) taken from [136]	1.45	-
Williamson (Exp.) [181]	-	0.165

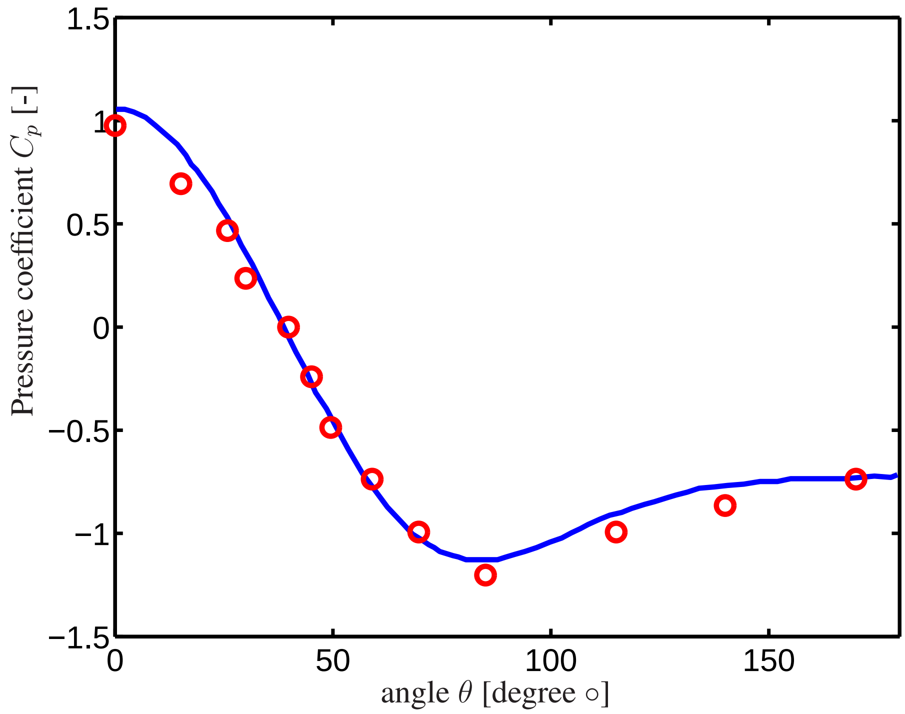


Figure 7.6: Flow past a cylinder at $Re = 100$: Pressure coefficient C_p (circle) versus angle in comparison with the solution of Park [123] (solid line)

Table 7.2: Flow past a sphere: Comparison with previous simulations

$Re = 100$	Drag coefficient	Lift coefficient	Strouhal number
pIBM (M =0.1)	1.15	-	-
Fornberg [61]	1.09	-	-
Kim *et al.* FV IBM [93]	1.09	-	-
Fadlun *et al.* [59]	1.08	-	-
$Re = 300$			
pIBM (M =0.1)	0.71	0.062	0.133
Johnson and Patel [88]	0.66	0.069	0.137
Kim *et al.* FV IBM [93]	0.66	0.067	0.134
Ploumhans *et al.* [128]	0.68	0.066	0.137

7.3.3 Flow past a sphere

Wakes of incompressible fluid behind spheres are observed to be steady for Reynolds numbers below 270. Above this limit vortices break off and are periodically released to form vortex loops that are connected like in a chain. We consider the flow past a sphere at $M = 0.1$ and $Re = 100$ and $Re = 300$. Table 7.2 shows that the drag and lift coefficient of the pIBM compare well with the simulation results considering incompressible fluid showing a discrepancy in the range of 5% to 10%. The domain size is $10d \times 10d \times 15d$, the particle spacing $h = 0.052d$ where d is the diameter of the sphere. The spacing of the boundary points is in average the same. The time integrator is Runge Kutta 4 using a time step of $\Delta t = 0.005$. Fig. 7.7 shows the three-dimensional vorticity structure at $Re = 300$. The surface of the vortices is identified by the λ_2 method of Jeong and Hussain [87]. At $Re = 300$ the flow is unsteady and the vortices shed asymmetrically. This flow behavior matches with the results of Johnson and Patel [90]. The agreement in the flow structure, as well as in the drag and lift coefficients indicates that the present method accurately captures the three-dimensional vorticity field.

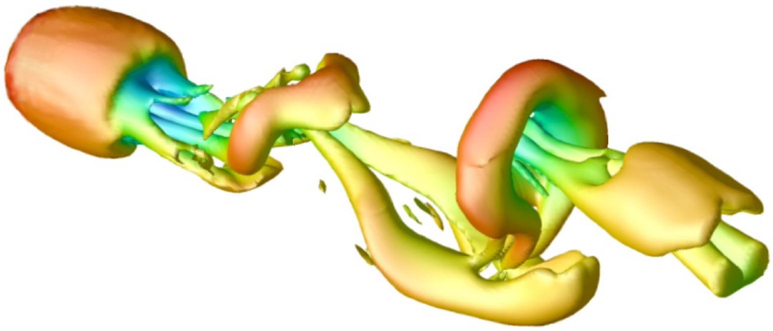


Figure 7.7: Flow past a sphere at $Re = 300$. The vortices behind the sphere are visualized using the λ_2 method [87]. The color represent the local flow velocity.

7.3.4 Falling Sphere

The problem of a falling sphere is a simple test case involving fluid-structure interactions. Kern *et al.* [92] presented this test case to validate the fluid-body coupling procedure. We consider a rigid sphere of density $\rho_s = 1.041 > \rho_0$ at Reynolds number $Re = 100$ and at Mach number of $M = 0.25$. The sphere is released from rest and accelerates until it reaches its asymptotic falling velocity. The sphere diameter d is set to $d = 1$ and the gravity $g = 20$. The size of the periodic domain is set to $6 \times 20 \times 6$, the time integration is Runge Kutta 2nd order with a time step of $\Delta t = 0.001$. Remeshing is applied every time step. An asymptotic falling velocity of $\overline{U} = 0.95$ is reached at time $t = 10$ using a particle spacing of $1/16$. This velocity is in reasonable agreement with the results of Johnson and Patel [88]. Table 7.3.4 summarizes the results of the falling sphere.

Table 7.3: Falling sphere: Convergence study of the falling velocity

Particle spacing h) ($\Delta t = 0.001$)	Falling velocity (t=10)	Time step Δt ($h = 1/16$)	Falling velocity (t=2)
1/8	1.02	0.004	0.602
1/16	0.95	0.002	0.592
1/32	0.93	0.001	0.596
Johnson *et al.* [88]	1.00	0.0005	0.595

7.4 Simulation of Anguilliform Swimming

Complex structures interacting with ambient fluids appear in many biological systems. To demonstrate the performance of the pIBM approach, we present the simulation of anguilliform swimming of a self-propelled eel-like body immersed in a viscous fluid. Anguilliform swimmers, such as lamprey, propel themselves by propagating curvature waves backwards along the body. We compare the simulations results with an incompressible finite-volume solution presented by Kern *et al.* [92]. The solution of Kern *et al.* is second order accurate in space and first order accurate in time using an adaptive grid.

7.4.1 Introduction

The motion of the body is described by the two-dimensional deformation of the mid-line based on the simulations of Carling *et al.* [26]. The lateral displacement of the mid-line $y_s(s,t)$ in a local system is defined as

$$y_s(s,t) = 0.125\frac{s/L + 0.03125}{1.03125}\sin(2\pi(s/L - t/T)) \tag{7.13}$$

where s is the arc length along the mid-line of the body ($0 \leq s \leq L$), t is the time, T the periodic time.

The three dimensional body of the swimmer is described by spatially varying ellipsoid cross sections. The length of the two half axis $w(s)$ and $h(s)$ are defined as

$$w(s) = \begin{cases} \sqrt{2w_h s - s^2} & 0 \le s \le s_b \\ w_h - (w_h - w_t)\left(\frac{s-s_b}{s_t-s_b}\right)^2 & s_b \le s \le s_t \\ w_t \frac{L-s}{L-s_t} & s_t \le s \le L \end{cases} \tag{7.14}$$

$$h(s) = b\sqrt{1 - \left(\frac{s-a}{a}\right)^2} \tag{7.15}$$

where $w_h = s_b = 0.04L$, $s_t = 0.95L$, $w_t = 0.01L$, $a = 0.51L$ and $b = 0.08L$. We apply a no-slip boundary condition on the surface of the body. The mid-line of the body is embedded into a non-inertial (x', y')-system where the center of mass of the deforming body remains and the total angular momentum is conserved. The fluid-body interactions are computed in the inertial system (x,y,z) where the swimmer is subject to rigid body translation and rotation. Thus, the motion of the body in the global system (O, x, y, z) is described by the Newtons equations of motion:

$$m\ddot{x}_c = F, \tag{7.16}$$

$$\dot{I}_z\dot{\varphi}_c + I_z\ddot{\varphi}_c = M_z, \tag{7.17}$$

where m is the total mass of the immersed body, x_c represents the position of the center of mass, φ_c the global angle with respect to the initial position, F and M_z are the fluid force and yaw torque acting on the body surface. The time-dependency of the inertial moment $\dot{I}_z$ about the yaw axis is also taken into account although it is small compared to the inertial moment itself.

We set the viscosity of the fluid to be $\mu = 1.4 \cdot 10^{-4}$, the body length $L = 1$, the density $\rho_{0,fluid} = \rho_{body} = \rho = 1$ resulting in a Reynolds number of 3850 based on the final swimming speed.

The fluid forces acting on the body are shown as non-dimensional coefficients $C_\| = F_\|/(0.5\rho U_0^2 S)$ and $C_\perp = F_\perp/(0.5\rho U_0^2 S)$ parallel and lateral to the swimming direction, where S represents the circumference in two-dimensions and the surface of the body in three dimensions. The yaw torque is measured by the non-dimensional coefficient $C_M = M_z/(0.5\rho U_0^2 LS)$.

7.4.2 Equations of the Anguilliform Swimmer

The position $\mathbf{x}_c$ and the angle φ_c of anguilliform swimmer evolve by the following set of equations based on Eqs. (7.16) and (7.17)

$$\begin{aligned}
\frac{d\mathbf{x}_c}{dt} &= \mathbf{u}_c, \\
\frac{d\mathbf{u}_c}{dt} &= \frac{\mathbf{F}}{m}, \\
\frac{d\varphi_c}{dt} &= \omega_c, \\
\frac{d\omega_c}{dt} &= \frac{M_z - \dot{I}_z\omega_c}{I_z},
\end{aligned} \tag{7.18}$$

where u_c denotes the velocity of the swimmer and ω_c the angular velocity. We solve this set of equations simultaneously with the particle equations (Eq. 7.7-7.12) that describe the fluid behavior.

7.4.3 Computational Setup

The particles are initially distributed uniformly in the domain and remeshed every time step. We integrate the Eqns.(7.7)-(7.12), and (7.18) with respect to time using a explicit 4th order Runge-Kutta scheme with time step of $\Delta t = 0.001$. We consider the domain as an noninertial coordinate system that moves with the opposite x_1-component of the fish velocity such that x_1-position of the fish is constant in the noninertial coordinate system (cf. Appendix C). Thus, we accelerate the fluid in x_1-direction by the opposite force that acts on the swimmer and the swimmer remains on its x_1-position. We impose an inlet and an outlet condition to the boundary ahead and rear of the swimming body, respectively. This approach enables us to reduce the computational effort significantly because our particle solver is currently limited to a uniform resolution. The size of the domain is 4×2 in two dimensions and $3 \times 2 \times 2$ in three dimensions. This domain size is tested to be sufficiently large to neglect the influence of the boundary.

The simulations are based on $1.3 \cdot 10^5$ particles in two dimensions, and $2.5 \cdot 10^7$ particles in three dimensions.

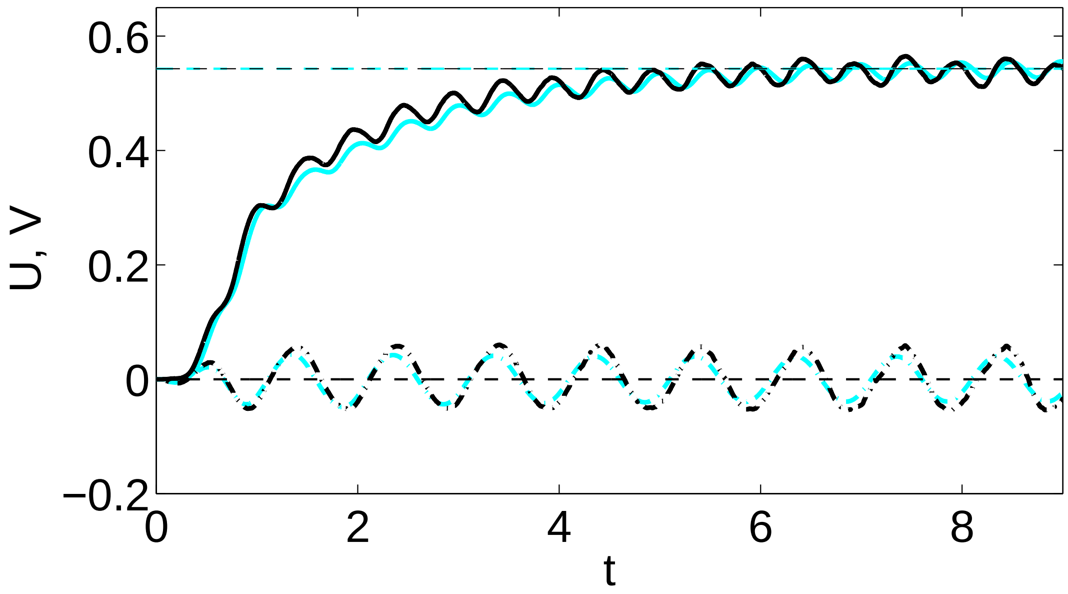


Figure 7.8: Longitudinal (solid line) and lateral velocity (dashed line) of the two dimensional swimmer compared to finite volume solution (light blue) [92]

7.4.4 Results

Two-Dimensional Anguilliform Swimmer

We present a comparison in the two dimensional case with the work of Kern *et al.* [92] in terms of velocity of the swimmer, as well as forces and torque acting on the swimmer. The swimmer accelerates from rest to an asymptotic mean forward velocity of $\bar{U}_{\parallel} = 0.54$ in about seven undulation cycles. The velocity varies slightly during a cycle while the lateral velocity $U_{\perp}$ has an amplitude of 0.04. The time history of the longitudinal and lateral velocity agrees very well with the incompressible solution (Fig. 7.8). The velocity differs the most at time $1 < t < 4$ where the density variations are larger than at later time steps. The higher density variations lead to higher pressure variations resulting in larger forces acting on the swimmer. The incompressible solution is approximated sufficiently with a Mach number of $M = 0.1$.

The longitudinal and lateral forces and the torque (Fig. 7.9) agree very well with the incompressible solution. The force and moment coefficient $C_{\parallel}, C_{\perp}$ and C_M converge to oscillation modes with zero mean and a constant amplitude of 0.03, 0.04 and 0.03, respectively.

Kern *et al.* [92] applied a low pass filter to the fluid force F and the torque M_z to stabilize the simulation of the incompressible flow. We can omit the use of a low pass filter in our study. The compressibility of the fluid causes unresolved pressure waves resulting in high frequent noise in the flow structure. A second order filter [125] is applied to the mass and the momentum during the remeshing process every 100 steps to suppress the small scale pressure waves in the range of the Nyquist frequency. A drawback of the simulation, however, is the remaining noise in the pressure field resulting in noisy forces acting on the swimmer that remains even when applying the low pass filter presented by Kern *et al.* [92].

Fig. 7.10 and 7.11 show the vorticity field of the swimmer during one period at the final swimming speed. The main differences in the vorticity field result from the fact that the particle solution is uniformly resolved, whereas the finite volume solution involves an adaptive re-gridding. Thus, the vorticity shedding at the boundary layer is better resolved in the finite volume solution, the wake pattern behind the tail in the particle solution.

The tail beat amplitude is $A = 0.16$ and the corresponding Strouhal number is $St = 0.59$. The wave velocity is $V = 0.73$, which results in a slip of $\bar{U}_{\|}/V = 0.74$.

Three-Dimensional Anguilliform Swimmer

In three dimensions the forces acting on the fish compare well with the finite volume solution (Fig. 7.13). The net force and moment coefficient $C_{\|}, C_{\perp}$ and C_M oscillate with a mean of zero and amplitudes of 0.04, 0.06 and 0.03, respectively. The final swimming speed in the particle solution ($u_{pIBM} = 0.448$) is 12% higher than the result in the finite volume solution ($u_{FV} = 0.402$). This result matches well with the drag comparison in the flow past a sphere at $Re = 300$ where the drag coefficient differs approximately 10% from the results of grid based methods (Table 7.2). The forward velocity $U_{\|}$ oscillates with an amplitude of 0.01. The lateral velocity $U_{\perp}$ has a zero mean and an amplitude of 0.03. The wave velocity $V = 0.73$ is equal to the two dimensional case resulting in the slip of

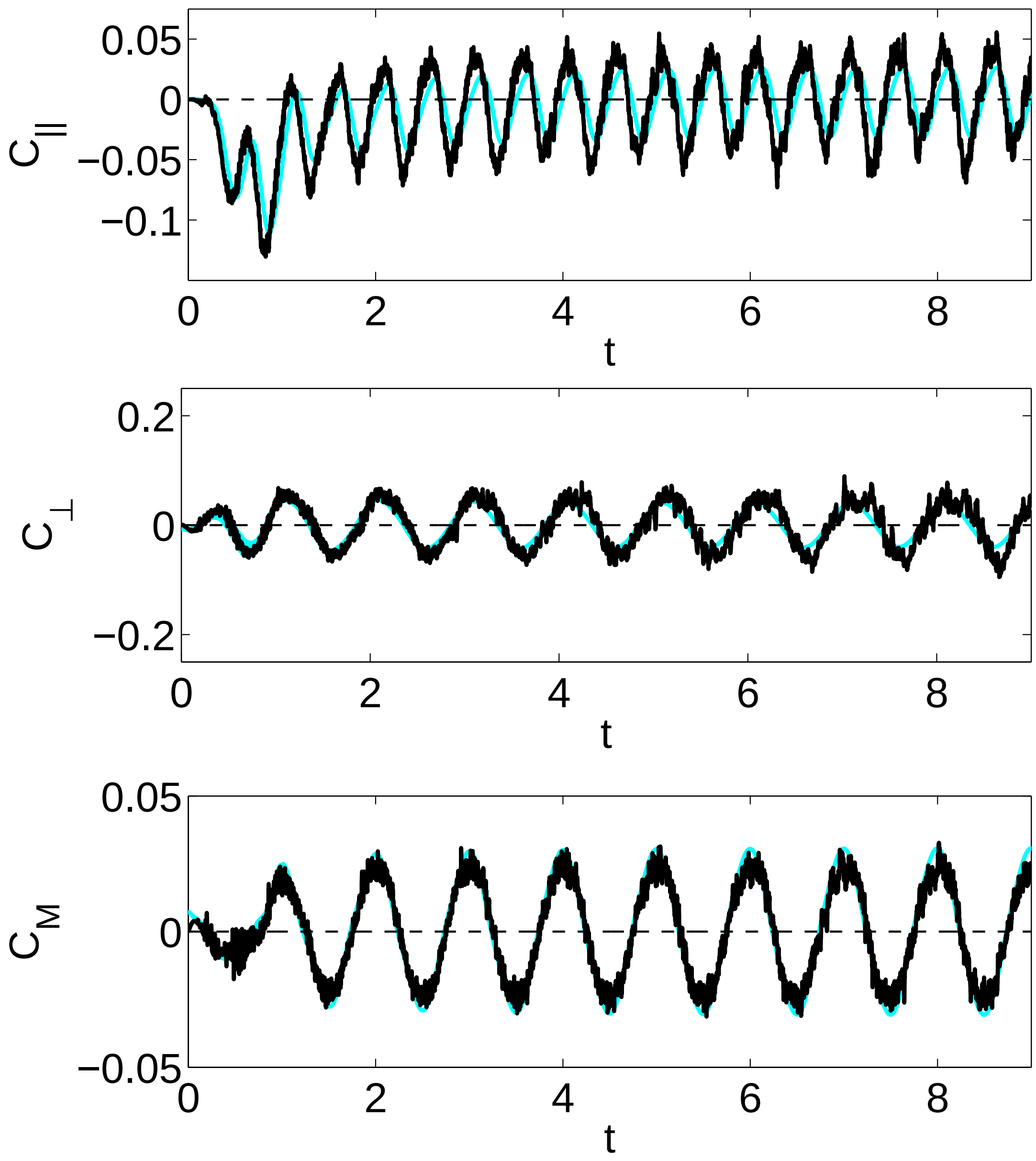


Figure 7.9: Longitudinal force $C_{\|}$, lateral force $C_{\perp}$ and torque C_M of the two dimensional swimmer (black) compared to finite volume solution (light blue) [92]

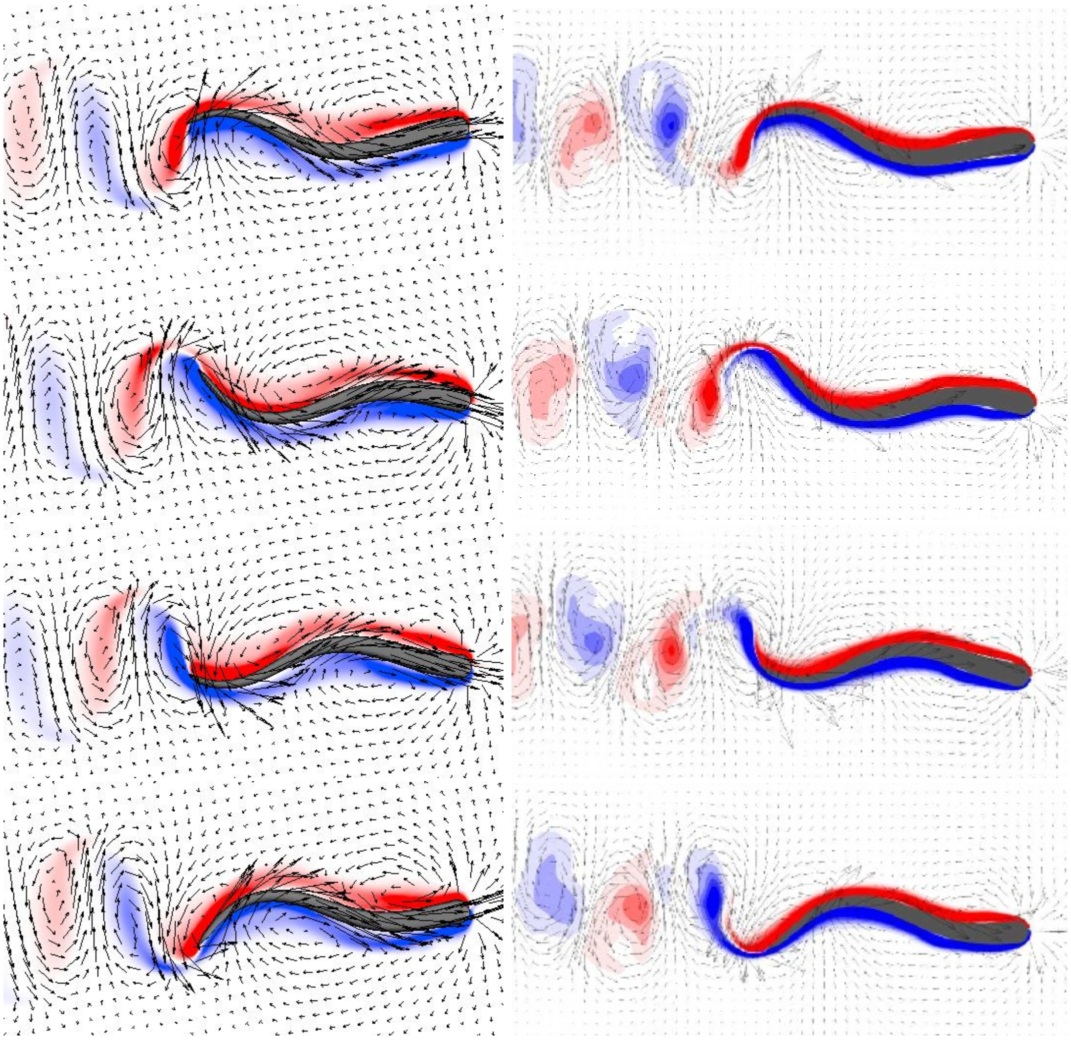


Figure 7.10: Vorticity field of the two-dimensional swimmer using pIBM (left) and the reference solution [92] (right) for one swmming cycle at time t, t+0.25T, t+0.5T, t+0.75T

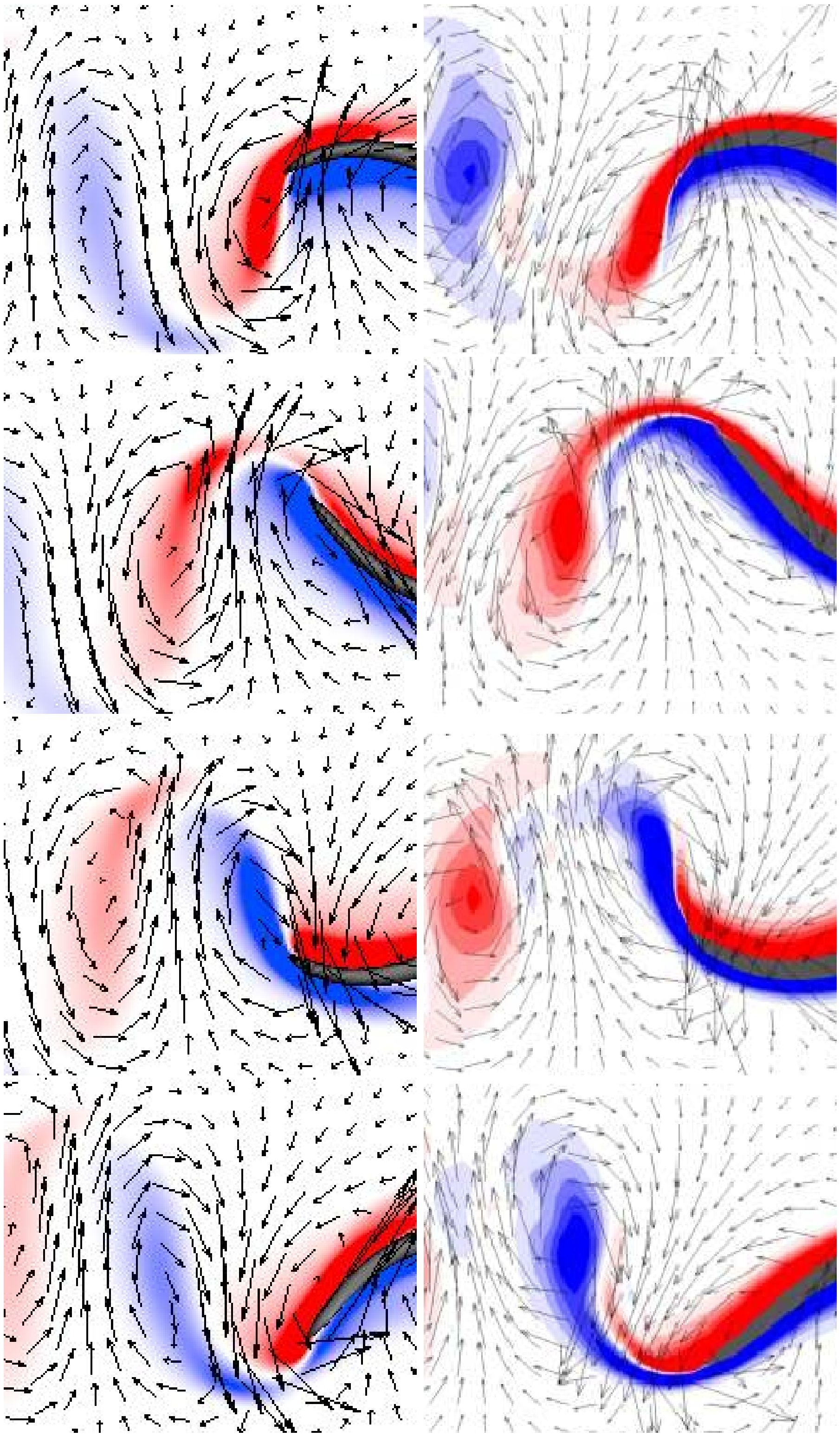


Figure 7.11: Zoom of the vorticity field at the tail of the two-dimensional swimmer using pIBM (left) and the reference solution [92] (right) for one swmming cycle at time t, t+0.25T, t+0.5T, t+0.75T

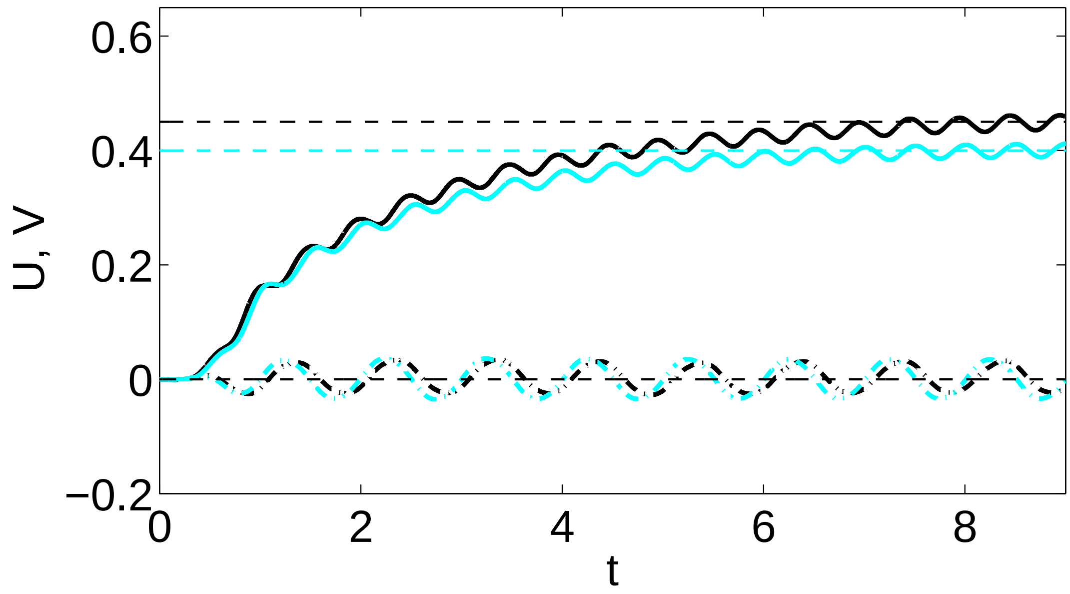


Figure 7.12: Longitudinal (solid line) and lateral velocity (dashed line) of the three dimensional swimmer compared to finite volume solution (light blue) [92]

$\bar{U}_{\|}/V = 0.61$. The tail beat amplitude is determined to be $A = 0.15$ with $St = 0.67$.

The oscillating tail of the swimmer creates a three-dimensional vortex shedding in the frequency of the swimming motion (Fig. 7.14-7.16). Both, the particle and the finite volume solution show the vorticity shed in every half tail beat cycle that breaks up into two vortices forming lateral jets. The vorticity field of particle solution appears smoother and shows less small-scale structures. The vortex rings are less recognizable. As the finite-volume grid feature a four times higher resolution in the boundary layer of the tail than the particle solution, the absence of the small-scale structures in the boundary layer can be associated with a lack of resolution. The small vorticity structures between the shedding vortex pair result are mainly spurious and result from the highly dynamic refinement of the finite volume grid.

Overall, the agreement between the particle and the finite volume solutions is good and shows that the pIBM is appropriate to solve flow-structure interactions accurately.

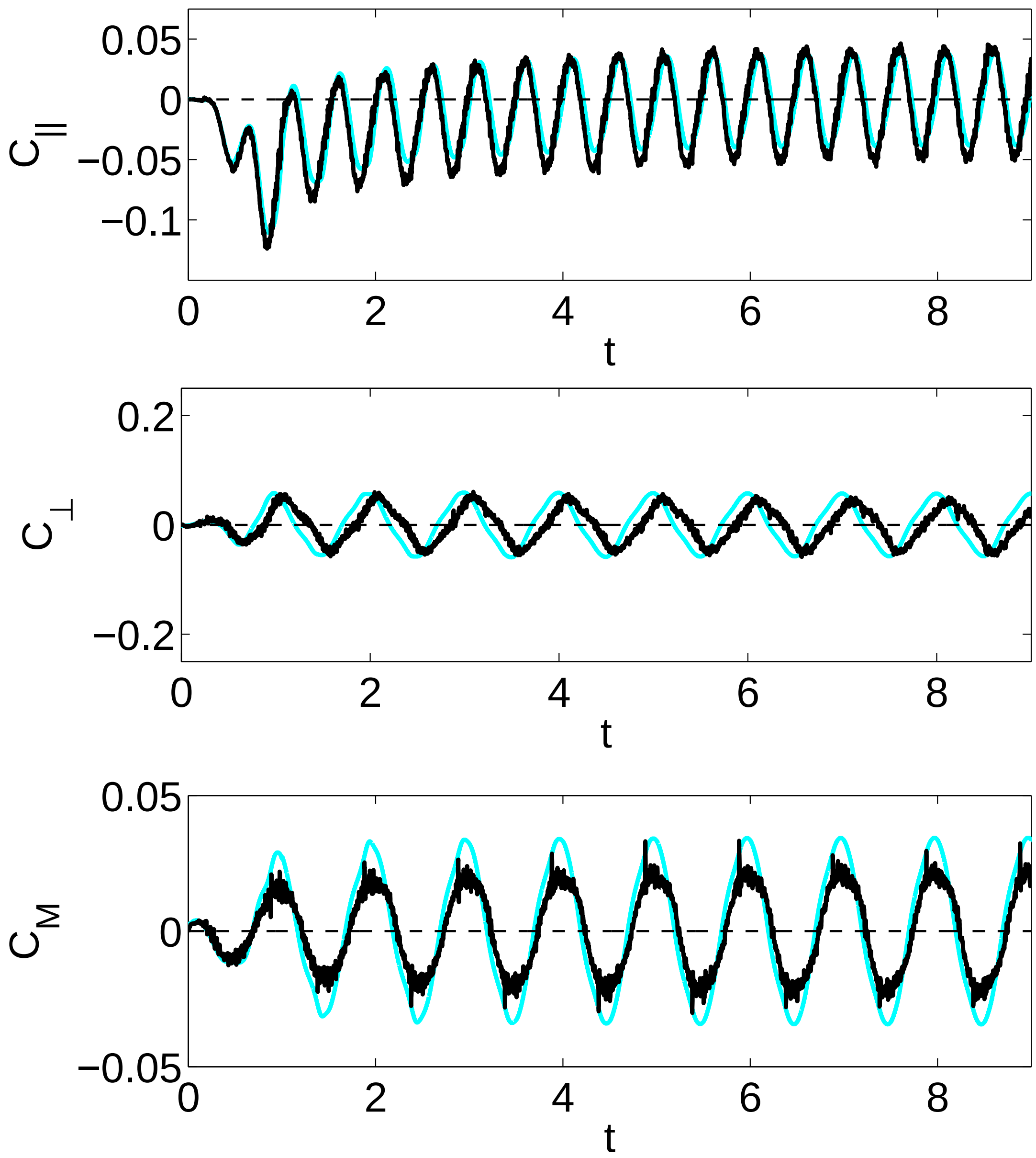


Figure 7.13: Longitudinal force $C_{\parallel}$, lateral force $C_{\perp}$ and torque C_M of the three dimensional swimmer (black) compared to finite volume solution (light blue) [92]

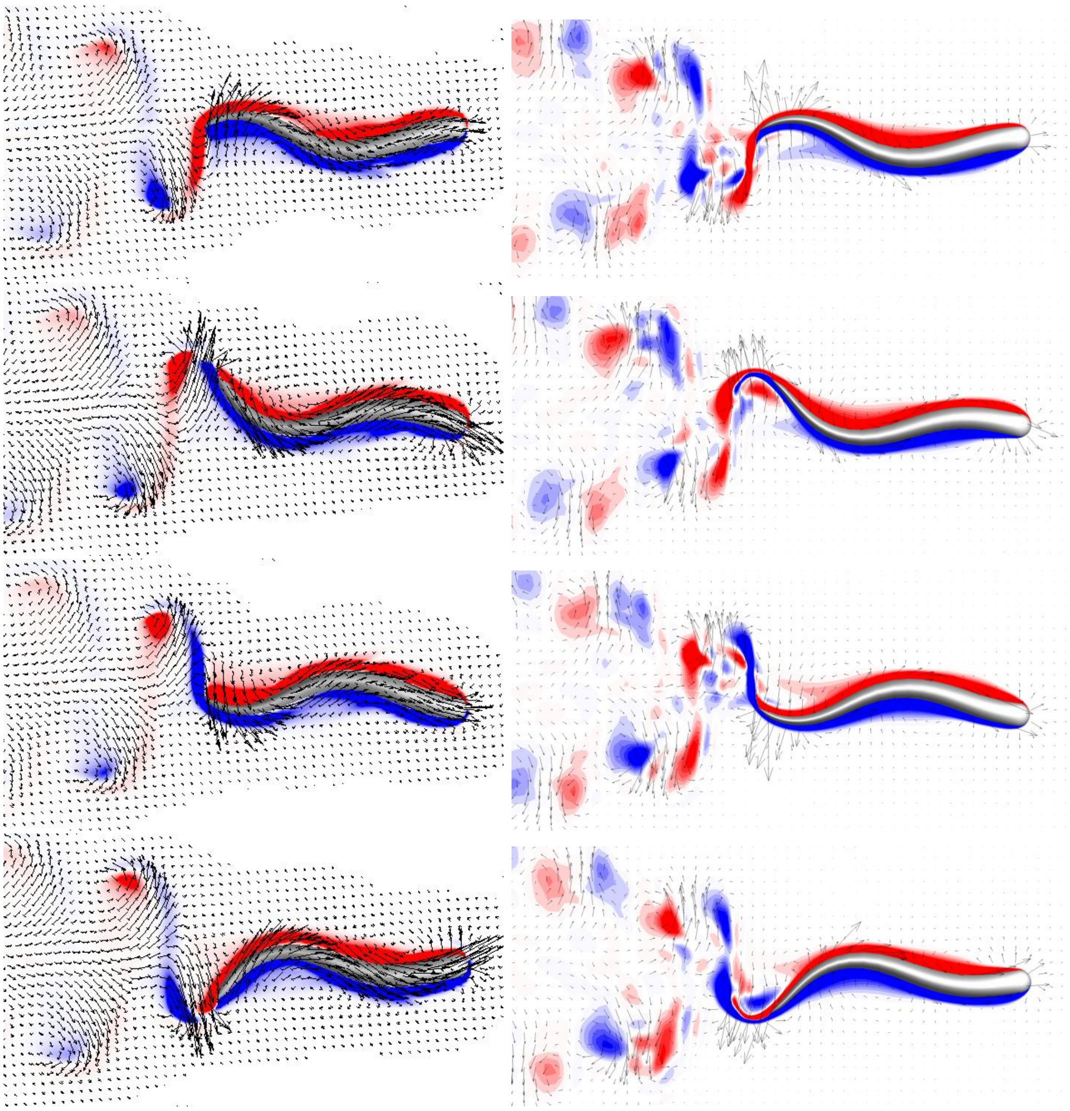


Figure 7.14: Vorticity field of the three-dimensional swimmer using pIBM (left) and the reference solution [92] (right) for one swimming cycle at time t, t+0.25T, t+0.5T, t+0.75T

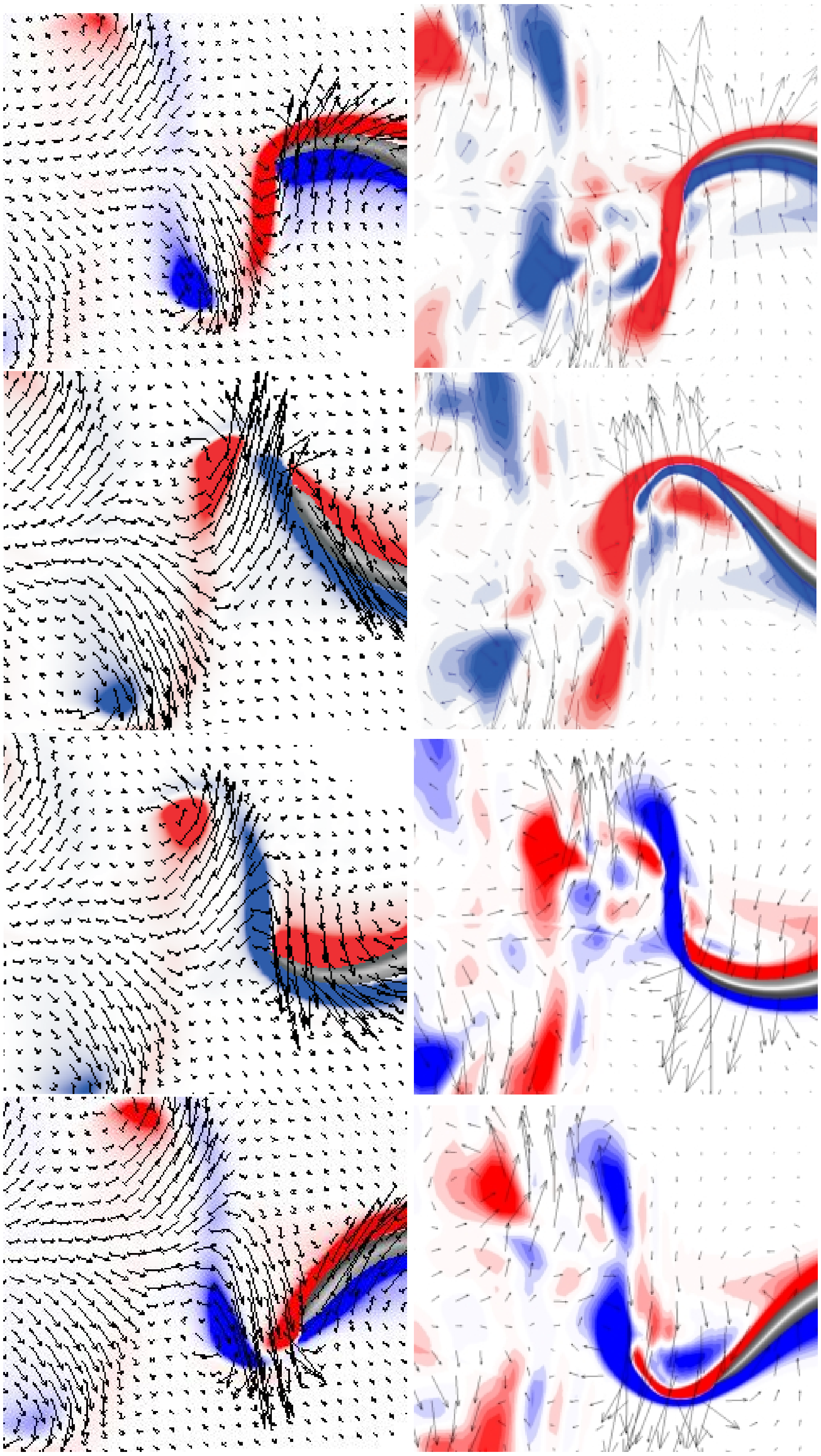


Figure 7.15: Zoom of the vorticity field at the tail of the three-dimensional swimmer using pIBM (left) and the reference solution [92] (right) for one swmming cycle at time t, t+0.25T, t+0.5T, t+0.75T

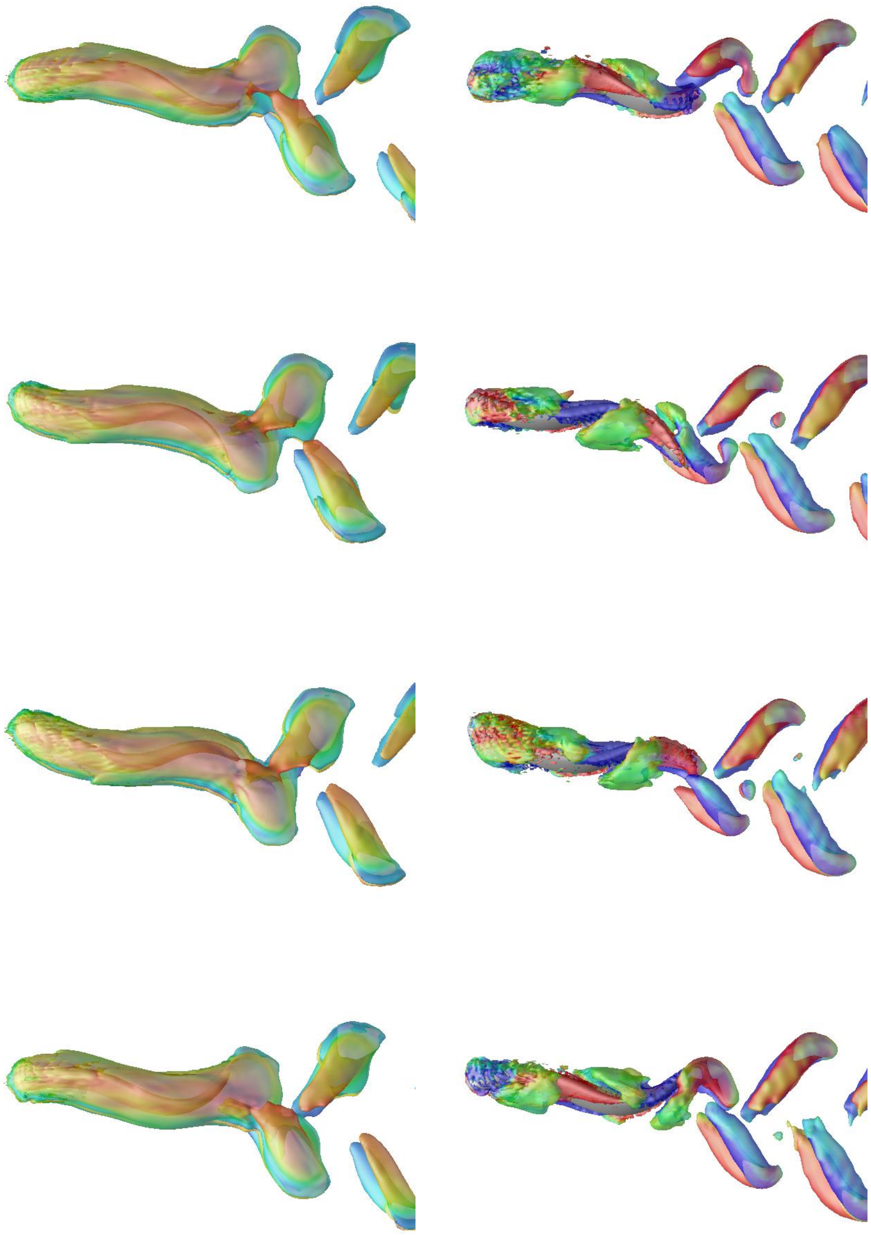


Figure 7.16: Isosurface of the vorticity magnitude (left) and vortices visualized by the λ_2-method (right) of the three-dimensional swimmer using pIBM for one swmming cycle at time t, t+0.25T, t+0.5T, t+0.75T

Chapter 8

Fluid-Solid Interactions

8.1 Introduction

In many biological systems, fluid-solid interactions are of special interest. Their numerical simulation is essential, particularly, in virtual surgery because approximately 80% of the human body consists of water and and most vital organs are filled with fluid. However, the coupling of the fluid-solid models causes several numerical problems. Often, the governing equations of the fluid-solid system are solved by different numerical methods making the coupling technically difficult because of the conservation requirement of certain quantities. Thus, fluids are commonly simulated using finite volume or particle methods whereas elastic solids are usually analyzed by using finite element methods. An additional problem is the moving interface between the fluid and solid domain, which typically requires complex mesh movements and adaption. A classical way to solve fluid-structure interactions are Arbitrary Lagrange Euler (ALE) methods where the solid is treated in a reference frame formulation and the fluid is solved in a Lagrangian way. However, this approach can be problematic when the deformations are large due to the excessive mesh distortion [154, 33]. Methods based on the concept of immersed boundaries offer promising alternatives to ALE methods because they do not require complex grids to resolve the fluid-solid interface. Thus, boundary conditions are translated into forcing terms in the governing equations. Following the idea of immersed boundaries, Cottet [39] has developed a one-dimensional particle model based on a unified conservation equation for the solid-fluid problem. A key aspect of this approach is the implicit treatment of the interface which avoids an explicit tracking of the interface. In this chapter, we consider a Smoothed Particle Hydrodynamic (SPH) solution of the particle model presented in [107]. The implementation uses a remeshing scheme (rSPH) to ensure the convergence of the method [29]. The unified formulation contains anisotropic diffusion terms at the interface which we solve numerically using the method of particle strength exchange (PSE) [192] or using isotropic one-sided differentiation. We expand the model to cover the compressible Navier-Stokes in one- and two dimensions. The model expansion requires the consideration of the continuity equation and the pressure gradient in the momentum equation.

8.2 Particle Model of Cottet

The model of Cottet [39] describes the fluid by the Burger's equation and the solid by a one-dimensional wave equation. The Burger's equation is a one-dimensional approximation of the Navier Stokes equations which neglects pressure gradients. The fluid velocity u and the solid displacement d are governed by

Burger's equation (fluid)

$$\frac{\partial u}{\partial t} + \frac{3}{2} u \frac{\partial u}{\partial x} - \mu\rho \frac{\partial^2 u}{\partial x^2} = 0 \qquad \text{for } x \in [-0.5, \gamma(t)] \tag{8.1}$$

One-dimensional wave equation (solid)

$$\frac{\partial^2 d}{\partial t^2} - E \frac{\partial^2 d}{\partial \xi^2} = 0 \qquad \text{for } \xi \in [0, 0.5] \tag{8.2}$$

Interface condition: equilibrium of stress at $\gamma(t) = d(0,t)$

$$\mu \frac{\partial u(\gamma(t), t)}{\partial x} = E \frac{\partial d(0,t)}{\partial \xi} \tag{8.3}$$

where μ denotes the fluid viscosity and E represents the elasticity coefficient. ξ is the Lagrangian variable for the solid which is related to the Eulerian coordinate by $x(\xi,t) = \xi + d(\xi,t)$.

A unified formulation of the system of equations (Eq. (8.1 - 8.3)) was derived by Cottet [39]. As a result, both materials, the fluid and the solid, are described by one single conservation equation which is define for all $x \in [-0.5, 0.5]$ and $\xi \in [0, 0.5]$:

$$\begin{aligned}\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} =& -\frac{1}{2}\chi^F u \frac{\partial u}{\partial x} \\ &+ \left(\chi^F + \frac{\partial x}{\partial \xi}\chi^s\right)\frac{\partial}{\partial x}\left(\mu\frac{\partial u}{\partial x}\chi^F + E\frac{\partial d}{\partial \xi}\chi^S\right).\end{aligned} \tag{8.4}$$

The characteristic functions χ^F and χ^S are Heaviside functions which are nonzero in the fluid and solid domain, respectively. The Jacobian of the flow map, $\frac{\partial x}{\partial \xi}$, links derivatives with respect to the Lagrangian and the Eulerian coordinates.

The considered test case involves the following initial displacement and velocity fields:

$$\begin{aligned} d(\xi, 0) &= 0 && \text{for } \xi \in [0, 0.5] \\ u(x, 0) = \frac{\partial d(\xi, 0)}{\partial t} &= -0.1(\cos(2\pi x) + 1) && \text{for } x \in [-0.5, 0.5] \end{aligned} \tag{8.5}$$

The imposed boundary conditions require the materials to be at rest at the outer limits of the computational domain and are expressed by

$$u(-0.5, t) = d(0.5, t) = \frac{\partial d(0.5, t)}{\partial t} = 0 \text{ for } t \geq 0. \tag{8.6}$$

8.3 SPH solution of Cottet Model

8.3.1 Particle discretization of governing equations

According to Eq.(8.4) the particle position x_p, volume v_p and velocity u_p evolve by the following system of ordinary differential equations

$$\begin{aligned}\frac{dx_p}{dt} &= u_p, \\ \frac{dv_p}{dt} &= \left\langle \frac{\partial u}{\partial x} \right\rangle_p v_p, \\ \frac{du_p}{dt} &= -\frac{1}{2}\left\langle \frac{\partial u}{\partial x} \right\rangle_p u_p \chi^F \\ &+ \left(\chi^F + \left\langle \frac{\partial x}{\partial \xi} \right\rangle_p \chi^s\right)\left(\left\langle \frac{\partial}{\partial x}\mu\chi^F\frac{\partial u}{\partial x} \right\rangle_p + \left\langle \frac{\partial \xi}{\partial x} \right\rangle_p \left\langle \frac{\partial}{\partial \xi}E\chi^S\frac{\partial d}{\partial \xi} \right\rangle_p\right),\end{aligned} \tag{8.7}$$

where $\langle \diamond \rangle_p$ denotes the derivative approximation on a particle p (cf. Eq.(2.20)).

The Jacobian of the flow mapping is approximated by $\left\langle \frac{\partial x}{\partial \xi} \right\rangle_p = \frac{v_p}{h}$. We solve the anisotropic diffusion terms in Eq.(8.7) using the method of particle strength exchange (PSE) for anisotropic diffusion [192, 50]. Note, that the diffusion tensor in multiple dimension is reduced to a simple average in one dimension. Therefore,

$$\begin{aligned}
\left\langle \frac{\partial}{\partial x} \mu \chi^F \frac{\partial u}{\partial x} \right\rangle_p &= \sum_j (u_j - u_p) \frac{1}{2} (\mu_p \chi_p^F + \mu_j \chi_j^F) v_j \frac{\partial^2}{\partial x^2} W(x_p - x_j, h) \\
&= \frac{1}{2} \mu \sum_j (u_j - u_p) (\chi_p^F + \chi_j^F) v_j \frac{\partial^2}{\partial x^2} W(x_p - x_j, h) \qquad (8.8) \\
\left\langle \frac{\partial}{\partial \xi} E \chi^S \frac{\partial d}{\partial \xi} \right\rangle_p &= \sum_j (d_j - d_p) \frac{1}{2} (E_p \chi_p^S + E_j \chi_j^S) v_j \frac{\partial^2}{\partial \xi^2} W(\xi_p - \xi_j, h) \\
&= \frac{1}{2} E \sum_j (d_j - d_p) (\chi_p^S + \chi_j^S) v_j \frac{\partial^2}{\partial \xi^2} W(\xi_p - \xi_j, h) \qquad (8.9)
\end{aligned}$$

Using this approach, we do not apply the one-sided remeshing technique. Each particle is redistributed by the full space kernel M'4. A critical point of this approach is the fact that the displacement needs to be known of fluid particles located in the support of a solid particle for evaluating the anisotropic diffusion term according to Eq. (8.9). This requires the extrapolation of the solid displacement over the interface. As a first approach, we assign the displacement of the nearest solid particle to the interface fluid particle.

A different approach to solve Eq. (8.4) results from the idea to use first derivatives only. Therefore, the evolution of the particle velocity u is expressed by

$$\begin{aligned}
\frac{du_p}{dt} &= -\frac{1}{2} \left\langle \frac{\partial u}{\partial x} \right\rangle_p u_p \chi^F \qquad (8.10) \\
&+ \left(\chi^F + \left\langle \frac{\partial x}{\partial \xi} \right\rangle_p \chi^s \right) \left\langle \frac{\partial}{\partial x} \left(\mu \left\langle \frac{\partial u}{\partial x} \right\rangle_p \chi^F + E \left\langle \frac{\partial d}{\partial \xi} \right\rangle_p \chi^S \right) \right\rangle_p ,
\end{aligned}$$

where $\left\langle \frac{\partial u}{\partial x} \right\rangle_p \chi^F$ and $\left\langle \frac{\partial d}{\partial \xi} \right\rangle_p \chi^S$ are one-sided derivatives within the corresponding domain. The evaluation of the one-sided derivatives involves the normalization of the derivative of the SPH Kernel as described in [130]. Using one-sided formula we need to impose an additional interface condition. This condition requires the continuity of velocity at the interface, thus $u(\gamma(t), t) = \frac{\partial d(\gamma(t), t)}{\partial t}$. The interface velocity is approximated by the average velocity of the two particles surrounding the interface and assigned to these particles.

8.3.2 Boundary Conditions

In both approaches, the no-slip boundary condition at $x = -0.5$ is solved using ghost particles with an extrapolated velocity [162]. The solid particle at $x = 0.5$ is fixed to its initial position.

8.3.3 Numerical Results

The system parameters of the simulations are $\mu = 0.001$ for the viscosity and $E = 1$ for the elasticity. For time integration of the equations (8.7) and (8.11) we use a fourth order Runge-Kutta scheme with a timestep of 0.005. We compare the results against an ALE-method simulation based on centered finite-differences and a hybrid finite-difference particle method (PSE) solution of Cottet presented in [39]. The L_2-error of the velocity field is evaluated based on a well converged solution of 400 particles using rSPH with anisotropic diffusion. As a reference velocity we use the mean velocity of the initial condition ($\bar{u} = -0.1$).

rSPH with anisotropic diffusion

Fig. 8.1 compares the results of the rSPH approach using anisotropic diffusion (Eq. 8.7) with the ALE results. In both simulations, the domain is discretized by 100 computational elements. At early time ($t \leq 1$), the results are very similar. However, at $t = 2.0$, the ALE solution reveals a stability problem, in regions where the velocity gradients of the fluid are high, whereas the rSPH solution is observed to remain stable. Compared to a high resolution simulation, the rSPH method is more accurate than the ALE solution with L^2-errors of 0.04 and 0.14, respectively. The rSPH results with two different resolutions match very well with the particle solution of Cottet (Fig. 8.2). The hybrid finite difference-particle method has an L^2-error of 0.03. The lower error is mainly due to a different treatment of the interface. In the particle method of Cottet, the fluid particle at the interface is not redistributed, and its displacement is defined as its distance to its initial position. To compensate the resulting non-equidistant spacing of the particles at

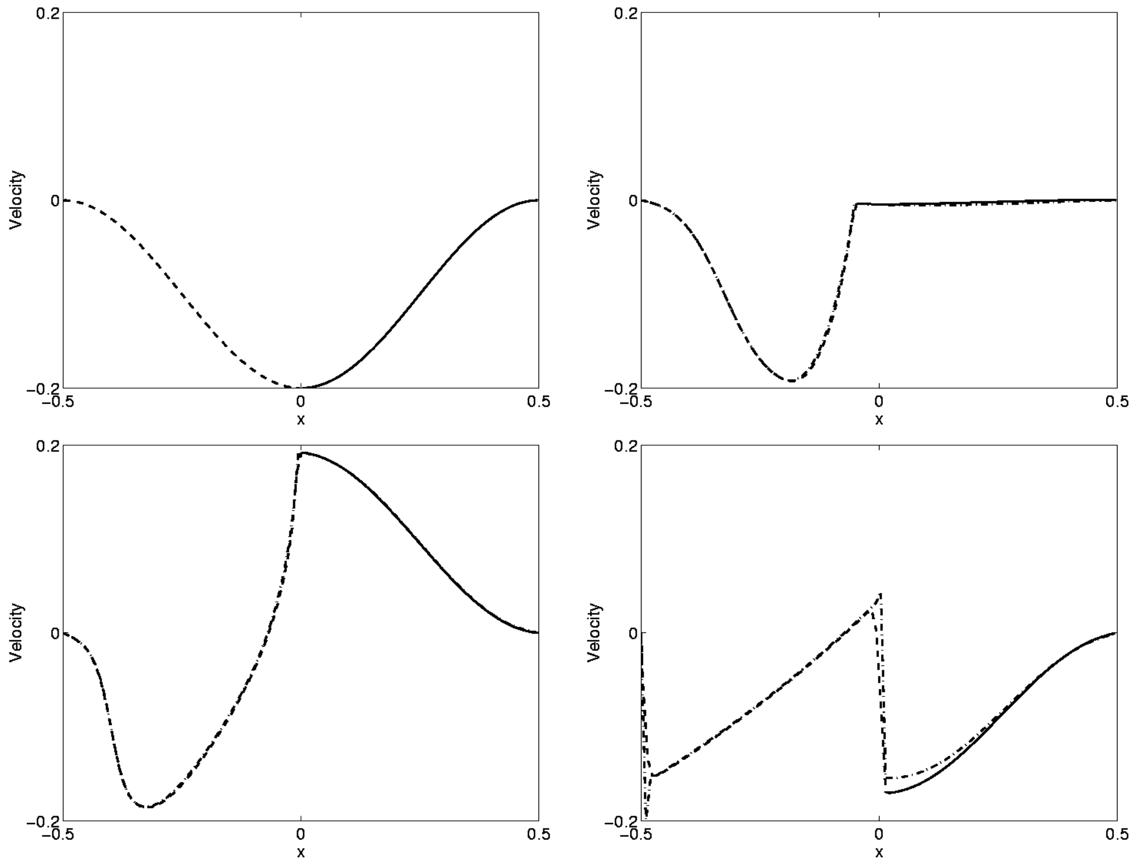


Figure 8.1: Velocity profiles at $t = 0.0$*, 0.5, 1.0 and 2.0 of the rSPH with anisotropic diffusion (fluid (- -), solid (–))and the ALE solution (*$\cdot -$*) (left to right, top to bottom)*

the interface, an asymmetric kernel is used which improves the accuracy of the method. We, however, do not apply a different kernel at the interface but remesh the interface fluid particle. We assign the interface displacement to the fluid particle which is approximated by the displacement of the solid particle closest to the interface.

rSPH with one-sided differentiation

The results of the rSPH with one-sided differentiation are compared to the ALE solution in Fig. 8.3. Again, the domain is divided into 100 computational elements. This particle approach is also more robust that the ALE method. The velocity profile is more accurate than the one of the ALE method but less accurate that the particle solution of Cottet (Fig. 8.4) with L^2-errors of 0.10, 0.14, and 0.03, respectively. Regarding the computa-

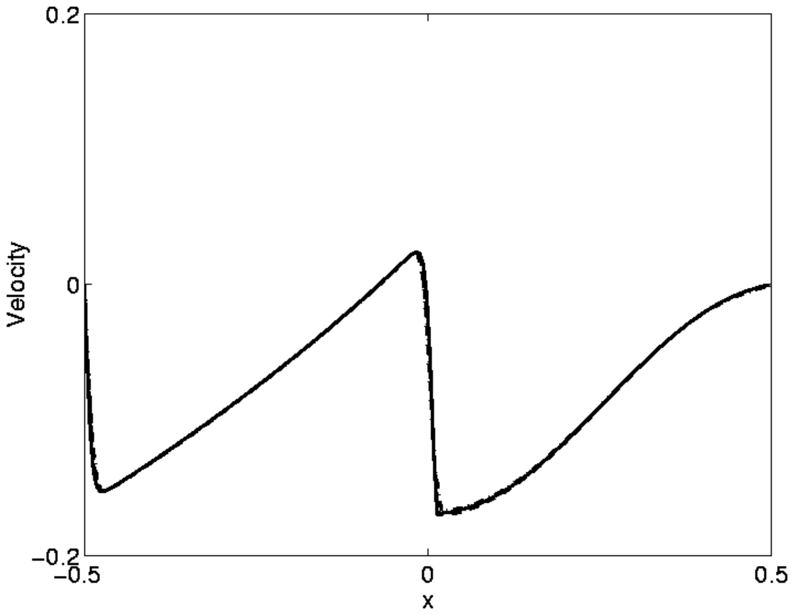


Figure 8.2: Velocity profiles at $t = 2.0$ of the rSPH with anisotropic diffusion (100 particle (- -), 200 particles (–)) and the particle solution (· -) of Cottet [39].

tional cost this approach is more expensive than rSPH with anisotropic diffusion because more differential operations need to be approximated. Moreover, the treatment of the interface is complicated by the one-sided differentiation, but does not require the extrapolation of the solid displacement to the fluid particle in the vicinity of the interface.

Convergence

Fig. 8.5 shows the convergence of both rSPH approaches measured by the L^2-error. It demonstrates that the rSPH method with one-sided differentiation has basically a linear convergence rate because of the second-order full-space and the first-order one-sided spatial differentiation. The plot indicates that the rSPH approach with anisotropic diffusion converges linearly. The small irregularities in the convergence is due to the approximation of the fluid displacement at the interface.

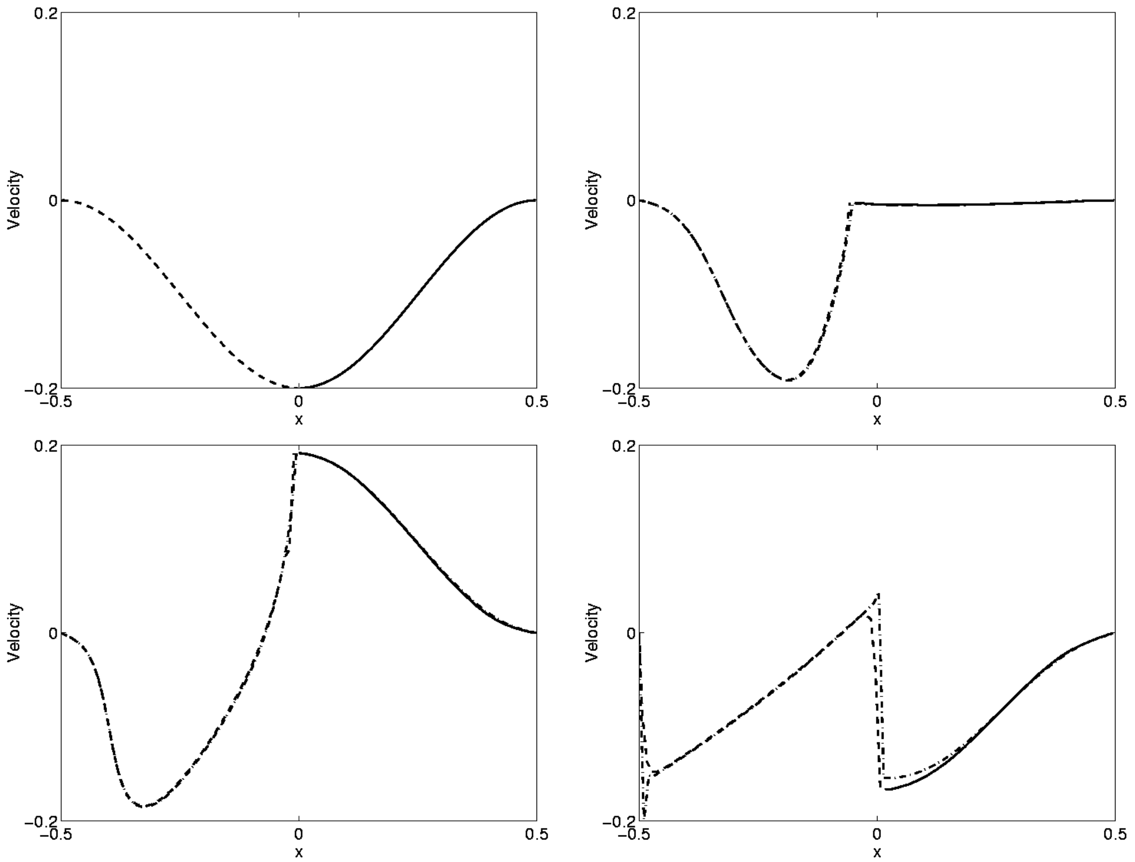


Figure 8.3: Velocity profiles at $t = 0.0$, 0.5, 1.0 and 2.0 of the rSPH with one-sided differentiation (fluid (- -), solid (–))and the ALE solution ($\cdot$ –)(left to right, top to bottom)

8.4 Model Extension for Compressible Fluid in 1D

8.4.1 Governing Equations

The expansion of the Burger's equation to a compressible Navier-Stokes equation involves the variability of the fluid density ρ and the consideration of pressure term in the momentum equation (Eq.(8.1)) leading to a one-dimensional Navier-Stokes equation.

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = -\frac{1}{\rho}\left(\frac{\partial p}{\partial x} + \frac{4}{3}\nu\frac{\partial^2 u}{\partial x^2}\right) \qquad \text{for } x \in [-0.5, \gamma(t)], \tag{8.11}$$

where p is the pressure. The factor $\frac{4}{3}$ is based on the Stokes' Postulate for compressible fluid [180].

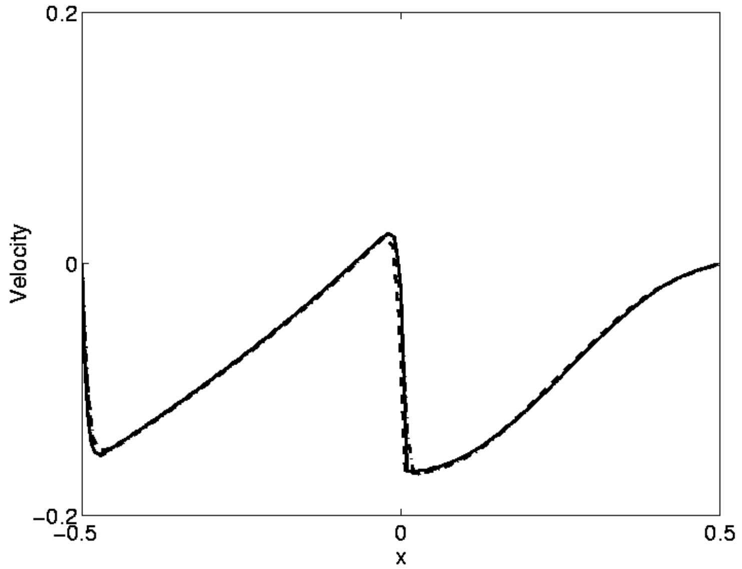


Figure 8.4: Velocity profiles at $t = 2.0$ of the rSPH with one-sided differentiation (100 particle (- -), 200 particles (–))and the particle solution (· -) of Cottet [39]

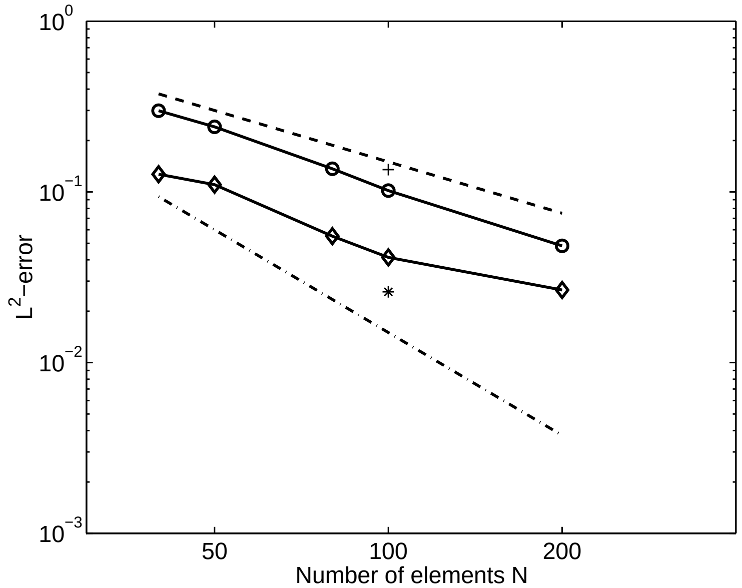


Figure 8.5: Error plot of the rSPH solutions (anisotropic diffusion (⋄-) and one-sided differentiation (○-)) at $t = 2.0$ compared to hybrid finite-difference particle (⋆) and ALE (+) solution of Cottet [39] (first order - - , second order · -)

The set of governing equations is closed by the equation of state for an ideal gas

$$p = \rho R T \tag{8.12}$$

where R is the specific gas constant and T the temperature. We assume the temperature $T = T_0$ to be constant in space and time.

The solid displacement remains to evolve according to the one-dimensional wave equation (Eq.(8.2)). The equilibrium of stress equation at the interface contains an additional term, that takes the fluid pressure into account.

$$-p(\rho(\gamma(t), t)) + \mu \frac{\partial u(\gamma(t), t)}{\partial x} = E \frac{\partial d(0, t)}{\partial \xi} \tag{8.13}$$

Thus, the solid needs initially to be pre-stressed (Fig. 8.6) for a non-zero pressure and $\frac{\partial u}{\partial x} = 0$.

We obtain a uniform formulation describing both, fluid and solid, by following the approach of Cottet [39]

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} = \frac{1}{\rho} \frac{\partial}{\partial x} \left[\left(\frac{4}{3} \mu \frac{\partial u}{\partial x} - p \right) \chi^F + E \frac{\partial d}{\partial \xi} \chi^S \right] \tag{8.14}$$

The boundary conditions remain as decribed in Eq.(8.6). The initial displacement of the interface is obtained by solving the ODE $E\frac{\partial d(\xi)}{\partial \xi} = -p$ with the boundary condition $d(0.5, 0) = 0$. Thus, the initial conditions differ from Eq.(8.5) and are expressed by

$$\begin{aligned} d(\xi, 0) &= -\frac{p}{E}\xi + \frac{1}{2}\frac{p}{E} && \text{for } \xi \in [0, 0.5] \\ u(x, 0) = \frac{\partial d(\xi, 0)}{\partial t} &= -0.1(\cos(2\pi x) + 1) && \text{for } x \in [-0.5, 0.5]. \end{aligned} \tag{8.15}$$

8.4.2 Particle discretization of governing equations

Particle position x_p density ρ_p and velocity u_p evolve by the following ordinary differential equations derived from equation (8.14)

$$\frac{dx_p}{dt} = u_p \tag{8.16}$$

$$\frac{d\rho_p}{dt} = -\rho_p \left\langle \frac{\partial u}{\partial x} \right\rangle_p \tag{8.17}$$

$$\frac{du_p}{dt} = F(x_p, x_q; \xi_p, \xi_q; u_p, u_q), \tag{8.18}$$

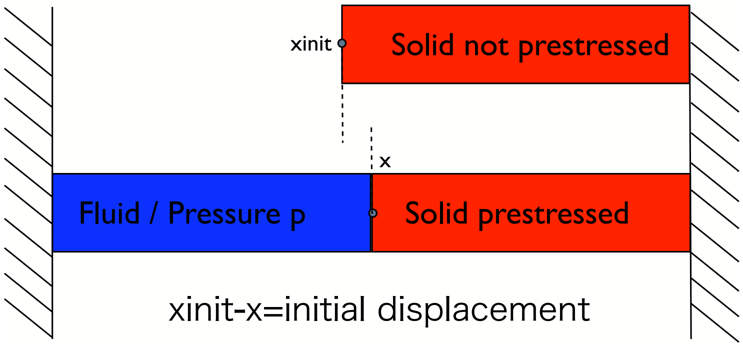


Figure 8.6: Pre-stressed solid to obtain equilibrium of stress at the interface

where $\langle\diamond\rangle_p$ denotes the derivative approximation on a particle p (cf. Eq. (2.20)). The function F includes the discretized right hand side of the momentum equation (Eq.8.14) that can be solved using one-sided differentiation

$$F = \frac{1}{\rho_p}\left\langle \frac{\partial}{\partial x}\left[\left(\frac{4}{3}\mu\left\langle\frac{\partial u}{\partial x}\right\rangle_p - p\right)\chi^F + E\left\langle\frac{\partial d}{\partial \xi}\right\rangle_p \chi^S\right]\right\rangle_p \tag{8.19}$$

where $\left\langle\frac{\partial u}{\partial x}\right\rangle_p \chi^F$ and $\left\langle\frac{\partial d}{\partial \xi}\right\rangle_p \chi^S$ are one-sided derivatives at the interface. Using one-sided derivatives the interface condition $u(\gamma(t),t) = \frac{\partial d(\gamma(t),t)}{\partial t}$ is mandatory to guarantee stability. The interface velocity and force are approximated by the averaging over the two particles adjacent to the interface and assigned to these particles. The advantage of the rSPH with one-sided differentiation is the ease of handling the interface.

Alternatively, we can use the approach of rSPH with anisotropic diffusion involving second derivatives

$$\begin{aligned} F &= \frac{1}{\rho_p}\left(\chi^F + \left\langle\frac{\partial x}{\partial \xi}\right\rangle_p \chi^S\right) \\ &\quad \left[\left(\left\langle\frac{\partial}{\partial x}\chi^F\frac{4}{3}\mu\frac{\partial u}{\partial x}\right\rangle_p - \left\langle\frac{\partial}{\partial x}\chi^F p\right\rangle_p\right) + \left\langle\frac{\partial \xi}{\partial x}\right\rangle_p \left\langle\frac{\partial}{\partial \xi}E\chi^S\frac{\partial d}{\partial \xi}\right\rangle_p\right]. \end{aligned} \tag{8.20}$$

The boundary conditions are identical with the test case described in Section 8.3.2.

8.4.3 Results

We show the numerical results of the test problem using one-sided differentiation. Regarding the anisotropic approach, the extrapolation of the displacement over the interface is crucial element that needs more investigation.

The viscosity in the simulation is set to $\mu = 0.01$ while the Young's modulus is defined to be $E = 1$. A set of 300 particles are initialized to describe the fluid-solid domain. A fourth order Runge-Kutta scheme with a time-step of 0.00033 is used for time integration of Eq.(8.18). The fluid particles are remeshed after every time-integration step.

Fig. 8.7-8.10 show snapshots of the test case simulation considering an initial velocity of $u = -0.1(\cos(2\pi x) + 1)$. Initially, the material moves towards the left boundary. Therefore, the density increases at the left wall and the fluid is finally repelled due to the resulting pressure at the wall. On the right side of the domain, the solid is first stretched and later compressed again as it overshoots its equilibrium. These physical phenomena leads to pressure waves traveling through the domain that are reflected by the walls. The strength of the waves is damped by the fluid viscosity causing energy to dissipate.

8.5 Model Extension for a Compressible Fluid in 2D

The extension to two dimension requires the consideration of shear stresses. We validate our results by comparing with the one-dimensional results, and test the shear stress behavior of a channel flow.

8.5.1 Fluid Governing Equations

The Navier Stokes equation for a compressible fluid [180] is expressed by

$$\rho \frac{d\mathbf{u}}{dt} = -\nabla p + \nabla(2\mu\mathbf{S} - \frac{2}{3}\mu\nabla\mathbf{u}\mathbf{I}), \tag{8.21}$$

where the strain-rate tensor in two dimensions is defined by

$$\nabla\mathbf{S} = \begin{bmatrix} \frac{\partial}{\partial x} & \frac{\partial}{\partial y} \end{bmatrix} \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{1}{2}\left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\right) \\ \frac{1}{2}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) & \frac{\partial v}{\partial y} \end{bmatrix}. \tag{8.22}$$

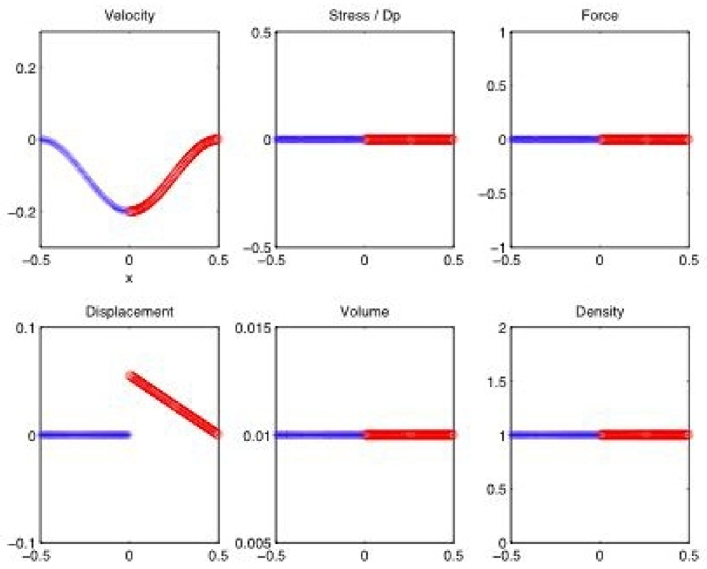


Figure 8.7: Fluid-solid interactions in 1D. Profile of velocity, stress, body force, displacement, volume and density at $t = 0$. Fluid on the left, solid on the right.

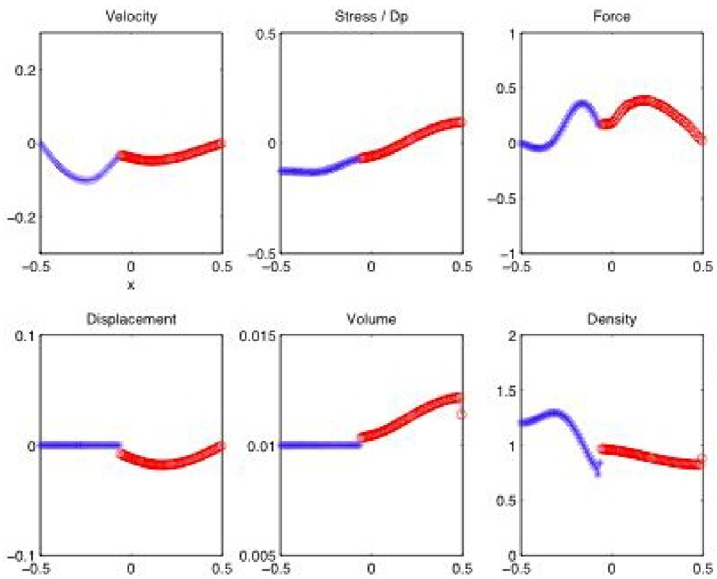


Figure 8.8: Fluid-solid interactions in 1D. Profile of velocity, stress, body force, displacement, volume and density at $t = 0.5$. Fluid on the left, solid on the right.

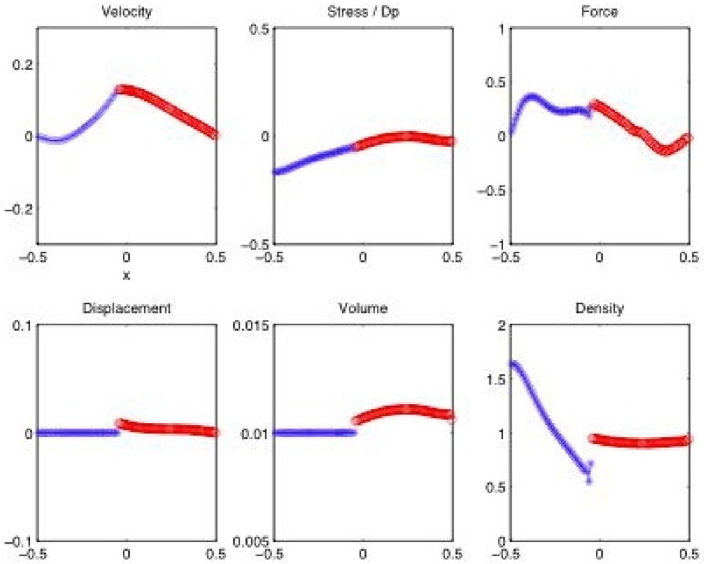


Figure 8.9: Fluid-solid interactions in 1D. Profile of velocity, stress, body force, displacement, volume and density at $t = 1$. Fluid on the left, solid on the right.

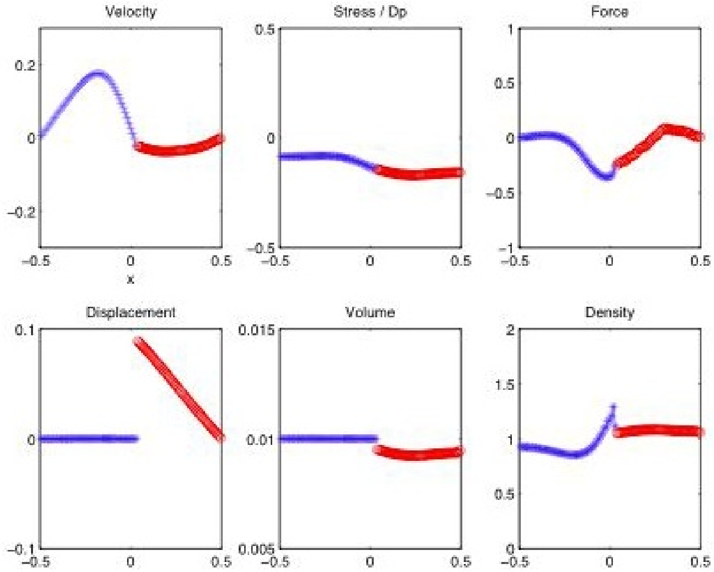


Figure 8.10: Fluid-solid interactions in 1D. Profile of velocity, stress, body force, displacement, volume and density at $t = 2$. Fluid on the left, solid on the right.

8.5.2 Solid Governing Equations

The solid is described by a linear elastic model in plane stress governed by the equations of continuum mechanics. The continuum equilibrium condition in two dimensions is defined by

$$\rho \frac{Du}{Dt} = \frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \tau_{xy}}{\partial y} + f_1 \tag{8.23}$$

$$\rho \frac{Dv}{Dt} = \frac{\partial \tau_{yx}}{\partial x} + \frac{\partial \sigma_{yy}}{\partial y} + f_2 \tag{8.24}$$

where σ_{ij}, τ_{ij} represent the components of the material stress tensor and f_1, f_2 external forces. The constitutive model describes the stress-strain relationship and is expressed in the case of linear elasticity by

$$\sigma_{ij} = 2\mu_s \varepsilon_{ij} + \lambda \delta_{ij} \varepsilon_{kk} \tag{8.25}$$

$$\mu_S = \frac{E}{2(1+\nu)} \tag{8.26}$$

$$\lambda = \frac{\nu E}{(1-2\nu)(1+\nu)}$$

where μ_s is the shear modulus, E is the Young's modulus, and ν is the Poisson ration. The governing equations are closed by the kinematic relationships

$$\begin{aligned} \varepsilon_{\xi\xi} &= \frac{\partial d_\xi}{\partial \xi} \\ \varepsilon_{\eta\eta} &= \frac{\partial d_\eta}{\partial \eta} \\ \varepsilon_{\xi\eta} &= \frac{1}{2}\left(\frac{\partial d_\xi}{\partial \eta} + \frac{\partial d_\eta}{\partial \xi}\right) \end{aligned} \tag{8.27}$$

where ξ and η are the components of the Eulerian variable.

8.5.3 Interface Equilibrium Condition

The fluid stress is expressed based on Eq.(8.21) in the Lagrangian variables (x, y) by

$$\sigma_{\mathbf{xx}} = -p + 2\mu \frac{\partial u}{\partial x} - \frac{2}{3}\mu \nabla \mathbf{u} = -p + \frac{2}{3}\mu \left(2\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y}\right) \tag{8.28}$$

$$\sigma_{\mathbf{yy}} = -p + \frac{2}{3}\mu \left(2\frac{\partial v}{\partial y} - \frac{\partial u}{\partial x}\right) \tag{8.29}$$

$$\tau_{\mathbf{xy}} = \mu \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right), \tag{8.30}$$

whereas the solid stress can be defined based on Eqs.(8.25)-(8.27) in Eulerian variables (ξ, η) by

$$\sigma_{\xi\xi} = 2\mu_S \frac{\partial d_\xi}{\partial \xi} + \lambda \left(\frac{\partial d_\xi}{\partial \xi} + \frac{\partial d_\eta}{\partial \eta} \right) \tag{8.31}$$

$$\sigma_{\eta\eta} = 2\mu_S \frac{\partial d_\eta}{\partial \xi} + \lambda \left(\frac{\partial d_\xi}{\partial \xi} + \frac{\partial d_\eta}{\partial \eta} \right) \tag{8.32}$$

$$\tau_{\xi\eta} = \mu_S \left(\frac{\partial d_\xi}{\partial \eta} + \frac{\partial d_\eta}{\partial \xi} \right) \tag{8.33}$$

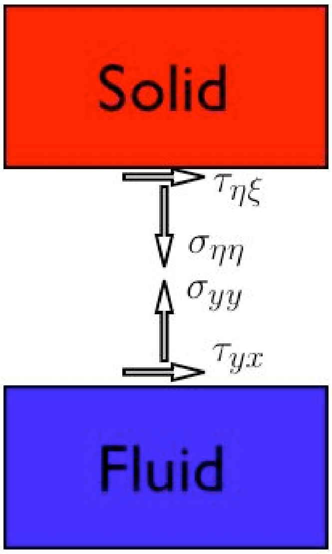


Figure 8.11: Interface stress components regarding a horizontal cut

Applying a horizontal cut at the interface (cf. Fig. 8.11) we obtain the following stress balance

$$\underbrace{-p + \frac{2}{3}\mu \left(2\frac{\partial v}{\partial y} - \frac{\partial u}{\partial x} \right)}_{\sigma_{yy}} = \underbrace{2\mu_S \frac{\partial d_\eta}{\partial \xi} + \lambda \left(\frac{\partial d_\xi}{\partial \xi} + \frac{\partial d_\eta}{\partial \eta} \right)}_{\sigma_{\eta\eta}} \tag{8.34}$$

$$\underbrace{\mu \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right)}_{\tau_{xy}} = \underbrace{\mu_S \left(\frac{\partial d_\xi}{\partial \eta} + \frac{\partial d_\eta}{\partial \xi} \right)}_{\tau_{\xi\eta}} \tag{8.35}$$

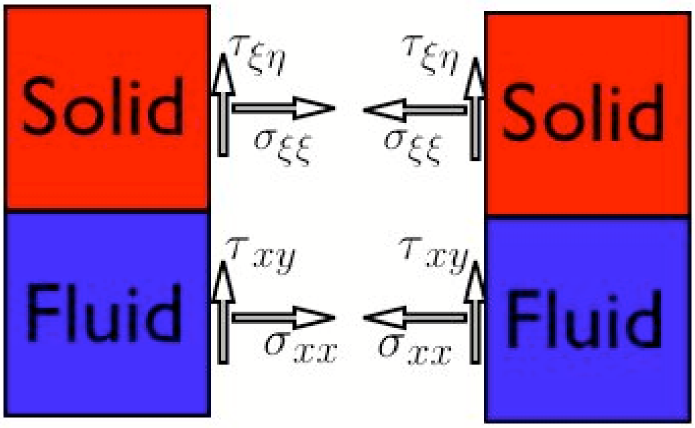


Figure 8.12: Interface stress components regarding a vertical cut

Cutting the interface vertically (cf. Fig. 8.12), leads to the following stress balance

$$\underbrace{-p + \frac{2}{3}\mu\left(2\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y}\right)}_{\sigma_{xx}} = \underbrace{2\mu_S\frac{\partial d_\xi}{\partial \xi} + \lambda\left(\frac{\partial d_\xi}{\partial \xi} + \frac{\partial d_\eta}{\partial \eta}\right) + const(y)}_{\sigma_{\xi\xi}} \tag{8.36}$$

$$\underbrace{\mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)}_{\tau_{yx}} = \underbrace{\mu_S\left(\frac{\partial d_\xi}{\partial \eta} + \frac{\partial d_\eta}{\partial \xi}\right)}_{\tau_{\eta\xi}} \tag{8.37}$$

The symmetry of the shear stress tensor τ requires $\tau_{xy} = \tau_{yx}, \;\; \tau_{\xi\eta} = \tau_{\eta\xi}$. Together with the stress balance, this leads to a continuity in the stress distribution of the form

$$\begin{aligned} &\sigma_{xx}\chi^F + \sigma_{\xi\xi}\chi^S && \text{smooth for } x \in [-0.5, 0.5] \\ &\sigma_{yy}\chi^F + \sigma_{\eta\eta}\chi^S && \text{smooth for } y \in [-0.5, 0.5] \\ &\tau_{xy}\chi^F + \tau_{\xi\eta}\chi^S && \text{smooth for } x \text{ and } y \in [-0.5, 0.5] \end{aligned} \tag{8.38}$$

Thus, we can apply Eqs.(8.23) and (8.24) over the fluid-solid interface resulting in

$$\rho\frac{Du}{Dt} = \frac{\partial}{\partial x}\left[\sigma_{xx}\chi^F + \sigma_{\xi\xi}\chi^S\right] + \frac{\partial}{\partial y}\left[\tau_{yx}\chi^F + \tau_{\eta\xi}\chi^S\right] \tag{8.39}$$

$$\rho\frac{Dv}{Dt} = \frac{\partial}{\partial y}\left[\sigma_{yy}\chi^F + \sigma_{\eta\eta}\chi^S\right] + \frac{\partial}{\partial x}\left[\tau_{xy}\chi^F + \tau_{\xi\eta}\chi^S\right] \tag{8.40}$$

8.5.4 Particle Equations

For simplification, we set the poisson ratio to zero ($\nu = 0$). The ODEs governing the evolution of particle attributes are based on Eqs. (8.39) and (8.40). Particle position $\mathbf{x}_p$ density ρ_p and velocity $\mathbf{u}_p$ evolve according to

$$\frac{d\mathbf{x}_p}{dt} = \mathbf{u}_p, \qquad \mathbf{x}_p = \begin{pmatrix} x_p \\ y_p \end{pmatrix} \tag{8.41}$$

$$\frac{d\rho_p}{dt} = -\rho\nabla\mathbf{u}_p, \qquad \mathbf{u}_p = \begin{pmatrix} u_p \\ v_p \end{pmatrix} \tag{8.42}$$

The evolution of the velocity x-component u_p is expressed by

$$\rho_p\frac{du_p}{dt} = \left\langle \frac{\partial}{\partial x}\left[\sigma_{xx}\chi^F + \sigma_{\xi\xi}\chi^S\right]\right\rangle_p + \left\langle \frac{\partial}{\partial y}\left[\tau_{yx}\chi^F + \tau_{\eta\xi}\chi^S\right]\right\rangle_p \tag{8.43}$$

where

$$\sigma_{xx} = -p + \frac{2}{3}\mu\left(2\left\langle\frac{\partial u}{\partial x}\right\rangle_p - \left\langle\frac{\partial v}{\partial y}\right\rangle_p\right) \tag{8.44}$$

$$\sigma_{\xi\xi} = E\left\langle\frac{\partial d_\xi}{\partial \xi}\right\rangle_p \tag{8.45}$$

$$\tau_{yx} = \mu\left(\left\langle\frac{\partial u}{\partial y}\right\rangle_p + \left\langle\frac{\partial v}{\partial x}\right\rangle_p\right) \tag{8.46}$$

$$\tau_{\eta\xi} = \frac{E}{2}\left(\left\langle\frac{\partial d_\xi}{\partial \eta}\right\rangle_p + \left\langle\frac{\partial d_\eta}{\partial \xi}\right\rangle_p\right) \tag{8.47}$$

In y-direction, we obtain for the y-component v_p

$$\rho_p\frac{dv_p}{dt} = \left\langle \frac{\partial}{\partial y}\left[\sigma_{yy}\chi^F + \sigma_{\eta\eta}\chi^S\right]\right\rangle_p + \left\langle \frac{\partial}{\partial x}\left[\tau_{xy}\chi^F + \tau_{\xi\eta}\chi^S\right]\right\rangle_p \tag{8.48}$$

where

$$\sigma_{yy} = -p + \frac{2}{3}\mu\left(2\left\langle\frac{\partial v}{\partial y}\right\rangle_p - \left\langle\frac{\partial u}{\partial x}\right\rangle_p\right) \tag{8.49}$$

$$\sigma_{\eta\eta} = E\left\langle\frac{\partial d_\eta}{\partial \eta}\right\rangle_p \tag{8.50}$$

$$\tau_{xy} = \mu\left(\left\langle\frac{\partial u}{\partial y}\right\rangle_p + \left\langle\frac{\partial v}{\partial x}\right\rangle_p\right) \tag{8.51}$$

$$\tau_{\xi\eta} = \frac{E}{2}\left(\left\langle\frac{\partial d_\xi}{\partial \eta}\right\rangle_p + \left\langle\frac{\partial d_\eta}{\partial \xi}\right\rangle_p\right) \tag{8.52}$$

8.5.5 Results

In two dimensions, we investigate two test cases, the first one being normal stress the second one shear stress driven. Both test cases have the same setup shown in Fig. 8.13. The shear and normal stresses in a fluid are related to the time-rate-of-change of the deformation of the fluid element. As a result, both shear and normal stresses depend on velocity gradients in the flow. In most viscous flows, normal stresses are much smaller than shear stresses, and they are often neglected. Normal stresses become important when the normal velocity gradients ($\frac{\partial u}{\partial x}$ and $\frac{\partial v}{\partial y}$) are very large, such inside a shock wave.

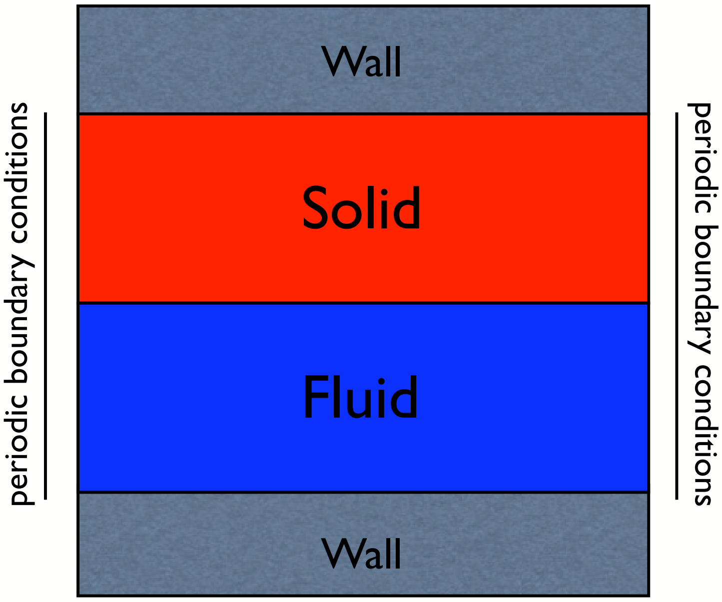


Figure 8.13: Setup of the two-dimensional test cases

Test Case for Normal Stresses

The first case is an extension of the one dimensional case (normal stress driven). The solid material resides initially in the positive y-domain and a fluid in the negative y-domain. While x-component of the initial velocity is set to zero $u(\mathbf{x}, 0) = 0$,the y-component $v(\mathbf{x}, 0) = -0.1(\cos 2\pi y + 1)$ is equivalent to the velocity profile in one-dimensional test case to compare the two cases. Similar to the one-dimensional case an analytical solution is not available.

We impose a no-slip condition at the boundary $y = -0.5$.Again, this boundary condition is obtained by using ghost particles with an extrapolated velocity. At the solid boundary $y = 0.5$ ghost particles are assign with an extrapolated displacement, to get a zero displacement at the solid-wall boundary.

The interface velocity and force are again approximated by the average of the interface adjacent particles and assigned to these.

The system parameters of the simulation are $\mu = 0.01$ for the viscosity of the fluid, the E-module is set to one as in the one-dimensional simulation. For time-integration a fourth order Runge-Kutta scheme with a timestep of 0.005 was used.

Since this test case is the extended version of the one-dimensional test case, the velocity profile are supposed to match at all times. Fig. 8.14 shows the final velocity profiles at $t = 2$ for both one and two dimensions. Even though the one dimensional simulation has a much higher resolution with 300 particles used and a time-step 15 times smaller than the two dimensional simulation, the agreement is good in the final velocity profile. Fig. 8.15 shows the velocity profile of the entire domain.

We observe that simulation results are sensitive to the remeshing procedure. The remeshing scheme can generate particles very close to the boundary that can cause numerical problems, because the particle velocity is extrapolated to the ghost particles over the boundary to maintain the no slip condition.

8.5.6 Test Case for Shear Stresses

The second test case involves the consideration of a channel flow, as shown in Fig. 8.16. The fluid will perform a classical motion described as channel flow, the plane Poiseuille flow. The Poiseuille flow is a two-dimensional flow between parallel plates, with a constant pressure gradient or in this case a body force f_x. At the interface the passing fluid will exert shear stress on the solid resulting in a deformation of the solid. The extent of

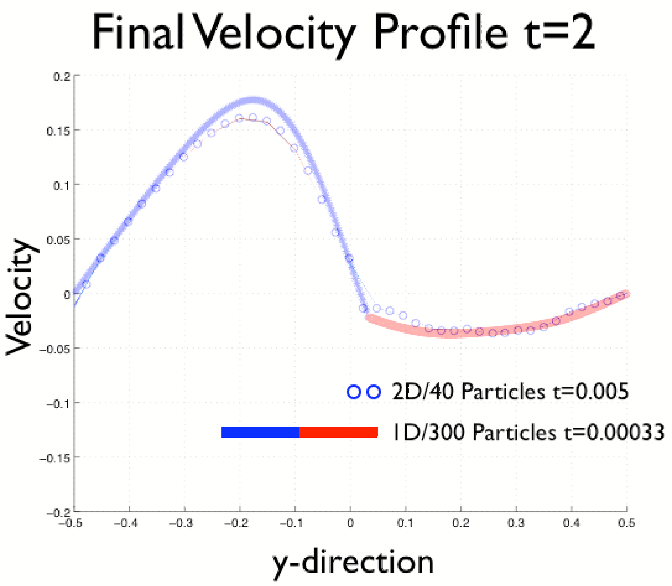


Figure 8.14: Comparison of final velocity profile in one and two-dimensions

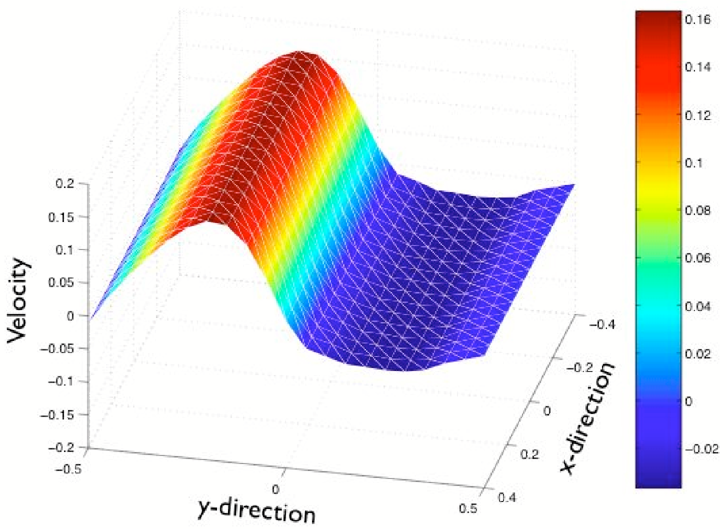


Figure 8.15: Final velocity trofile at $t = 2$ plotted over the x-y-domain

the deformation depends on the shear modulus μ_s of the solid.

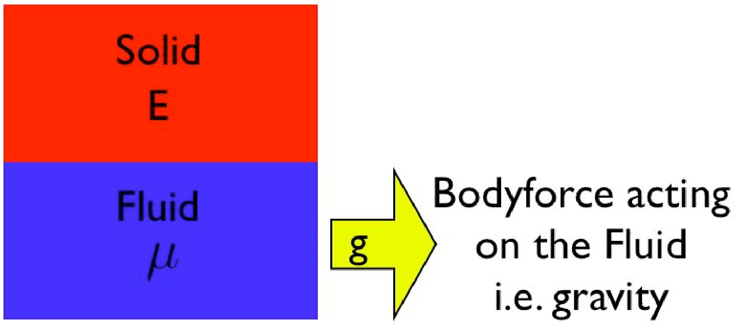


Figure 8.16: Setup of the second test case involving a channel flow

We consider, again, a no-slip condition at the fluid-wall interface ($y = -0.5$), and impose a zero-displacement condition at solid-wall interface ($y = 0.5$) using ghost particles.

The viscosity of the fluid is $\mu = 0.01$ and the E-module varies for each simulation. For time-integration a fourth order Runge-Kutta scheme with a timestep of 0.005 is used.

This simulation does not require remeshing because the particles overlap well during the simulation time. Fig. 8.17 shows the final positions of the particles after one time-period for different coefficients of elasticity (E-module). The behavior of the solid applied to shear stresses is dominated by the choice of the E-module because the shear module in these simulations is defined by $\mu_s = \frac{E}{2}$ (cf. Eq. 8.26) because we set the Poisson-ratio ν to zero. With increasing the value of the Young's modulus E the solid behaves stiffer, therefore, deformations are smaller as shown in Fig. 8.17. The displacement at the interface is shown for various Young's moduli in Fig. 8.18. The relation between the Young's modulus and the displacement resembles a hyperbola, which is excepted because doubling the Young's module is expected to halve the displacement. Plotting the relationship between Young's module and displacement double logarithmically, the data points reside close to a straight line as expected.

We know from solving the Navier-Stokes equation for a channel flow analytically that the velocity profile has the form of a parabolic function. The steady state behavior of the

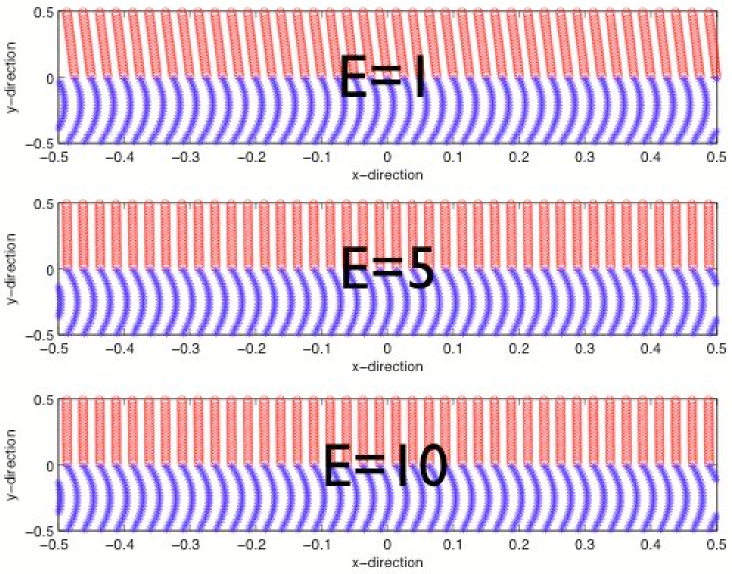


Figure 8.17: Particle positions at $t = 2$ for different Young's moduli E (not remeshed)

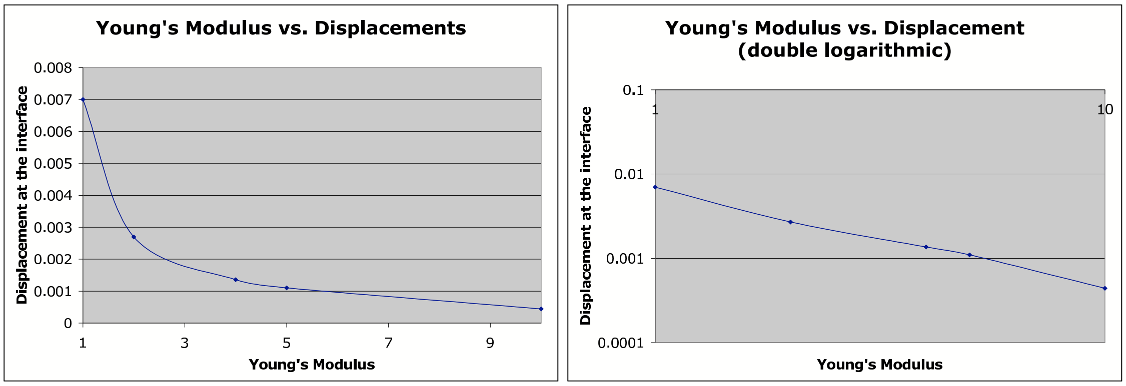


Figure 8.18: Effect of the Young's modulus E on the displacement of the interface

fluid is described by

$$\mu \frac{\partial^2 u}{\partial y^2} + \rho g = 0 \tag{8.53}$$

where g is gravitational force acting in x-direction. Integrating Eq.(8.53) twice leads to

$$u(y) = -\frac{\rho}{\mu} g \frac{y^2}{2} + Ay + B \tag{8.54}$$

The integration constants (A, B) can be derived using the two no-slip conditions $u(y = 0) = 0$ and $u(y = -0.5) = 0$. Thus, the analytical velocity profile is expressed by

$$u(y) = -\frac{\rho}{\mu} g \frac{y^2}{2} - \frac{1}{4}\frac{\rho}{\mu} gy \tag{8.55}$$

The maximum velocity is found at the position $y = -0.25$. Considering $\rho = 1, \mu = 0.01$ and $g = 0.03$ the maximum velocity in the channel flow becomes $u_{max} = 0.09375$. The velocity in the corresponding fluid simulation is $u(y = -.25) = 0.0953$ at $t = 15$. The simulation shows a discrepancy of less than 2% from the analytical solution. The final velocity profile is shown in Fig. 8.19.

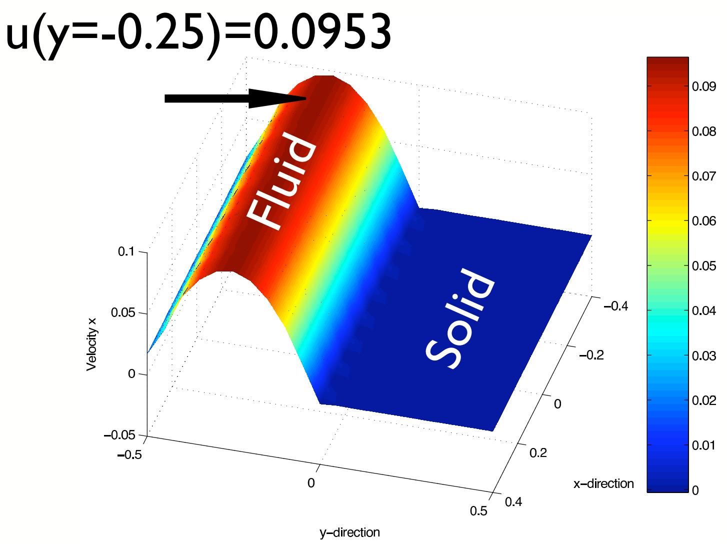


Figure 8.19: Velocity profile for channel flow at steady state

We can estimate the displacement of the interface at the steady state conditions by using the force balance at the interface

$$\tau_{xy} = \mu \frac{\partial u}{\partial y} = \frac{E}{2} \frac{\partial d_\xi}{\partial \eta} \tag{8.56}$$

Thus,

$$\frac{\partial d_\xi}{\partial \eta} = \frac{2}{E} \mu \frac{\partial u}{\partial y} = \frac{2}{E} \left(-\frac{1}{4} \rho g \right). \tag{8.57}$$

At steady state $\frac{\partial d_\xi}{\partial \eta}$ is constant for all η within the solid domain. Suppose that the distance of the interface to the wall is $\Delta\eta = 0.5$, the displacement of the interface of the stationary solution can be approximated by

$$d_{\xi,interface} = \frac{\partial d_\xi}{\partial \eta} \Delta\eta \tag{8.58}$$

$$= -\frac{\rho g \Delta\eta}{2E} \tag{8.59}$$

$$= 0.0018 \qquad \text{for} \quad E = 4 \tag{8.60}$$

The simulation results in a displacement of $d = 0.0014$ which is in good agreement with the estimated displacement.

Chapter 9

Conclusions

9.1 Introduction

The simulation of soft biological tissue is a crucial component of surgical simulator systems. The complexity of biomechanical systems leads to several numerical challenges, such as the handling of complex geometries and interactions with medical devices and body fluids.

This thesis has addressed a number of issues pertaining to the particle simulation of complex multi-physical systems for the use in virtual surgery. In particular, we have developed several numerical techniques for particle methods to improve their feasibility for virtual surgery environments. We presented a novel Lagrangian Particle Level Set method for accurate surface capturing. We considered a novel particle Immersed Boundary Method to handle complex geometries with fluid environments and demonstrated its use in fluid-structure interactions for anguilliform swimming. Moreover, we employed for the first time the remeshing technique in particle simulations of elastic solids. This also involved the development of a novel Lagrangian particle model for nonlinear elastic solids. In general, we can conclude that particles are viable tools for the use in virtual surgery. The main contributions and conclusions of this thesis are summarized below.

9.2 Particle Level Set Method

A reliable surface description is of central importance for the simulation of tissue in surgical simulators. To capture surfaces and interfaces we developed a novel Lagrangian particle level set method. The method is adaptive as particles adapt to resolve the evolution of the level sets and a consistent remeshing procedure is employed in order to ensure the convergence of the method when the particles get distorted by the flow map. Remeshing also enables the use of fast marching methods for the reinitialization of the level sets. The Lagrangian particle method provides a bridging description of interface tracking and interface capturing methodologies as it is adaptive and at the same time involves an implicit description of the interface. In Chapter 3, the efficiency and accuracy of the method, as well as a comparison with related methodologies, is demonstrated in a number of two and three dimensional benchmark problems. The method is shown to be well capable in curvature induced motion and it is capable of capturing changes in surface profiles, such as they may occur in the simulation of the etching and deposition processes. The simplicity of the method in reconstructing interfaces by a linear particle superposition makes it suitable for rapid tracking of large deformations and cutting of interfaces in virtual surgery environments where fast simulations of cutting of biological tissue is necessary.

9.3 Simulation of Elastic Solids

A reliable simulation of a tissue modeled as an elastic solid is essential in virtual surgery. In Chapter 5, we presented a particle solver for elastic solids that can handle linear and nonlinear constitutive models. We simulated, for the first time, an elastic solid described by a nonlinear hyperelastic model in a Lagrangian framework. The model development required a new set of governing equations that contained the evolution of the deformation gradient. The remeshing scheme resolves the numerical problem of tensile instability and assures the convergence of the method. We showed that the particle formulation has a similar accuracy as the finite element method. In the plane strain compression test, the finite element solution reveals severe numerical artifacts in regions where the material is forced to rotate 90 degrees. In contrast, the particle solution remains physically plausible. The particle solver can handle the material rotation in a more reliable way due to its Lagrangian methodology for both linear and nonlinear elasticity problems. We showed that the particle solver is suitable to simulate soft biological tissue exposed to an aspiration test. A comparison with the experimental measurement showed that particle solution captures the displacement of the tissue very well.

In Chapter 6, parallel scaling and efficiency were assessed in several test cases. All applications showed parallel efficiencies reaching or exceeding the present state of the art, and favorable run-times on large systems. We presented a state of the art PSE simulation using 1 billion particles, a VM simulation using 268 million particles – to our knowledge the largest VM done so far –, an SPH simulation exceeding the parallel efficiency of the currently fastest domain-specific code, simulations sustaining up to 33% of the machine peak performance, and a multigrid Poisson routine solving for half a billion unknowns in less than 7 seconds on 64 processors. Moreover, vectorization as tested on the NEC SX-5 computer demonstrated the suitability of the PPM library for vector architectures.

9.4 Particle Immersed Boundary Method

In Chapter 7, we considered a novel particle Immersed Boundary method for solving the no-slip boundary condition on complex boundaries in two and three dimensions. This method offers accurate particle simulations of flow-structure interaction. At a Mach Number of 0.1 the simulations agree well with the corresponding incompressible solutions. The accuracy of the method, as well as comparison with related methodologies, is demonstrated in a number of two and three dimensional benchmark problems involving flow past a cylinder and sphere. The method is shown to be well capable in solving fluid-structure interaction. The simplicity of the method in handling complex boundaries makes it suitable for simulating complex movement of flexible structures as they appear, for example, in anguilliform swimming. A drawback of this method is associated with the compressibility of the flow. Fast motions of the boundary in low viscous flow cause pressure waves in the fluid that can lead to numerical problems. Due to pressure waves the time step is restricted by the inverse of the speed of sound.

9.5 Fluid-Solid Interactions

The unified formulation of the fluid-solid interactions proposed in [39] provides an elegant description of the fluid-structure interaction since it avoids the need of explicitly tracking the interface. In Chapter 8, we presented two rSPH solutions to this problem, one with anisotropic diffusion, and one with one-sided differentiation. The simulations have shown that rSPH solutions are more robust and more accurate than the ALE method. Due to a different treatment of the interface, the hybrid finite-difference PSE solution of Cottet [39] provides more accurate results than the rSPH approaches. Comparing the two rSPH approaches, the rSPH with anisotropic diffusion is computationally less expensive, and is more accurate. However it relies on a displacement of a fluid particle whose value needs to be extrapolated. In this respect, it is more natural to use the one-sided differentiation approach.

We extended this model to consider compressible fluids in one and two dimensions. This model required a continuity equation, an equation of state and a pressure term in the momentum equation.

We employed the one-sided differentiation approach because the evaluation of the displacement in the fluid and the Jacobian matrix become obsolete. However, special attention is required for the preservation of the continuity in the velocity profile.

The accuracy of the two-dimensional implementation is demonstrated in two test cases. The first test is the two-dimensional extension of the one-dimensional case, and tests the impact of the normal stresses. The cross-section of the two-dimensional velocity distribution matches well with the one-dimensional result. The second test considers a channel flow where the material behavior is mainly influenced by shear stresses. The result compares well with the analytical flow profile in steady state. As expected, the steady state displacement of the interface is proportional to the reciprocal value of the solid stiffness, namely the Young's modulus.

9.6 Parallel Particle-Mesh Library

The Parallel Particle-Mesh library (PPM) provides a general-purpose, physics-independent infrastructure for simulating systems using particle methods. The library integrates particle, mesh, and hybrid particle-mesh algorithms and its design goals include ease of use, flexibility, state-of-the-art parallel scaling, good vectorization, and platform independence. The library provides a computational tool for large scale simulations using particle methods.

Flexibility and independence from specific physics was demonstrated by having various simulation client applications from fluid mechanics (SPH, VM) to diffusion problems (PSE).

Chapter 10

Outlook and Future Work

This chapter outlines the possible extensions of the present work as well as potential application in future research.

10.1 Particle Methods

Particle methods will offer a robust and flexible tool for simulating material behavior. Its accuracy, however, can be improved, for example, by the use of higher order kernels [54]. Resolving the complex structure of these kernel requires a significantly large number of particles in the kernel support resulting in a higher computational cost per particle. Another aspect is the treatment of boundary conditions. In general, further investigation is needed to achieve higher accuracy at boundaries. The commonly used normalization of the superposition and the use of image or ghost particle leads to first-order accuracy only.

10.2 Particle Simulation of Fluids

In terms of computational cost, it is promising to employ mesh-based derivative approximation, such as in Particle-Mesh Hydrodynamics [31]. The particles carrying the physical quantities move with the material velocity and are redistributed on mesh positions every time step. Mesh-based Finite-difference approximations determine the further particle evolution governed by the particle equations.

Moreover, particle multi-scaling techniques [18, 97] in space and time may improve the computational performance of the particle simulations. A local refinement in the region of interest or a numerical singularity will increase the accuracy and reliability of the simulation significantly. Adaptive time integration schemes adjust the time step according to the current situation and may lead to a large speed up with respect to computational time.

10.3 Particle Simulation of Elastic Solids

The presented particle solver for elastic solids will offer an promising alternative to the finite element solver, an established solver for continuum mechanics problems. However, the particle solver should be tested on more problems concerning stability and flexibility. Towards the use in virtual surgery, possible extensions include the development of particle model for further nonlinear elasticity models, collision detection and collision handling allowing for virtual cutting [21]. In particular, the issue of modeling viscosity effects needs further investigation.

Soft biological tissue can often be approximated by an incompressible material. Therefore, it may be possible to improve the computational efficiency of the solver for the use in virtual surgery, similar to the fluid simulations. Some of the techniques for achieving a zero divergence velocity field may also be applicable to solid simulation, for example the projection method. Exploiting the incompressibility of soft biological tissue may lead to stable simulations with larger time steps.

A further improvement in the terms of computational efficiency may be achieved by the Particle-Mesh Hydrodynamics approach [31], as mentioned in Section 10.1.

Particle simulations are particularly useful in systems where the solid undergoes large deformations or has a complex geometric structure. Therefore, particle solvers are promising tools in the simulation of challenging engineering problems. Potential applications of the particle solid solver range from the investigation of epithelial lumen formation in cells [189] over the development of gaskets and tires [184] to investigation of fracture behavior in concrete [129].

10.4 Particle Immersed Boundary Method

The particle immersed boundary method enables the realization of the no-slip boundary condition for complex boundaries in particle fluid simulation. This method can be exploited, for example, in biomedical simulation of the blood circulation in the human body. The simulation of vascular valves in blood flow could help in the development of cardiac artificial valves to prevent a vascular stenosis [7].

In problems involving water or blood flow, the compressibility of the fluid can be neglected. Therefore, it may be computational more efficient to consider the Navier-Stokes equation of an incompressible fluid. As the speed of sound limits the time-step in simulations of compressible fluids, the consideration of an incompressible fluid would allow for stable simulations using larger time steps. However, the simulation of incompressible fluid requires a numerical scheme to ensure the zero divergence of the velocity field. This can be accomplished by either solving the Poisson equation for the pressure directly [71] or by a projection method [47]. Koshizuka *et al.* [94, 95, 186], for example, keeps the particle density constant by introducing a corrective pressure term obtained by solving a Poisson equation. Recently, Colin [38] presented a particle solution of the Helmholtz-Hodge decomposition to solve for a zero divergence velocity field.

The present methodology can be extended to consider viscoelastic fluid [55]. Viscoelastic models are important in blood simulations because they take the non-Newtonian character of blood into account.

The particle immersed boundary method cannot only applied to fluids but can be extend to other material, such as elastic solid. The enforcement of a no-slip boundary condition would lead a fixed boundary of the solid.

10.5 Fluid-Solid Interaction

The present implementation of fluid-solid interaction is useful for two-dimensional problems. It can be extended to three dimensions by considering the three-dimensional stress balance at the interface. A Drawback of the model is the consideration of two different coordinate systems, in a Lagrangian and an Eulerian frame. This is inconvenient not only for the derivation of the model but also for the implementation, in particular in higher dimensions. As a result, the remeshing scheme, for example, is only applied to the fluid particles. A limitation to one coordinate system would, therefore, be desirable, particularly in respect to a parallel implementation. A solution to this problem can be the replacement of the Eulerian solid description by a Lagrangian description as presented in Section 5.

For surgical simulations, the behavior of soft biological requires the consideration of nonlinear elasticity and viscoelasticity effect. Therefore, it is essential to embed nonlinear elasticity and viscoelasticity models into the implementation. The particle solver can then be applied for simulations in the area of angioplasty [60]. Angioplasty is a medical procedure in which a balloon is used to open narrowed or blocked blood vessels of the heart (coronary arteries). After a heart attack it is often necessary to place a stent into the affected blood vessel. Particle simulations of fluid-solid interactions could help to optimize the shape of such stents.

A further example of flow-tissue interactions is the behavior of endothelial cells at the vascular wall. They are known to function in many physiological processes, e.g. blood pressure control. The endothelial tissue lines the interior surface of blood vessels and reacts very sensitively to the ambient blood flow. It is known that the endothelial cells change their shape according to the shear stresses acting in the flow.

10.6 Parallel Particle-Mesh Library

The PPM library will be a useful computational tool for challenging problems in large-scale particle-mesh simulations.

The next major version will include support for multi-level structures with selectively allocated memory patches that can be arbitrarily placed in the computational domain. This will enable the implementation of multi-domain and multi-level particle schemes in the spirit of mesh-oriented projects such as CHOMBO [9]. Solvers based on Adaptive Mesh

Refinement [18] can be added to the library architecture. Integration of modern software engineering concepts may improve the maintainability and flexibility of the code to ease the use of the library.

10.7 Lagrangian Particle Level Set Method

The Lagrangian Particle Level Set method to capture interfaces and surfaces can be employed virtually in every systems where interfaces are evolved.

Besides for simulations in virtual surgery and microchip fabrication, as shown in Section 3, level set method can be used, for example, in computer graphics, in computer vision and engineering problem [147]. In computer graphics, the level set method can be exploited for smoothing contours and noise removal in images. In computer vision, mesh-based level set methods are popular for shape detection in images. Therefore, the particle level set method may be useful for the three-dimensional segmentation of human organ for medical purposes. Within this scope, level set methods enjoy a good reputation for their stability and clarity of the resulting structures. Further possible engineering application include the interface capturing in combustion, crystal growth and dendritic solidification.

Since the signed distance function of the level set approach provides the signed distance information to the surface, the level set method may be extendable to detect collision of objects and may be involved in the collision handling. However, the aspect of efficiency needs further investigation within this scope.

Appendix A

A Virtual Cutting Approach Using a Simplified Solid Model

An important issue in virtual surgery simulation is the treatment of collisions with the biological tissue. While other researchers focus on the modelling of tissue interaction with a scalpel using tetrahedral meshes [21], this chapter presents the possibilities of using Smoothed Particle Hydrodynamics (SPH) based on a simpified solid model. The underlying computational structure, the set of particles, is not affected by the cut, only the particle-particle interactions resulting in a conservation of mass.

A.1 Governing Equations and Particle Discretization

As in [82], the mechanical behavior of soft biological tissue is modeled as a linear viscoelastic material for small strains. The momentum equation is expressed by

$$\rho \frac{D\mathbf{u}}{Dt} = \nabla \cdot \sigma + \mathbf{f_{ext}}, \tag{A.1}$$

where ρ is the density and $\mathbf{u}$ is the velocity. For simplicity, we approximate the density by $\rho = 1$ and the divergence is taken with respect to the initial positions. $\frac{D}{Dt}$ denotes the material derivative. $\mathbf{f_{ext}}$ is an external body force, here a gravitational force and σ is the Cauchy stress tensor that depends on the constitutive model of the considered material. The solid model is based on the generalized Hooke's law and is extended by the Kelvin–Voigt damping model [85]. Thus, the components σ_{ij} of the stress tensor σ depend linearly on the components ϵ_{ij} of the Cauchy Green strain tensor ϵ and the strain rate $\dot{\epsilon}_{ij}$,

$$\sigma_{ij} = 2\mu(\epsilon_{ij} + T\dot{\epsilon}_{ij}) + \lambda\delta_{ij}(\epsilon_{kk} + T\dot{\epsilon}_{kk}) \tag{A.2}$$

The indices $i, j, k = 1, 2, 3$ follow the Einstein's summation convention and δ_{ij} is the Kronecker delta symbol. The time constant T is the relaxation time of the damping, and μ and λ are the Lame constants,

$$\begin{aligned} \mu &= \frac{E}{2(1+\nu)} \\ \lambda &= \frac{\nu E}{(1-2\nu)(1+\nu)} \end{aligned} \tag{A.3}$$

where E represents the Young's modulus and ν the Poisson ratio.

The components ϵ_{ij} of the strain tensor are proportional to the spatial derivative of the displacement $\mathbf{d}$,

$$\epsilon_{ij} = \frac{1}{2}\left(\frac{\partial d_i}{\partial \xi_j} + \frac{\partial d_j}{\partial \xi_i}\right), \tag{A.4}$$

where ξ is the reference position, that is in this case equivalent to the initial position. Combining Eqs. (A.1), (A.2) and (A.4)results in Eqs. (A.5)–(A.7) below.

$$
\begin{aligned}
\frac{Du_1}{Dt} &= (2\mu+\lambda)\frac{\partial^2 d_1}{\partial \xi_1^2} + \lambda\frac{\partial^2 d_2}{\partial \xi_1 \partial \xi_2} + \lambda\frac{\partial^2 d_3}{\partial \xi_1 \partial \xi_3} \\
&+ \mu(\frac{\partial^2 d_1}{\partial \xi_2^2} + \frac{\partial^2 d_2}{\partial \xi_1 \partial \xi_2}) + \mu(\frac{\partial^2 d_3}{\partial \xi_3 \partial \xi_1} + \frac{\partial^2 d_1}{\partial \xi_3^2}) \\
&+ T\Big((2\mu+\lambda)\frac{\partial^2 u_1}{\partial \xi_1^2} + \lambda\frac{\partial^2 u_2}{\partial \xi_1 \partial \xi_2} + \lambda\frac{\partial^2 u_3}{\partial \xi_1 \partial \xi_3} \\
&+ \mu(\frac{\partial^2 u_1}{\partial \xi_2^2} + \frac{\partial^2 u_2}{\partial \xi_1 \partial \xi_2}) + \mu(\frac{\partial^2 u_3}{\partial \xi_3 \partial \xi_1} + \frac{\partial^2 u_1}{\partial \xi_3^2})\Big)
\end{aligned}
\tag{A.5}
$$

$$
\begin{aligned}
\frac{Du_2}{Dt} &= \lambda\frac{\partial^2 d_1}{\partial \xi_1 \partial \xi_2} + (2\mu+\lambda)\frac{\partial^2 d_2}{\partial \xi_2^2} + \lambda\frac{\partial^2 d_3}{\partial \xi_2 \partial \xi_3} \\
&+ \mu(\frac{\partial^2 d_1}{\partial \xi_2 \partial \xi_1} + \frac{\partial^2 d_2}{\partial \xi_1^2}) + \mu(\frac{\partial^2 d_2}{\partial \xi_3^2} + \frac{\partial^2 d_3}{\partial \xi_2 \partial \xi_3}) \\
&+ T\Big(\lambda\frac{\partial^2 u_1}{\partial \xi_1 \partial \xi_2} + (2\mu+\lambda)\frac{\partial^2 u_2}{\partial \xi_2^2} + \lambda\frac{\partial^2 u_3}{\partial \xi_2 \partial \xi_3} \\
&+ \mu(\frac{\partial^2 u_1}{\partial \xi_2 \partial \xi_1} + \frac{\partial^2 u_2}{\partial \xi_1^2}) + \mu(\frac{\partial^2 u_2}{\partial \xi_3^2} + \frac{\partial^2 u_3}{\partial \xi_2 \partial \xi_3})\Big)
\end{aligned}
\tag{A.6}
$$

$$
\begin{aligned}
\frac{Du_3}{Dt} &= \lambda\frac{\partial^2 d_1}{\partial \xi_1 \partial \xi_3} + \lambda\frac{\partial^2 d_2}{\partial \xi_2 \partial \xi_3} + (2\mu+\lambda)\frac{\partial^2 d_3}{\partial \xi_3^2} \\
&+ \mu(\frac{\partial^2 d_1}{\partial \xi_3 \partial \xi_1} + \frac{\partial^2 d_3}{\partial \xi_1^2} + \mu(\frac{\partial^2 d_2}{\partial \xi_3 \partial \xi_2}) + \frac{\partial^2 d_3}{\partial \xi_2^2}) \\
&+ T\Big(\lambda\frac{\partial^2 u_1}{\partial \xi_1 \partial \xi_3} + \lambda\frac{\partial^2 u_2}{\partial \xi_2 \partial \xi_3} + (2\mu+\lambda)\frac{\partial^2 u_3}{\partial \xi_3^2} \\
&+ \mu(\frac{\partial^2 u_1}{\partial \xi_3 \partial \xi_1}) + \frac{\partial^2 u_3}{\partial \xi_1^2} + \mu(\frac{\partial^2 u_2}{\partial \xi_3 \partial \xi_2} + \frac{\partial^2 u_3}{\partial \xi_2^2})\Big)
\end{aligned}
\tag{A.7}
$$

We solve Eqs. (A.5)–(A.7) by using the SPH methology. The spatial derivatives of the displacement based on the initial position can be approximated by the use of Eq.(2.20). As a result, particles carry both an initial position ξ_p as well as the current position $\mathbf{x}_p$ of the particle. They are related by

$$\mathbf{x}_p = \xi_p + \mathbf{d}_p, \tag{A.8}$$

where $\mathbf{d}_p$ is the displacement of the particle.

The particle velocity u_p evolves according to ODEs based on Eqs. (A.5)–(A.7) where the material derivative becomes a standard time derivative and the spatial derivatives with respect to the initial position are discretized using Eq. 2.20. Since the particle are initialized on a regular map, the particle overlap at all times and the accuracy is assured without remeshing.

A.1.1 Integration Method

Integration of the particles involves a third order Beeman integrator [15]

$$\mathbf{u}(t+\delta) = \mathbf{u}(t) + \delta(\frac{5}{3}\mathbf{a}(t) + \frac{2}{3}\mathbf{a}(t-\delta) - \frac{1}{12}\mathbf{a}(t-2\delta)) \quad \text{(A.9)}$$

$$\mathbf{x}(t+\delta) = \mathbf{x}(t) + \delta(\mathbf{u}(t) + \frac{2}{3}\mathbf{a}(t-\delta)\delta - \frac{1}{6}\mathbf{a}(t-2\delta)\delta) \quad \text{(A.10)}$$

Note that this scheme only uses one evaluation per time–step, in contrast to other high order methods, such as Runge–Kutta schemes. However, to obtain a higher order, the integration is based on an interpolation of the acceleration in time. This means that sudden changes in the topology (e.g. by cutting or pushing particles) can result in instabilities. For performance reasons, this integration scheme is used whenever possible. A simpler Euler–Cromer integrator (Eqs. (A.11)–(A.12)) is used temporarily when the particles are subject to sudden changes, as well as to initialize Eqs. (A.9)–(A.10) at the beginning of the simulation.

$$\mathbf{u}(t+\delta) = \mathbf{u}(t) + \mathbf{a}(t)\delta \quad \text{(A.11)}$$

$$\mathbf{x}(t+\delta) = \mathbf{x}(t) + \mathbf{u}(t+\delta)\delta \quad \text{(A.12)}$$

A.1.2 Visualization of the surface

We use the marching cubes algorithm to evaluate the surface based on the density distribution. The density is evaluated by

$$\rho(\mathbf{x}) = \sum_b m_b \zeta_\epsilon(\mathbf{x} - \mathbf{x_b}), \quad \text{(A.13)}$$

where the particle mass m_b is chosen to be $m_b = V_b \rho_{init}$. The initial density ρ_{init} is assumed to be 1.

For performance reasons, a simplified kernel was used for evaluating the density

$$\zeta_\epsilon(\mathbf{x}) = \frac{1}{\epsilon^3}(1 - \frac{\|\mathbf{x}\|}{\epsilon}). \tag{A.14}$$

The computational performance of the simulation is improved by the use of Cell Lists (cf. Section 6.5)

A.2 Boundary Conditions

To get a well posed problem, Eqs. (A.5)–(A.7) need to be associated with initial and boundary conditions. The initial condition describes the initial velocity profile of the material. Two different types of boundary conditions are considered.

- **Fixed boundaries**, used to clamp the material, enforce a given displacement at the boundary.
- **Stress–free boundaries** are boundaries where the surface traction is 0.

A.2.1 Ghost Particles

In this chapter, the implementation of the boundary conditions is based on ghost particles (ghosts), inspired by [162].
Ghosts are particles residing outside of the actual domain. They are purely passive particles that don't evolve. However, ghosts are accounted for in the superposition of Eq. (2.20). Furthermore, while they have specific initial positions ξ, their displacements $\mathbf{d}$ are adjusted on the fly according to the underlying boundary condition.

The advantage of this method lies in its efficiency, since it circumvents the need of one-sided differentiation.

In the following sections, the displacement field for a particle a is calculated. Its displacement is denoted as $\mathbf{d_a}$. Similarly, g denotes a ghost particle, with $\mathbf{d_g}$ being its displacement.

For simplicity, we consider the material in the shape of a cube. Ghost particles are added in layers around the cube, where the number of layers needed is determined by the support radius of the kernel.

A.2.2 Fixed Boundary

The goal of a fixed boundary condition to impose a specific displacement at the boundary. Since particles are, in general, not located on the boundary, the idea is obtain the desired displacement by adjusting the ghost particles displacement outside of the domain accordingly.
During the calculation of the particle evolution based on Eqs. (A.5)–(A.7), the displacement $\mathbf{d_g}$ of ghost particle g is linearly extrapolated on the fly such that the displacement on the boundary satisfies the boundary condition.

For instance, if the material is clamped to a horizontal $y-z$ plane at position ξ_{plane}, with a displacement of $\mathbf{d_{boundary}}$, one obtains Eq. (A.15). ξ and ξ_a denote the x-component of the initial position.
Figure A.1 shows a one-dimensional equivalent.

$$\mathbf{d_g} = \frac{\xi_g - \xi_a}{\xi_{plane} - \xi_a}(\mathbf{d_{boundary}} - \mathbf{d_a}) + \mathbf{d_a} \tag{A.15}$$

A.2.3 Stress-free Boundary

On a stress-free boundary, the displacement quantities have to behave, such that no force acts on the surface. The stress-free boundary ghosts behave like a passive, unstretched material that is glued to the real soft material.

This is accomplished by linking each ghost particle to an actual particle on the boundary of the domain, called neighbor. Whenever ghost particle g is consulted, its displacement

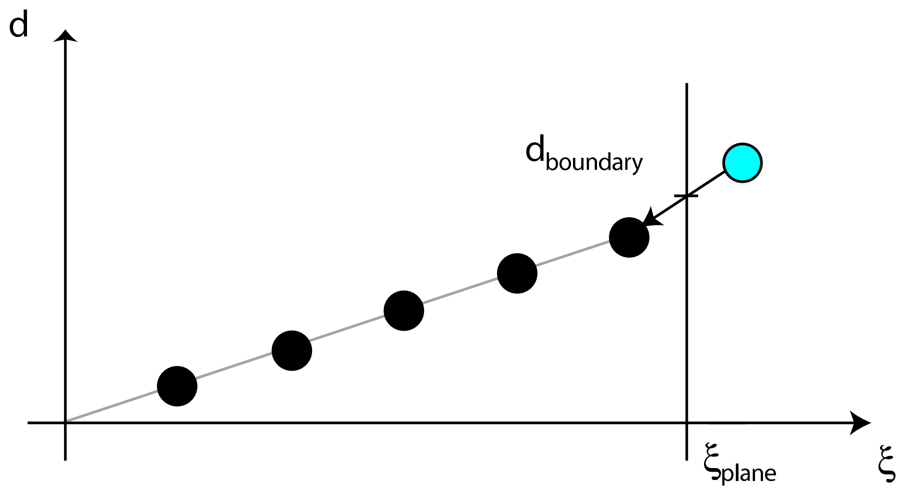


Figure A.1: Fixed boundary using ghost particles in 1D

$\mathbf{d_g}$ is equal to the displacement of this neighbor.

$$\mathbf{d_g} = \mathbf{d_n} \tag{A.16}$$

Figure A.2 shows how ghost particles are linked to parent material particles for different boundaries. Note that for the third case, it is not clear where g should adjust to when considered during the calculation of particle d. Best results were achieved by using an average of g's two orthogonal neighbors a and b.

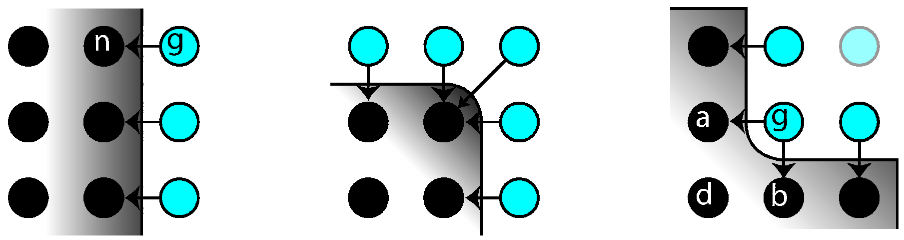


Figure A.2: Stress–free boundary using ghost particles

A.3 Virtual Cutting Using Ghost Particles

During the cutting processes material is split into parts creating new boundaries with the stress-free boundary condition holds. In this section, two approaches based on ghost particles are described to realize virtual cutting.

A.3.1 Basic Idea

At a stress–free boundary, a ghost particle usually adapts to on neighbor particle. During cutting, new stress-free boundaries are created between particles. Thus, ghost particles at the cut have at least two direct neighbors, as shown in Figure A.3. In the following, $\mathcal{N}$ denotes the set of neighbors for a ghost particle.

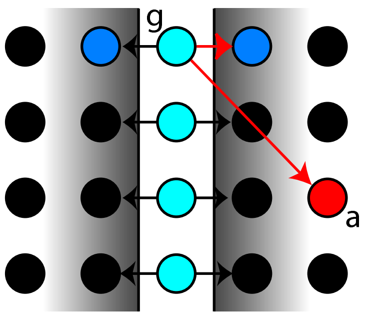


Figure A.3: Ghost particles in a cut

When a ghost is considered during the SPH approximation of Eqs. (A.5)–(A.7) for a particle a, the displacement of the closest material neighbor particle is considered. To determine the closest neighbor, we need to compute the normalized vectors from the ghost particle g to the particle a (Eq. (A.17)) to the set of neighbor particles (Eq. (A.18)).

$$\mathbf{d1} = \frac{\xi_\mathbf{a} - \xi_\mathbf{g}}{\|\xi_\mathbf{a} - \xi_\mathbf{g}\|} \tag{A.17}$$

$$\mathbf{d2_n} = \frac{\xi_\mathbf{n} - \xi_\mathbf{g}}{\|\xi_\mathbf{n} - \xi_\mathbf{g}\|}, \text{for all } n \in \mathcal{N} \tag{A.18}$$

During the evaluation of the evolution of particle a, the ghost g adapt to the neighbor maximizing the dot product of Eqs. (A.17) and (A.18).

$$m = \arg\max_{n \in N} \mathbf{d1} \cdot \mathbf{d2_n} \tag{A.19}$$

If the maximum is ambivalent (i.e., two or more neighbors have the same dot product up to tolerance ϵ_{tol}), the displacements of the corresponding neighbors are averaged accordingly.

A.3.2 Cutting by Converting Particles

A first approach for cutting is based on the conversion of particles inside the scalpel into ghost particles. The ghost particle are linked to their immediate neighbors on the fly. Figure A.4 shows an two-dimensional example. In a first step, the red particle p affected by the cut is converted into a ghost particle. This means that a new stress–free boundary is created for all of its nearest neighbors. Furthermore, the ghost particle g that previously kept p as a neighbor has to search for a new set of neighbors, since it cannot adapt to a ghost particle.

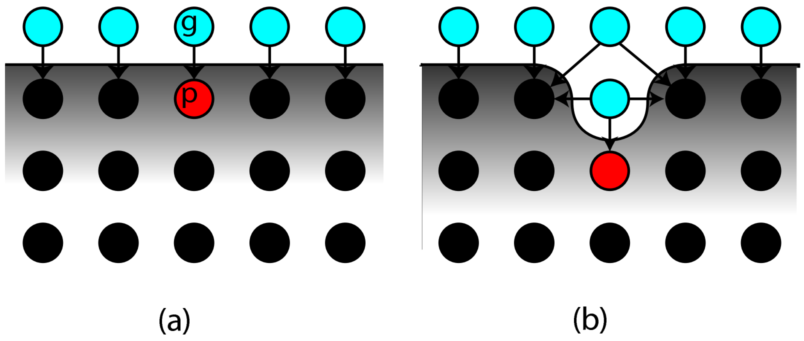


Figure A.4: Cutting by converting particles

This yields the following algorithm when converting a material particle p to a ghost particle:

1. Change type of p to ghost particle.
2. Find immediate neighbors of p.
3. For all ghost particles g that had a link to particle p:
   (a) Clear neighborhood of g.
   (b) Find nearest material particle(s), link them.

The immediate neighbors of step 2 are defined as the 6 closest neighbors of the particle in x, y, z-direction. Diagonal linkages did not yield better results with respect to stability.

As mentioned above, it can happen that Eq.(A.19) has not only one maximum—as shown in case (c) of Figure A.2. As an approximation, the averaged displacement of the two closest neighbors is chosen.

Obviously, volume is going to be lost when transforming material particles into ghost particles. A local refinement of the cutting region may reduce this effect.

A.3.3 Cutting by Splitting Particles

Alternatively, we can split material particle by introducing *hybrid particles* in the vicinity of a cut. Hybrid particles can behave as material or ghost particles, depending on the situation. In particle interactions crossing a cut hybrid particles act as ghost particles, otherwise as a material particle.

The consideration a hybrid particle p_h during the particle superpositions for a particle p, therefore, requires a check where the particle h_p is located with respect to the cut. If it is on the same side as particle p, the hybrid particle h_p will behave like a standard material particle, otherwise, it mirrors the displacement of the particle on the other side of the cut, thus behaving like a ghost particle for a stress–free boundary. (Figure A.5)

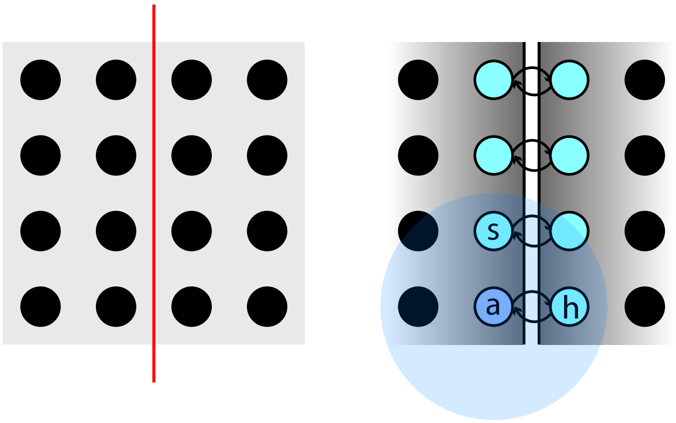


Figure A.5: Cutting between particles using hybrid particles

As shown in Figure A.3.3, the hybrid particle h keeps a list of all the particles within the kernel support. For all the particles residing on the same side of the cuts as particle h (5, 7, 8), h behaves like a standard material particle. For the other particles, the particle h will return the displacement of the particles indicated by the arrows. Thus, h will behave like a stress–free boundary in x–direction for particles 4 and 6, and like a stress–free boundary in y–direction for particles 2 and 3. For the outer edge particle 1, particle h will behave as shown in case (b) of Figure A.2.

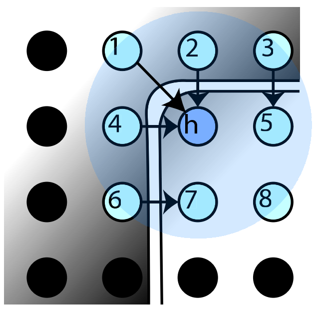


Figure A.6: Hybrid particle h and its interactions to neighboring particles

Cut Surface

The cut surface is defined as the surface that separates two parts of an object that are to be splitted [21]. It is inferred from the movement of the scalpel, as can be seen in Fig. A.7.

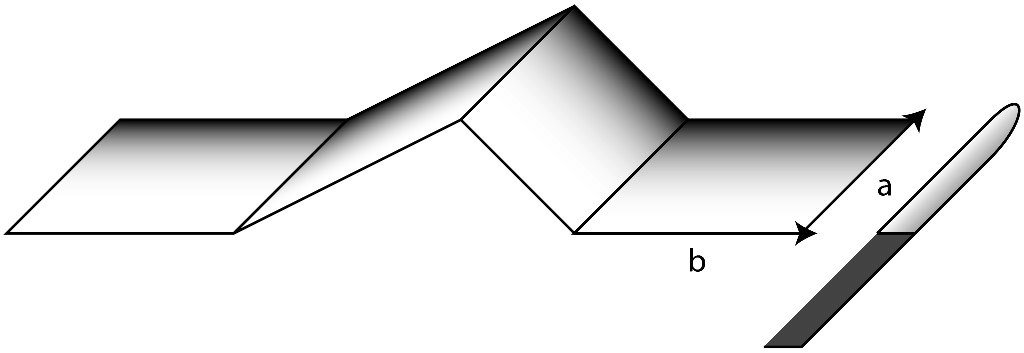


Figure A.7: Cut surface defined by scalpel

During the cutting process, the algorithm has to decide which particles are to be separated by the cut surface. To simplify this decision, the cut surface is assumed to be planar and rectangular.
In this thesis, only the generation of cut surfaces by translation is supported, but the addition of rotational effects should not entail major changes.

The cut surface is defined a series of parallelograms. One parallelogram is described by a position $\mathbf{x}$ and two edges represented by the vectors $\mathbf{a}$ and $\mathbf{b}$. The vector $\mathbf{a}$ corresponds to the cutting edge of the scalpel. The vector $\mathbf{b}$ is defined as $\mathbf{b} = \mathbf{x_2} - \mathbf{x_1}$, where $\mathbf{x_1}$ denotes the start point and $\mathbf{x_2}$ the end point of a straight cut. These are obtained from the position of the scalpel at two subsequent time–steps t_1 and t_2.
The vectors $\mathbf{e}_a$ and $\mathbf{e}_b$ are the unit vectors of $\mathbf{a}$ and $\mathbf{b}$, respectively.

To test whether a particle p with position $\mathbf{x}_p$ is affected by the cut, its position is projected onto the cut surface using the dot product. Thus, if both $(\mathbf{x} - \mathbf{x_1}) \cdot \mathbf{e}_a$ and

$(\mathbf{x} - \mathbf{x_1}) \cdot \mathbf{e}_b$ are within the bounds specified by the cut surface's size, particle p is converted into a hybrid particle .

During the cutting process, the collision detection routine presented in [78] is used to check whether the scalpel is inside the organ. If this is the case, cut surfaces are generated as described above and stored in a list.

Note that the size of the time–step influences the sampling the scalpel positions. Large time–steps can introduce heavy lags for the user. On the other hand, small time–steps can not only impact computational performance, but also affect the detection of diagonal edges. Two measures were taken: first, a cut surface is issued when the scalpel moves by a distance larger than $h\sqrt{3}$, where h denotes the particle spacing. Second, cut surfaces are chosen to overlap, i.e. if for the first cut surface, $\mathbf{b} = \mathbf{x_2} - \mathbf{x_1}$, the second cut surface will encompass $\mathbf{b} = \mathbf{x_3} - \mathbf{x_1} + \frac{1}{2}(\mathbf{x_2} - \mathbf{x_1})$. This improves detection of diagonal edges without introducing large modification of the cut surfaces.

Algorithm

The following algorithm that is used to infer the linking information from the cut surface. Only the current cut surface is considered—previous cuts are inherent in the hybrid particles.

1. Find particles whose projection fall into the cut surface, discarding particles do not reside in the kernel support.

2. Sort particles into two sets $\mathcal{A}$ and $\mathcal{B}$, depending on the side of the cut surface they are.

3. Find subsets $\mathcal{A}_{min}$ and $\mathcal{B}_{min}$ of particles that are nearest to the cut surface—these define the actual cut in the material.

4. For each particle $a \in \mathcal{A}$

For each particle s within particle a's support radius, check particle s with respect to the cut surface.

- Not at the cut surface or on same side as particle a: the relationship between the particles s and a doesn't change.
- On other side of cut surface: particle s has to act like a ghost particle with respect to particle a. Thus:
  (a) If s is not yet a hybrid particle, convert particle s into a hybrid particle.
  (b) Find nearest neighbor of particle s on particle a's side of cut, $n \in \mathcal{A}_{min}$.
  (c) As an action when invoked by particle a, particle s takes the displacement $\mathbf{d_n}$ of particle n, i.e. behaves like a stress–free boundary particle.

5. Do the same for each particle $b \in \mathcal{B}$

A.4 Results

This section contains a few examples of the cutting algorithms. For all examples, the same basic setup is chosen:
A set of 729 particles reside in a cube with an additional margin of 2 ghost particles on all sides, resulting in 2197 particles in total. The kernel support contains the 26 direct neighbors of a particle.
The cube is pre-stretched along the x–axis with a small displacement of $d(\xi) = 0.1\xi$ using two wall. The other boundaries are stress–free. The time–step of the simulation is chosen to be $\Delta t = 0.01$.

A.4.1 Cutting by Converting Particles

Fig. A.8 shows the simulation of a cutting process by converting particles. An incomplete perforation of the cube is done along the z–axis. Two particles are left untouched. The material properties are:

- Poisson's ratio $\nu = 0$
- Young's modulus $E = 1$
- Relaxation time $T = 0.01$
- Gravity constant = 0

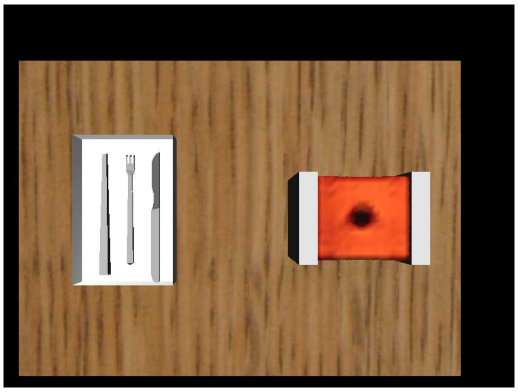


Figure A.8: Cutting by converting particles

As expected, material is lost at the location of the perforation. However, the hole doesn't collapse in itself, due to the stress–free boundary ghosts replacing the original material particles. The simulation is stable, even with a low damping rate with a relaxation time of $T = 0.01$.

A.4.2 Cutting by Splitting Particles

We consider two test cases with different cutting directions. Fig. A.9 demonstrates the process of cutting the cube apart along the z–axis. The material properties are:

- Poisson's ratio $\nu = 0.3$
- Young's modulus $E = 0.7$
- Relaxation time $T = 0.2$
- Gravity constant = 0.1

The cube separates nicely due to the clamping of the material, and the simulation remains stable. At an optimization level of `O2`, the simulation ran at about 17 frames per second.

The other example involves a diagonal cut on the surface, as shown in Fig. A.10. Again, the simulation remains stable and produces the a straight cut. Again, about 17 frames per second were achieved in the simulation.

- Poisson's ratio $\nu = 0$
- Young's modulus $E = 1$
- Relaxation time $T = 0.1$
- Gravity constant = 0.1

A.5 Discussion

We achieve a computational efficient particle simulation of the elastic solid by describing the particle model based on the initial positions where the material is undeformed. As this reference configuration is rigid in time, the particles always overlap and remeshing techniques can be omitted. The boundary treatment using ghost particle is a simple approach to impose stress–free and fixed boundary conditions in one-dimension. Its extension to higher dimension, however, is not trivial. A straight forward extension as used in this chapter leads to the required zero nominal stress at a free surface, but does not ensure

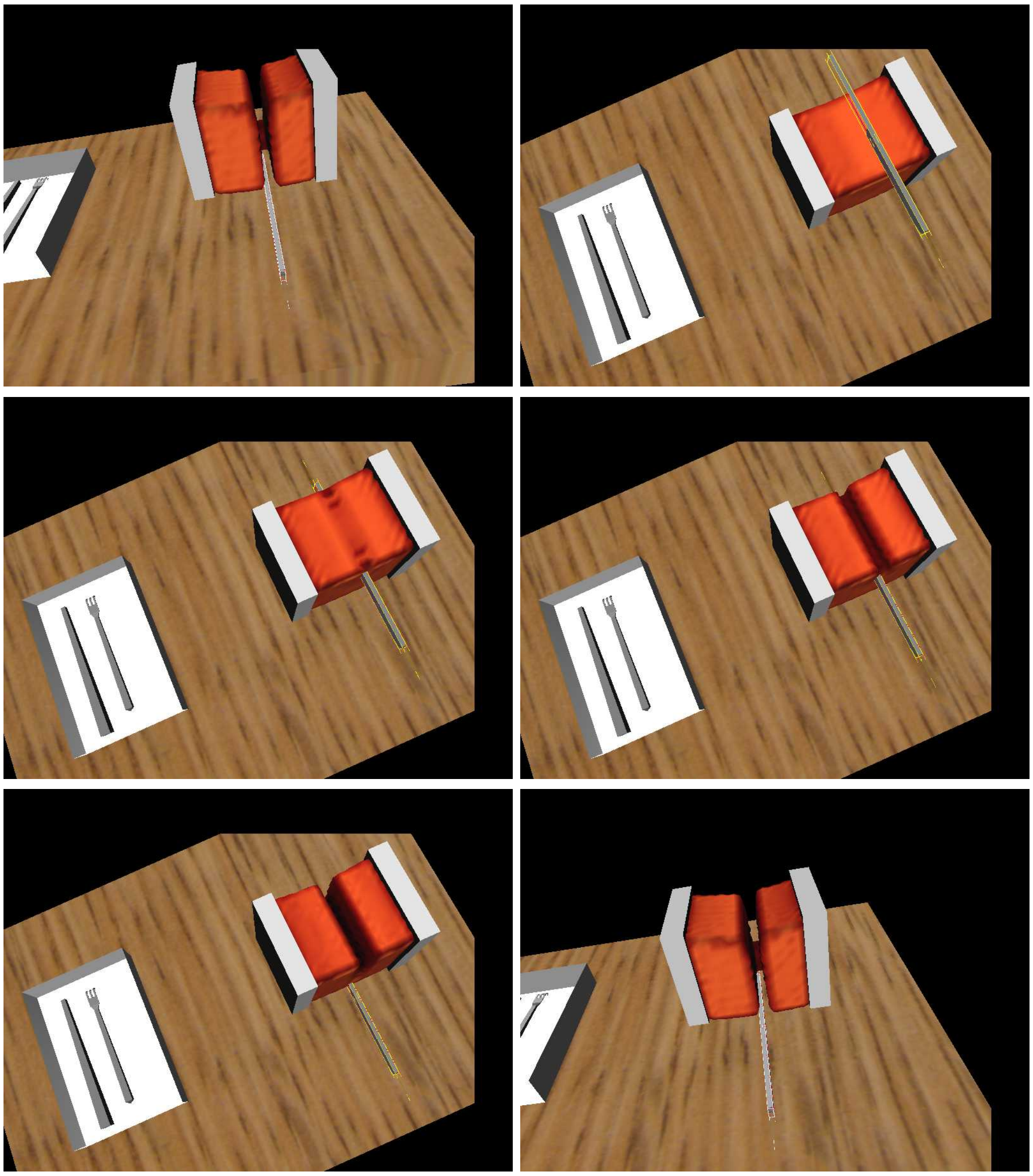


Figure A.9: Cutting by splitting particles—vertical cut

zero shear stresses. As a result, the material surface does not remain perpendicular to the wall it is attached to. Therefore, this approach was not further pursued within this thesis.

However, both cutting algorithms work well and demonstrate the possibility to cut material within a particle framework. An advantage of the particle approach is that no changes to the geometry of the computational elements are necessary as in grid–based methods,

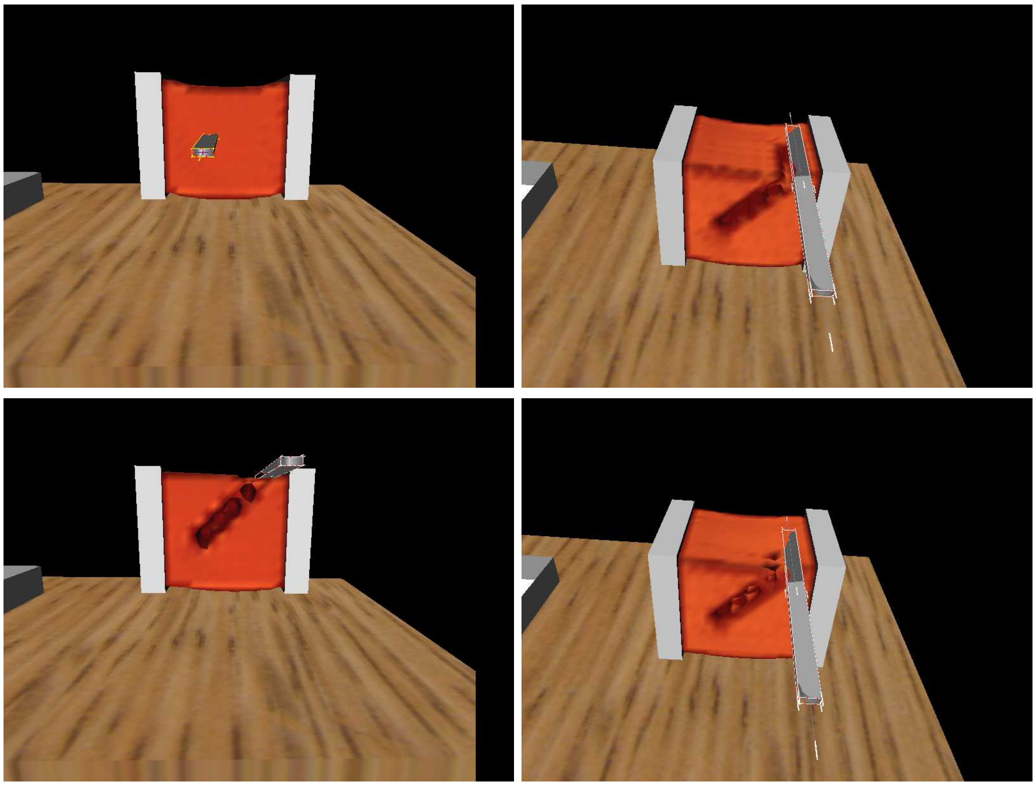


Figure A.10: Cutting by splitting particles—diagonal cut

such as finite elements.

Appendix B

Higher Order Kernels

In this chapter, we present a large collection of kernels η^{β} in two and three dimensions:

$$\eta^{(\alpha,\beta)(\mathbf{x})} = \frac{\partial^{\alpha+\beta}\eta(\mathbf{x})}{\partial x_1^{\alpha}\partial x_2^{\beta}} \tag{B.1}$$

$$\eta^{(\alpha,\beta,\gamma)(\mathbf{x})} = \frac{\partial^{\alpha+\beta+\gamma}\eta(\mathbf{x})}{\partial x_1^{\alpha}\partial x_2^{\beta}\partial x_3^{\gamma}} \tag{B.2}$$

They satisfy the moment conditions in Section 2.3 and are derived following the methodology of Eldredge *et al.* [54]. The kernels η_{ϵ}^{β} are defined by

$$\eta_{\epsilon}(\mathbf{x}) = \frac{1}{\epsilon^{d}}\eta(|\mathbf{x}|/\epsilon), \tag{B.3}$$

where d is the dimension and $\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ and $|\mathbf{x}| = \sqrt{x_1^2 + x_2^2}$ in two dimensions and $\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$ and $|\mathbf{x}| = \sqrt{x_1^2 + x_2^2 + x_3^2}$ in three dimensions.

B.1 Kernels in 2D

- Second order

$$\eta^{(0,0)}(\mathbf{x}) = \frac{1}{\pi} e^{-|x|^2} \tag{B.4}$$

$$\eta^{(1,0)}(\mathbf{x}) = (-2x_1) \frac{1}{\pi} e^{-|x|^2} \tag{B.5}$$

$$\eta^{(0,1)}(\mathbf{x}) = (-2x_2) \frac{1}{\pi} e^{-|x|^2} \tag{B.6}$$

$$\eta^{(2,0)}(\mathbf{x}) = \left(-2 + 4x_1^2\right) \frac{1}{\pi} e^{-|x|^2} \tag{B.7}$$

$$\eta^{(0,2)}(\mathbf{x}) = \left(-2 + 4x_2^2\right) \frac{1}{\pi} e^{-|x|^2} \tag{B.8}$$

$$\eta^{(1,1)}(\mathbf{x}) = (4x_1 x_2) \frac{1}{\pi} e^{-|x|^2} \tag{B.9}$$

- Forth order

$$\eta^{(0,0)}(\mathbf{x}) = \left(2 - |x|^2\right) \frac{1}{\pi} e^{-|x|^2} \tag{B.10}$$

$$\eta^{(1,0)}(\mathbf{x}) = \left(-6 + 2|x|^2\right) x_1 \frac{1}{\pi} e^{-|x|^2} \tag{B.11}$$

$$\eta^{(0,1)}(\mathbf{x}) = \left(-6 + 2|x|^2\right) x_2 \frac{1}{\pi} e^{-|x|^2} \tag{B.12}$$

$$\eta^{(2,0)}(\mathbf{x}) = \left(6 + 6x_1^2 - 10x_2^2 - 2x_1^4 + 2x_2^4\right) \frac{1}{\pi} e^{-|x|^2} \tag{B.13}$$

$$\eta^{(0,2)}(\mathbf{x}) = \left(6 + 6x_2^2 - 10x_1^2 - 2x_2^4 + 2x_1^4\right) \frac{1}{\pi} e^{-|x|^2} \tag{B.14}$$

$$\eta^{(1,1)}(\mathbf{x}) = \left(16 - 4|x|^2\right) x_1 x_2 \frac{1}{\pi} e^{-|x|^2} \tag{B.15}$$

- Eighth order

$$\eta^{(0,0)}(\mathbf{x}) = \left(4 - 6|x|^2 + 2|x|^4 + \frac{1}{6}|x|^6\right)\frac{1}{\pi}e^{-|x|^2} \tag{B.16}$$

$$\eta^{(1,0)}(\mathbf{x}) = \left(-20 - 20|x|^2 - 5|x|^4 + \frac{1}{3}|x|^6\right)x_1\frac{1}{\pi}e^{-|x|^2} \tag{B.17}$$

$$\eta^{(0,1)}(\mathbf{x}) = \left(-20 - 20|x|^2 - 5|x|^4 + \frac{1}{3}|x|^6\right)x_2\frac{1}{\pi}e^{-|x|^2} \tag{B.18}$$

$$\begin{aligned}\eta^{(2,0)}(\mathbf{x}) = \big(&20 + 20x_1^2 - 60x_2^2 - 25x_1^4 + 10x_1^2x_2^2 + 35x_2^4\\ &- \frac{17}{3}x_1^6 + 5x_1^4x_2^2 - 7x_1^2x_2^4 - \frac{19}{3}x_2^6\\ &-\frac{1}{3}x_1^8 - \frac{2}{3}x_1^6x_2^2 + \frac{2}{3}x_1^2x_2^6 - \frac{1}{3}x_2^8\Big)\frac{1}{\pi}e^{-|x|^2}\end{aligned} \tag{B.19}$$

$$\begin{aligned}\eta^{(0,2)}(\mathbf{x}) = \big(&20 + 20x_2^2 - 60x_1^2 - 25x_2^4 + 10x_2^2x_1^2 + 35x_1^4\\ &- \frac{17}{3}x_2^6 + 5x_2^4x_1^2 - 7x_2^2x_1^4 - \frac{19}{3}x_1^6\\ &-\frac{1}{3}x_2^8 - \frac{2}{3}x_2^6x_1^2 + \frac{2}{3}x_2^2x_1^6 - \frac{1}{3}x_1^8\Big)\frac{1}{\pi}e^{-|x|^2}\end{aligned} \tag{B.20}$$

B.2 Kernels in 3D

- Second order

$$\eta^{(0,0,0)}(\mathbf{x}) = \frac{1}{\pi\sqrt{\pi}} e^{-|x|^2} \tag{B.21}$$

$$\eta^{(1,0,0)}(\mathbf{x}) = (-2x_1)\,\frac{1}{\pi\sqrt{\pi}} e^{-|x|^2} \tag{B.22}$$

$$\eta^{(0,1,0)}(\mathbf{x}) = (-2x_2)\,\frac{1}{\pi\sqrt{\pi}} e^{-|x|^2} \tag{B.23}$$

$$\eta^{(0,0,1)}(\mathbf{x}) = (-2x_3)\,\frac{1}{\pi\sqrt{\pi}} e^{-|x|^2} \tag{B.24}$$

$$\eta^{(2,0,0)}(\mathbf{x}) = \left(-2 + 4x_1^2\right)\frac{1}{\pi\sqrt{\pi}} e^{-|x|^2} \tag{B.25}$$

$$\eta^{(0,2,0)}(\mathbf{x}) = \left(-2 + 4x_2^2\right)\frac{1}{\pi\sqrt{\pi}} e^{-|x|^2} \tag{B.26}$$

$$\eta^{(0,0,2)}(\mathbf{x}) = \left(-2 + 4x_3^2\right)\frac{1}{\pi\sqrt{\pi}} e^{-|x|^2} \tag{B.27}$$

$$\eta^{(1,1,0)}(\mathbf{x}) = (4x_1x_2)\,\frac{1}{\pi\sqrt{\pi}} e^{-|x|^2} \tag{B.28}$$

$$\eta^{(1,0,1)}(\mathbf{x}) = (4x_1x_3)\,\frac{1}{\pi\sqrt{\pi}} e^{-|x|^2} \tag{B.29}$$

$$\eta^{(0,1,1)}(\mathbf{x}) = (4x_2x_3)\,\frac{1}{\pi\sqrt{\pi}} e^{-|x|^2} \tag{B.30}$$

- Forth order

$$\eta^{(0,0,0)}(\mathbf{x}) = \left(\frac{5}{2} - |x|^2\right) \frac{1}{\pi\sqrt{\pi}} e^{-|x|^2} \tag{B.31}$$

$$\eta^{(1,0,0)}(\mathbf{x}) = \left(-7 + 2|x|^2\right) x_1 \frac{1}{\pi\sqrt{\pi}} e^{-|x|^2} \tag{B.32}$$

$$\eta^{(0,1,0)}(\mathbf{x}) = \left(-7 + 2|x|^2\right) x_2 \frac{1}{\pi\sqrt{\pi}} e^{-|x|^2} \tag{B.33}$$

$$\eta^{(0,0,1)}(\mathbf{x}) = \left(-7 + 2|x|^2\right) x_3 \frac{1}{\pi\sqrt{\pi}} e^{-|x|^2} \tag{B.34}$$

$$\eta^{(2,0,0)}(\mathbf{x}) = \left(\frac{21}{2} + 6x_1^2 - 12(x_2^2 + x_3^2) - 2x_1^4\right. \tag{B.35}$$

$$\left. +2(x_2^4 + x_3^4) + 4x_2^2 x_3^2\right) \frac{1}{\pi\sqrt{\pi}} e^{-|x|^2} \tag{B.36}$$

$$\eta^{(0,2,0)}(\mathbf{x}) = \left(\frac{21}{2} + 6x_2^2 - 12(x_1^2 + x_3^2) - 2x_2^4\right. \tag{B.37}$$

$$\left. +2(x_1^4 + x_3^4) + 4x_1^2 x_3^2\right) \frac{1}{\pi\sqrt{\pi}} e^{-|x|^2} \tag{B.38}$$

$$\eta^{(0,0,2)}(\mathbf{x}) = \left(\frac{21}{2} + 6x_3^2 - 12(x_1^2 + x_2^2) - 2x_3^4\right. \tag{B.39}$$

$$\left. +2(x_1^4 + x_2^4) + 4x_1^2 x_2^2\right) \frac{1}{\pi\sqrt{\pi}} e^{-|x|^2} \tag{B.40}$$

$$\eta^{(1,1,0)}(\mathbf{x}) = \left(18 - 4|x|^2\right) x_1 x_2 \frac{1}{\pi\sqrt{\pi}} e^{-|x|^2} \tag{B.41}$$

$$\eta^{(1,0,1)}(\mathbf{x}) = \left(18 - 4|x|^2\right) x_1 x_3 \frac{1}{\pi\sqrt{\pi}} e^{-|x|^2} \tag{B.42}$$

$$\eta^{(0,1,1)}(\mathbf{x}) = \left(18 - 4|x|^2\right) x_2 x_3 \frac{1}{\pi\sqrt{\pi}} e^{-|x|^2} \tag{B.43}$$

- Sixth order

$$\eta^{(0,0,0)}(\mathbf{x}) = \left(\frac{35}{8} - \frac{7}{2}|x|^2 + \frac{1}{2}|x|^4\right)\frac{1}{\pi\sqrt{\pi}}e^{-|x|^2} \tag{B.44}$$

$$\eta^{(1,0,0)}(\mathbf{x}) = \left(-\frac{63}{4} + 9|x|^2 - |x|^4\right)x_1\frac{1}{\pi\sqrt{\pi}}e^{-|x|^2} \tag{B.45}$$

$$\eta^{(0,1,0)}(\mathbf{x}) = \left(-\frac{63}{4} + 9|x|^2 - |x|^4\right)x_2\frac{1}{\pi\sqrt{\pi}}e^{-|x|^2} \tag{B.46}$$

$$\eta^{(0,0,1)}(\mathbf{x}) = \left(-\frac{63}{4} + 9|x|^2 - |x|^4\right)x_3\frac{1}{\pi\sqrt{\pi}}e^{-|x|^2} \tag{B.47}$$

$$\eta^{(2,0,0)}(\mathbf{x}) = \left(\frac{603}{32} + \frac{45}{16}x_1^2 - \frac{381}{16}(x_2^2 + x_3^2) - \frac{15}{8}x_1^4\right. \tag{B.48}$$

$$+\frac{9}{2}(x_2^4 + x_3^4) + \frac{91}{4}x_2^2x_3^2 + \frac{5}{4}x_1^4x_2^2 \tag{B.49}$$

$$\left.-\frac{3}{4}x_1^2x_2^4 - 3x_2^2x_3^4 - \frac{11}{4}x_2^4x_3^2\right)\frac{1}{\pi\sqrt{\pi}}e^{-|x|^2} \tag{B.50}$$

$$\eta^{(0,2,0)}(\mathbf{x}) = \left(\frac{603}{32} + \frac{45}{16}x_2^2 - \frac{381}{16}(x_1^2 + x_3^2) - \frac{15}{8}x_2^4\right. \tag{B.51}$$

$$+\frac{9}{2}(x_1^4 + x_3^4) + \frac{91}{4}x_1^2x_3^2 + \frac{5}{4}x_2^4x_1^2 \tag{B.52}$$

$$\left.-\frac{3}{4}x_2^2x_1^4 - 3x_1^2x_3^4 - \frac{11}{4}x_1^4x_3^2\right)\frac{1}{\pi\sqrt{\pi}}e^{-|x|^2} \tag{B.53}$$

$$\eta^{(0,0,2)}(\mathbf{x}) = \left(\frac{603}{32} + \frac{45}{16}x_3^2 - \frac{381}{16}(x_1^2 + x_2^2) - \frac{15}{8}x_3^4\right. \tag{B.54}$$

$$+\frac{9}{2}(x_1^4 + x_2^4) + \frac{91}{4}x_1^2x_2^2 + \frac{5}{4}x_3^4x_1^2 \tag{B.55}$$

$$\left.-\frac{3}{4}x_3^2x_1^4 - 3x_1^2x_2^4 - \frac{11}{4}x_1^4x_2^2\right)\frac{1}{\pi\sqrt{\pi}}e^{-|x|^2} \tag{B.56}$$

$$\eta^{(1,1,0)}(\mathbf{x}) = \left(\frac{99}{2} - 22|x|^2 + 2|x|^4\right)x_1x_2\frac{1}{\pi\sqrt{\pi}}e^{-|x|^2} \tag{B.57}$$

$$\eta^{(1,0,1)}(\mathbf{x}) = \left(\frac{99}{2} - 22|x|^2 + 2|x|^4\right)x_1x_3\frac{1}{\pi\sqrt{\pi}}e^{-|x|^2} \tag{B.58}$$

$$\eta^{(0,1,1)}(\mathbf{x}) = \left(\frac{99}{2} - 22|x|^2 + 2|x|^4\right)x_2x_3\frac{1}{\pi\sqrt{\pi}}e^{-|x|^2} \tag{B.59}$$

Appendix C

Moving Frameworks for Compressible Fluids

In the simulation of anguilliform swimming, the domain represents a frame moving with the swimmer body. In this chapter, we prove according to Panton [122] that we can accelerate the fluid with the forces acting on the body while the body itself remains at the initial position of the noninertial coordinate system.

Let $x_{i,t}$ be the an inertial reference frame. The origin of the noninertial frame is moving with the velocity $U_i(t)$, which is arbitrary in both magnitude and direction. Rotation of the system $\hat{x}_i$ is not allowed. We will prove that the same equations govern the flow in the moving system as in the inertial system. The coordinates and velocities are related by the transformations below

$$\hat{x}_i = x_i - \int_0^t U_i(\tau) d\tau \tag{C.1}$$

$$\hat{t} = t \tag{C.2}$$

$$\hat{u}_i = u_i - U_i. \tag{C.3}$$

The partial derivatives for $f(\hat{x}(x,t), \hat{t})$ are given by

$$\frac{\partial}{\partial x_i} = \frac{\partial}{\partial \hat{x}_i} \tag{C.4}$$

$$\frac{\partial}{\partial t} = \frac{\partial}{\partial \hat{t}} - U_i \frac{\partial}{\partial \hat{x}_i}. \tag{C.5}$$

(C.6)

The continuity and momentum equation in inertial coordinates is

$$\begin{aligned}
\frac{\partial\rho}{\partial t} + u_i\frac{\partial\rho}{\partial x_i} &= \frac{\partial u_i}{\partial x_i} \\
\frac{\partial u_j}{\partial t} + u_i\frac{\partial u_j}{\partial x_i} &= -\frac{1}{\rho}\left(\frac{\partial p}{\partial x_j} + \frac{\partial \tau_{ij}}{\partial x_i}\right) \\
\tau_{ij} &= \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\delta_{ij}\frac{\partial u_k}{\partial x_k}\right)
\end{aligned} \tag{C.7}$$

This set of equations transforms using Eqs.(C.1)-(C.6) into

$$\begin{aligned}
\frac{\partial\rho}{\partial\hat{t}} - U_i\frac{\partial\rho}{\partial\hat{x}_i} + (\hat{u}_i + U_i)\frac{\partial\rho}{\partial\hat{x}_i} &= \frac{\partial u_i}{\partial\hat{x}_i} \\
\frac{\partial(\hat{u}_j + U_i)}{\partial\hat{t}} - U_i\frac{\partial(\hat{u}_j + U_i)}{\partial\hat{x}_i} + (\hat{u}_i + U_i)\frac{\partial\hat{u}_j}{\partial\hat{x}_i} &= -\frac{1}{\rho}\left(\frac{\partial p}{\partial\hat{x}_j} + \frac{\partial\hat{\tau}_{ij}}{\partial\hat{x}_i}\right) \\
\hat{\tau}_{ij} &= \mu\left(\frac{\partial\hat{u}_i}{\partial\hat{x}_j} + \frac{\partial\hat{u}_j}{\partial\hat{x}_i} - \frac{2}{3}\delta_{ij}\frac{\partial\hat{u}_k}{\partial\hat{x}_k}\right)
\end{aligned} \tag{C.8}$$

resulting into

$$\begin{aligned}
\frac{\partial\rho}{\partial\hat{t}} + \hat{u}_i\frac{\partial\rho}{\partial\hat{x}_i} &= \frac{\partial u_i}{\partial\hat{x}_i} \\
\frac{\partial\hat{u}_j}{\partial\hat{t}} + \hat{u}_i\frac{\partial\hat{u}_j}{\partial\hat{x}_i} &= -\frac{1}{\rho}\left(\frac{\partial p}{\partial\hat{x}_j} + \frac{\partial\hat{\tau}_{ij}}{\partial\hat{x}_i}\right) - \frac{\partial U_i}{\partial\hat{t}} \\
\hat{\tau}_{ij} &= \mu\left(\frac{\partial\hat{u}_i}{\hat{x}_j} + \frac{\partial\hat{u}_j}{\hat{x}_i} - \frac{2}{3}\delta_{ij}\frac{\partial\hat{u}_k}{\partial\hat{x}_k}\right).
\end{aligned} \tag{C.9}$$

The continuity equation remains in the same form whereas the momentum equation includes an additional acceleration term ($-\frac{\partial U_i}{\partial\hat{t}}$). This term accounts for the acceleration of the body $\frac{\partial U_i}{\partial\hat{t}}$ and is, as expected, applied in opposite direction.

Appendix D

Bulk Viscosity

The bulk viscosity describes the viscosity effects with respect to volume change of a material and is associated with the material pressure. The evaluation of the pressure involves a time convolution with the time variant relaxation modulus $Y(t)$ [1]

$$p(t) = Y(0)\tilde{J}(t) + \int_0^t \dot{Y}(t)\tilde{J}(t-\tau)d\tau. \tag{D.1}$$

where $\tilde{J}(t) = (J(t) - 1)$. The direct numerical evaluation of the convolution in time is computationally expensive and memory-consuming because it requires the storage of volume change history $J(t)$. In present study, the relaxation modulus Y(t) is described by a Prony Series of order K [1]

$$Y(t) \quad = \quad \underbrace{\frac{Y^\infty}{1 - \sum_{k=1}^K \bar{g}_k^P}}_{\bar{Y}} \left(1 - \sum_{k=1}^K \bar{g}_k^P \left(1 - e^{-\frac{t}{\tau_k}}\right)\right), \tag{D.2}$$

where Y^∞ is the long term elastic modulus, τ_k are the characteristic times, and $\bar{g}_k^P$ are the Prony coefficients. Transforming Eq.(D.1) into Laplace space yields in consideration of Eq.(D.2):

$$\begin{aligned}
p(s) &= sY(s) \cdot \tilde{J}(s) \\
&= s\bar{Y} \left(\frac{1}{s} - \sum_{k=1}^K \bar{g}_k^P \frac{1}{s(1+\tau_k s)}\right) \cdot \tilde{J}(s) \\
&= \left(1 - \sum_{k=1}^K \bar{g}_k^P \frac{1}{1+\tau_k s}\right) \cdot \bar{Y}\tilde{J}(s) \\
&= \sum_{k=0}^K \bar{p}_k(s)
\end{aligned} \tag{D.3}$$

where

$$\bar{p}_0(s) = \bar{Y}\tilde{J}(s), \tag{D.4}$$

$$\bar{p}_k(s) = -\frac{\bar{g}_k^P}{1+\tau_k s}\bar{Y}\tilde{J}(s), \quad k = 1..K. \tag{D.5}$$

Eq.(D.5) is equivalent to a set of ODE in real space when it is rewritten as

$$\bar{p}_k(s) + \tau_k s \bar{p}_k(s) = -\bar{g}_k^P \bar{Y}\tilde{J}(s), \quad k = 1..K. \tag{D.6}$$

Finally, we obtain the pressure $p(t)$ in real space as composition of $K+1$ terms $\bar{p}_k(t)$ where K terms are governed by ODEs of first order

$$p(t) = \sum_{k=0}^{K} \bar{p}_k(t), \tag{D.7}$$

$$\begin{aligned} \bar{p}_0(t) &= \bar{Y}\tilde{J}(t) \\ &= \frac{Y^\infty}{1-\sum_{k=1}^{K}\bar{g}_k^P}(J(t)-1), \end{aligned} \tag{D.8}$$

$$\begin{aligned} \bar{p}_k(t) + \tau_k \dot{\bar{p}}_k(t) &= -\bar{g}_k^P \bar{Y}\tilde{J}(t) \\ &= -\bar{g}_k^P \frac{Y^\infty}{1-\sum_{k=1}^{K}\bar{g}_k^P}(J(t)-1), \quad k = 1..K. \end{aligned} \tag{D.9}$$

The set of equations (D.7) - (D.9) replaces evaluation of Eq.(D.1). Since the order K of the Prony Series is low in an average application, the computational effort to solve Eq. (D.7) - (D.9) is much smaller than solving the time convolution of Eq.(D.1).

Bibliography

[1] *Abaqus User's Manual Version 6.4*. Abaqus, Inc., 2003.

[2] D. Adalsteinsson and J. A. Sethian. A fast level set method for propagating interfaces. *J. Comput. Phys.*, 118(2):269–277, 1995.

[3] D. Adalsteinsson and J. A. Sethian. A level set approach to a unified model for etching, deposition, and lithography I: algorithms and 2-dimensional simulations. *J. Comput. Phys.*, 120(1):128–144, 1995.

[4] D. Adalsteinsson and J. A. Sethian. A level set approach to a unified model for etching, deposition, and lithography II: 3-dimensional simulations. *J. Comput. Phys.*, 122(2):348–366, 1995.

[5] M. F. Adams, H. H. Bayraktar, T. M. Keaveny, and P. Papadopoulos. Ultrascalable implicit finite element analyses in solid mechanics with over half a billion degrees of freedom. In *Proceedings of SC2003: High Performance Networking and Computing*, page Gordon Bell Award paper. ACM/IEEE, 2004.

[6] V. Alexiades, G. Amiez, and P.-A. Gremaud. Super-time-stepping acceleration of explicit schemes for parabolic problems. *Commun. Numer. Meth. Engng.*, 12(1):31–42, 1996.

[7] C. Alexiou, M. Galogavrou, Q. Chen, A. McDonald, A. P. Salmon, B. K. Keeton, M. P. Haw, and J. L. Monro. Mitral valve replacement with mechanical prostheses in children: improved operative risk and survival. *Eur. J. Cardiothorac Surgery*, 20:105–113, 2001.

[8] M. P. Allen and D. J. Tildesley. *Computer Simulation of Liquids*. Clarendon Press Oxford, Oxford, 1987.

[9] A. S. Almgren, J. B. Bell, P. Colella, L. H. Howell, and M. L. Welcome. A conservative adaptive projection method for the variable density incompressible Navier-Stokes equations. *J. Comput. Phys.*, 142:1–46, 1998.

[10] N. Ayache. Epidaure: A research project in medical image analysis, simulation and robotics at INRIA. *IEEE Transaction on Medical Imaging*, 2003. Invited Editorial.

[11] C. Basdogan, C.-H. Ho, and M. Srinivasan. Virtual environments in medical training: Graphical and haptic simulation of laparoscopic common bileduct exploration. *IEEE/ASME Transactions on Mechatronics*, 6(3):269–285, 2001.

[12] P. Bashook and J. Parboosingh. Continuing medical education: Recertification and the maintenance of competence. *British Medical Journal*, 316:545–548, 1998.

[13] D. Baur, Ch. Guzzoni and O. Georg. Virgy: A virtual reality and force feedback based endoscopy surgery simulator. In *Proc. MMVR'98*, pages 110–116, 1998.

[14] J. T. Beale. A convergent 3-D vortex method with grid-free stretching. *Math. Comput.*, 46:401–424, 1986.

[15] D. Beeman. Some multistep methods for the use in molecular dynamics calculations. *J. Comput. Phys.*, 20:130–139, 1976.

[16] J. B. Bell, P. Colella, and H. M. Glaz. A 2nd-order projection method for the incompressible Navier Stokes equations. *J. Comput. Phys.*, 85(2):257–283, 1989.

[17] G. Bentel. *Radiation Therapy Planning*. McGraw Hill, Inc., 1995.

[18] M. Bergdorf, G.-H. Cottet, and P. Koumoutsakos. Multilevel adaptive particle methods for convection-diffusion equations. *Multiscale Model. Simul.*, 4(1):328–357, 2005.

[19] M. Bergdorf and P. Koumoutsakos. A Lagrangian particle-wavelet method. *Multiscale Model. Simul.*, 5(3):980–995, 2006.

[20] J. Berkley, G. Turkiyyah, D. Berg, M. Ganter, and S. Weghorst. Real-time Finite Element modeling for surgery simulation: Application to virtual suturing. *IEEE Transactions on Visualization and Computer Graphics*, 10(3):1–12, 2004.

[21] D. Bielser, P. Glardon, M. Teschner, and M. Gross. A state machine for real-time cutting of tetrahedral meshes. In *Pacific Graphics 2003*, pages 377–386, 2003.

[22] M. Bro-Nielsen. Fast finite elements for surgery simulation. In *Medicine Meets Virtual Reality*, 1997.

[23] M. Bro-Nielsen, D. Helfrick, B. Glass, X. Zeng, and H. Connacher. Virtual reality simulation of abdominal trauma surgery. In *Medicine Meets Virtual Reality*, 1998.

[24] I. N. Bronstein and K. A. Semendyayev. *Handbook of Mathematics*. Van Nostrand Reinhold, 20 edition, 1991.

[25] S. Bryson and D. Levy. High-order central WENO schemes for multidimensional Hamilton-Jacobi equations. *SIAM J. Numer. Anal.*, 41(4):1339–1369, 2003.

[26] J. Carling, T. L. Williams, and G. Bowtell. Self-propelled anguilliform swimming: simultaneous solution of the two-dimensional Navier-Stokes equations and Newtons' laws of motion. *J. Exp. Biol.*, 201:3143–3166, 1998.

[27] E. A. Carmona and L. J. Chandler. On parallel PIC versatility and the structure of parallel PIC approaches. *Concurrency: Pract. Ex.*, 9(12):1377–1405, 1997.

[28] F. Carter, T. Frank, M. Davies, and A. Cuschieri. Measurement and modelling of the compliance of human and porcine organs. *Med. Image Anal.*, 5(4):231–236, 2001.

[29] A. K. Chaniotis, D. Poulikakos, and P. Koumoutsakos. Remeshed smoothed particle hydrodynamics for the simulation of viscous and heat conducting flows. *J. Comput. Phys.*, 182(1):67–90, 2002.

[30] S. R. Chapple and L. J. Clarke. The parallel utilities library. In *Proceedings of the IEEE Scalable Parallel Libraries Conference*, pages 21–30. IEEE, 1994.

[31] P. Chatelain, S. Hieber, P. Koumoutsakos, and G.-H. Cottet. P.M.H.: Particle-mesh hydrodynamics. In *Discrete Simulations of Fluid Dynamics*, 2006.

[32] H. Cheng, L. Greengard, and V. Rokhlin. A fast adaptive multipole algorithm in three dimensions. *J. Comput. Phys.*, 155:468–498, 1999.

[33] J. R. Cho and S. Y. Lee. Dynamic analysis of baffled fuel-storage tanks using the ALE finite element method. *Int. J. Numer. Meth. Fluids*, 41(2):185–208, 2003.

[34] D. L. Chopp. Computing minimal-surfaces via level set curvature flow. *J. Comput. Phys.*, 106(1):77–91, 1993.

[35] D. L. Chopp and J. A. Sethian. Flow under curvature: Singularity formation, minimal surfaces, and geodesics. *Exp. Mathematics*, 2(4):235–255, 1993.

[36] A. J. Chorin. Numerical study of slightly viscous flow. *J. Fluid Mech.*, 57(4):785–796, 1973.

[37] J. Cohen, M. Lin, D. Manocha, and M. Ponamgi. I-COLLIDE: An interactive and exact collision detection system for large-scale environments. In *Symposium on Interactive 3D Graphics*, pages 189–196, 1995.

[38] F. Colin, R. Egli, and F. Lin. Computing a null divergence velocity field using smoothed particle hydrodynamics. *J. Comput. Phys.*, 217(2):680–692, 2006.

[39] G. Cottet. A particle model for fluid-structure interaction. *C.R. Acad. Sci. Paris*, Ser. I(335):833–838, 2002.

[40] G.-H. Cottet. Artificial viscosity models for vortex and particle methods. *J. Comput. Phys.*, 127:299–308, 1996.

[41] G.-H. Cottet and P. Koumoutsakos. *Vortex Methods – Theory and Practice*. Cambridge University Press, New York, 2000.

[42] G.-H. Cottet, P. Koumoutsakos, and M. L. O. Salihi. Vortex methods with spatially varying cores. *J. Comput. Phys.*, 162(1):164–185, 2000.

[43] G.-H. Cottet and E. Maitre. A level-set method for fluid-structure interactions with immersed surfaces. *Math. Model Meth. Appl. Sci.*, 16(3):415–438, 2006.

[44] G.-H. Cottet, B. Michaux, S. Ossia, and G. VanderLinden. A comparison of spectral and vortex methods in three-dimensional incompressible flows. *J. Comput. Phys.*, 175:702–712, 2002.

[45] G.-H. Cottet and P. Poncet. Advances in direct numerical simulation of 3D wall-bounded flows by vortex-in-cell methods. *J. Comput. Phys.*, 193:136–158, 2003.

[46] R. Couturier and C. Chipot. Parallel molecular dynamics using OpenMP on a shared memory machine. *Comp. Phys. Commun.*, 124:49–59, 2000.

[47] S. J. Cummins and M. Rudman. An SPH projection method. *J. Comput. Phys.*, 152:584–607, 1999.

[48] G. Debunne, M. Desbrun, M.-P. Cani, and A. Barr. Dynamic real-time deformations using space and timing adaptive sampling. In E. Fiume, editor, *Conference Proceedings SIGGRAPH*, pages 31–36. ACM Press / ACM SIGGRAPH, 2001.

[49] P. Degond and S. Mas-Gallic. The weighted particle method for convection-diffusion equations. part 1: The case of an isotropic viscosity. *Math. Comput.*, 53(188):485–507, 1989. Oct.

[50] P. Degond and S. Mas-Gallic. The weighted particle method for convection-diffusion equations. part 2: The anisotropic case. *Math. Comput.*, 53(188):509–525, 1989. Oct.

[51] G. L. Djordjević and M. B. Tošić. A heuristic for scheduling task graphs with communication delays onto multiprocessors. *Parallel Computing*, 22:1197–1214, 1996.

[52] M. Doblare, E. Cueto, B. Calvo, M. Martinez, J. Garcia, and J. Cegonino. On the employ of meshless methods in biomechanics. *Comp. Meth. Appl. Mech. & Engng.*, 194:801–821, 2005.

[53] D. Durand, R. Jain, and D. Tseytlin. Parallel I/O scheduling using randomized, distributed edge coloring algorithms. *J. Parallel Distrib. Comput.*, 63:611–618, 2003.

[54] J. D. Eldredge, A. Leonard, and T. Colonius. A general determistic treatment of derivatives in particle methods. *J. Comput. Phys.*, 180(2):686–709, 2002.

[55] M. Ellero, M. Kröger, and S. Hess. Viscoelastic flows studied by smoothed particle dynamics. *J. Non-Newtonian Fluid Mech.*, 105(1):35–51, 2002.

[56] B. Engquist, A.-K. Tornberg, and R. Tsai. Discretization of dirac delta functions in level set methods. *J. Comput. Phys.*, 207(1), 2005.

[57] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell. A hybrid particle level set method for improved interface capturing. *J. Comput. Phys.*, 183(1):83–116, 2002.

[58] P. Espanol. Hydrodynamics from dissipative particle dynamics. *Phys. Rev. E*, 52(2):1734–1742, 1995.

[59] E. A. Fadlun, R. Verzicco, P. Orlandi, and J. Mohd-Yusof. Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations. *J. Comput. Phys.*, 161(1):35–60, 2000.

[60] D. L. Fishman, M. B. Leon, D. S. Baim, R. A. Schatz, M. P. Savage, I. Penn, K. Detre, L. Veltri, D. Ricci, M. Nobuyoshi, and M. Cleman. A randomized comparison of coronary-stent placement and balloon angioplasty in the treatment of coronary artery disease. *The New England Journal of Medicine*, 331:496–501, 1994.

[61] B. Fornberg. Steady viscous flow past a sphere at high Reynolds numbers. *J. Fluid Mech.*, 190:471–489, 1988.

[62] Y. C. Fung. *Biomechanics. Mechanical Properties of Living Tissues*. Springer-Verlag, 2nd edition, 1993.

[63] A. Ghoniem and O. Knio. The development and application of the transport element method to three dimensional reacting shear layers. In *Vortex Dynamics and Vortex Methods*, pages 165–218. American Mathematical Society, 1991.

[64] F. Gibou, R. Fedkiw, R. Caflisch, and S. Osher. A level set approach for the numerical simulation of dendritic growth. *J. Sci. Comput.*, 19(1-3):183–199, 2003.

[65] R. A. Gingold and J. J. Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Month Notices Roy. Astron. Soc.*, 181:375–389, 1977.

[66] S. Gottschalk, M. Lin, and D. Manocha. OBBTree: A hierarchical structure for rapid interference detection. *Comput. Graphics*, 30:171–180, 1996.

[67] J. P. Gray, J. J. Monaghan, and R. P. Swift. SPH elastic dynamics. *Comp. Meth. Appl. Mech. & Engng.*, 190:6641–6662, 2001.

[68] M. Grayson. The heat equation shrinks embedded plane curves to round points. *J. Diff. Geom.*, 26:555–558, 1987.

[69] M. Grayson. A short note on the evolution of surfaces via mean curvatures. *J. Diff. Geom.*, 58:285–314, 1989.

[70] L. Greengard and V. Rokhlin. The rapid evaluation of potential fields in three dimensions. *Lect. Notes Math.*, 1360:121–141, 1988.

[71] P. M. Gresho and R. L. Sani. On pressure boundary conditions for the incompressible navier-stokes equations. *Int. J. Numer. Meth. Fluids*, 7:1111–1145, 1987.

[72] J. Guilkey, J. Hoying, and J. Weiss. Computational modeling of multicellular constructs with the material point method. *Journal of Biomechanics*, 39(11):2079–2086, 2006.

[73] J. K. Hahn, R. Kaufman, A. B. Winick, T. Carleton, Y. Park, K.-M. Oh, R. Lindeman, N. Al-Ghreimil, R. Walsh, M. Loew, J. Gerber, and S. Sankar. Training environment for inferior vena caval filter placement. In *Proc. MMVR'98*, pages 291–297, 1998.

[74] J. M. Haile. *Molecular Dynamics Simulations. Elementary Methods*. John Wiley & Sons, 1992.

[75] O. H. Hald. Convergence of vortex methods for Euler's equations, III. *SIAM J. Numer. Anal.*, 24(3):538–582, 1987.

[76] M. Harders, M. Bajka, U. Spaelter, S. Tuchschmid, H. Bleuler, and G. Szekely. Highly-realistic, immersive training environment for hysteroscopy. In *Medicine Meets Virtual Reality*, 2006.

[77] F. H. Harlow. Particle-in-cell computing method for fluid dynamics. *Methods Comput. Phys.*, 3:319–343, 1964.

[78] B. Heidelberger, M. Teschner, T. Frauenfelder, and M. Gross. Collision handling of deformable anatomical models for real-time surgery simulation. *J. Technology and Health Care*, 12(3):235–243, 2004.

[79] B. Heidelberger, M. Teschner, and M. Gross. Real-time volumetric intersections of deforming objects. In *Vision, Modeling and Visualization*, pages 461–468. IOS Press, 2003.

[80] R. D. Henderson. Details of the drag curve near the onset of vortex shedding. *Phys. Fluids*, 7:2102–2104, 1995.

[81] S. E. Hieber and P. Koumoutsakos. A Lagrangian particle level set method. *J. Comput. Phys.*, 210:342–367, 2005.

[82] S. E. Hieber, J. H. Walther, and P. Koumoutsakos. Remeshed smoothed particle hydrodynamics simulation of the mechanical behavior of human organs. *J. Technology and Health Care*, 12(4):305–314, 2004.

[83] C. W. Hirt and B. D. Nichols. Volume of fluid (Vof) method for the dynamics of free boundaries. *J. Comput. Phys.*, 39(1):201–225, 1981.

[84] R. W. Hockney and J. W. Eastwood. *Computer Simulation Using Particles*. Institute of Physics Publishing, Bristol, PA, USA, 2 edition, 1988.

[85] G. Holzapfel. *Nonlinear Solid Mechanics*. John Wiley & Sons, 2000.

[86] J. Jakimowcz. The europaen association for endoscopic surgery: Recommendations for training in laparoscopic surgery. *Annales Chirurgiae et Gynaecologiae*, pages 137–141, 1994.

[87] H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and A.-L. Barabási. The large-scale organization of metabolic networks. *Nature*, 407:651–654, 2000.

[88] A. T. Johnson and V. C. Patel. Flow past a sphere up to a Reynolds number of 300. *J. Fluid Mech.*, 378:19–70, 1999.

[89] G. Johnson and S. Beissel. Normalized smoothing functions for SPH impact computations. *Int. J. for Numer. Methods In Engng.*, 16:2725–2741, 1997.

[90] G. R. Johnson, R. A. Stryk, and S. R. Beissel. SPH for high velocity impact computations. *Comp. Meth. Appl. Mech. & Engng.*, 139:347–373, 1996.

[91] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 20(1):359–392, 1998.

[92] S. Kern and P. Koumoutsakos. Simulations of optimized anguilliform swimming. *J. Exp. Biol.*, 209(24):4841–4857, 2006.

[93] J. Kim, D. Kim, and H. Choi. An immersed-boundary finite-volume method for simulations of flow in complex geometries. *J. Comput. Phys.*, 171:132–150, 2001.

[94] S. Koshizuka and Y. Oka. Moving-particle semi-implicit method for fragmentation of incompressible fluid. *Nuclear Sc. Eng.*, 123(3):421–434, 1996.

[95] S. Koshizuka, H. Tamako, and Y. Oka. Numerical analysis of breaking waves using the moving particle semi-implicit method. *Int. J. Numer. Meth. Fluids*, 26:751–769, 1998.

[96] P. Koumoutsakos. Vorticity flux control in a turbulent channel flow. *Phys. Fluids*, 11(2):248–250, 1999.

[97] P. Koumoutsakos. Multiscale flow simulations using particles. *Annu. Rev. Fluid Mech.*, 37:457–487, 2005.

[98] P. Koumoutsakos and A. Leonard. High-resolution simulation of the flow around an impulsively started cylinder using vortex methods. *J. Fluid Mech.*, 296:1–38, 1995.

[99] P. Koumoutsakos, A. Leonard, and F. Pépin. Boundary conditions for viscous vortex methods. *J. Comput. Phys.*, 113(1):52–61, 1994.

[100] U. Kuehnapfel, H. Krumm, C. Kuhn, M. Huebner, and B. Neisius. Endosurgery simulations with KISMET: A flexible tool for surgical instrument design, operation

room planning and VR technology based abdominal surgery training. In *Proc. Virtual reality World95*, pages 165–171, 1995.

[101] R. J. LeVeque. High-resolution conservative algorithms for advection in incompressible flow. *SIAM J. Numer. Anal.*, 33(2):627–665, 1996.

[102] A. Liu, F. Tendick, K. Cleary, and C. Kaufmann. A survey of surgical simulation: Applications, technology, and education. *Presence*, 12(6):599–614, 2003.

[103] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Comput. Graphics*, 21(4):163–169, 1987.

[104] E. Mazza, A. Nava, M. Bauer, R. Winter, M. Bajka, and G. Holzapfel. Mechanical properties of the human uterine cervix: An in-vivo study. *Med. Image Anal.*, 10:125–136, 2006.

[105] E. Molinari, M. Fato, G. DeLeo, D. Riccardo, and F. Beltrame. Simulation of the biomechanical behavior of the skin in virtual surgical applications by finite element method. *IEEE Transaction on Biomedical Engineering*, 52(9):1514–1521, 2005.

[106] J. J. Monaghan. Extrapolating B splines for interpolation. *J. Comput. Phys.*, 60(2):253–262, 1985.

[107] J. J. Monaghan. Smoothed particle hydrodynamics. *Annu. Rev. Astron. Astrophys.*, 30:543–574, 1992.

[108] J. J. Monaghan. SPH without a tensile instability. *J. Comput. Phys.*, 159:290–311, 2000.

[109] J. J. Monaghan. Smoothed particle hydrodynamics. *J. Comput. Phys.*, 68(8):1703–1759, 2005.

[110] K. Montgomery, C. Bruyns, J. Brown, G. Thonier, A. Tellier, and J.-C. Latombe. Spring: A general framework for collaborative, real-time surgical simulation. In *Medicine Meets Virtual Reality*, pages 296–303, 2002.

[111] B. Moon and J. Saltz. Adaptive runtime support for direct simulation Monte Carlo methods on distributed memory architectures. In *Proceedings of the IEEE Scalable High-Performance Computing Conference*, pages 176–183. IEEE, 1994.

[112] D. W. Moore. The effect of compressibility on the speed of propagation of a vortex ring. *Proc. R. Soc. Lond. A*, 397(1812):87–97, 1985.

[113] J. P. Morris. Simulating surface tension with smoothed particle hydrodynamics. *Int. J. Numer. Meth. Fluids*, 33(3):333–353, 2000.

[114] National Library of Medicine. The Visible Human Project. `http://www.nlm.nih.gov/research/visible`.

[115] A. Nava, E. Mazza, F. Kleinermann, N. Avis, and J. McClure. Determination of the mechanical properties of soft human tissue through aspiration experiments. *Lect. Notes Comput. Sc.*, 1878:222–229, 2003.

[116] A. Nava, E. Mazza, F. Kleinermann, N. Avis, J. McClure, and M. Bajka. Evaluation of the mechanical properties of human liver and kidney through aspiration experiments. *J. Technology and Health Care*, 12:269–280, 2004.

[117] C. Nieter and J. R. Cary. VORPAL: a versatile plasma simulation code. *J. Comput. Phys.*, 196(2):448–473, 2004.

[118] S. Osher and R. Fedkiw. *The Level Set Method and Dynamics Implicit Surfaces*. Springer-Verlag, New York, 2002.

[119] S. Osher and R. P. Fedkiw. Level set methods: An overview and some recent results. *J. Comput. Phys.*, 169(2):463–502, 2001.

[120] S. Osher and J. A. Sethian. Front propagating with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulation. *J. Comput. Phys.*, 79(1):12–49, 1988.

[121] M. Ottensmeyer and J. Salisbury. In vivo data acquisition instrument for solid organ mechanical property measurement. In *MICCAI'01*, pages 975–982, 2001.

[122] R. L. Panton. *Incompressible Flow*. John Wiley & Sons, 2 edition, 1996.

[123] J. Park, K. Kwon, and H. Choi. Numerical solutions of flow past a cylinder at Reynolds number up to 160. *KSME Int. J.*, 12(6):1200–1205, 1998.

[124] C. Peskin. Flow patterns around heart valves: A numerical study. *J. Comput. Phys.*, 10:252–271, 1972.

[125] R. Peyret and T. Taylor. *Computational Methods for Fluid Flow*. Springer-Verlag, 1983.

[126] B. Pflesser, A. Petersik, U. Tiede, K. Hoehne, and R. Leuwer. Volume cutting for virtual petrous bone surgery. *Computer Aided Surgery*, 7(2):74–83, 2002.

[127] S. J. Plimpton, M. F. Seidel, David B. Pasik, R. S. Coats, and G. R. Montry. A load-balancing algorithm for a parallel electromagnetic particle-in-cell code. *Comp. Phys. Commun.*, 152:227–241, 2003.

[128] P. Ploumhans, G. S. Winckelmans, J. K. Salmon, A. Leonard, and M. S. Warren. Vortex methods for direct numerical simulation of three-dimensional bluff body flows: Applications to the sphere at $Re = 300$, 500 and 1000. *J. Comput. Phys.*, 178:427–463, 2002.

[129] T. Rabczuk and J. Eibl. Simulation of high velocity concrete fragmentation using SPH/MLSPH. *Int. J. for Numer. Methods In Engng.*, 56:1421–1444, 2003.

[130] P. W. Randles and L. D. Libersky. Smoothed particle hydrodynamics: Some recent improvements and applications. *Comp. Meth. Appl. Mech. & Engng.*, 139:375–408, 1996.

[131] P. W. Randles and L. D. Libersky. Normalized SPH with stress points. *Int. J. for Numer. Methods In Engng.*, 48:1445–1462, 2000.

[132] J. Reddy. *Finite Element Method*. McGraw Hill, Inc., 2nd edition edition, 1993.

[133] M. Reznek, C. Rawn, and T. Krummel. Evaluation of the educational effectiveness of a virtual reality intravenous insertion simulator. *Academic Emergency Medicine*, 9(11):1319–1325, 2002.

[134] W. J. Rider and D. B. Kothe. Reconstructing volume tracking. *J. Comput. Phys.*, 141:112–152, 1998.

[135] L. Rosenhead. The formation of vortices from a surface of discontinuity. *Proc. R. Soc. Lond. A*, 134:170–192, 1931.

[136] A. Roshko. Experiments on the flow past a circular cylinder at very high Reynolds number. *J. Fluid Mech.*, 10(3):345–356, 1961.

[137] P. G. Saffman. *Vortex Dynamics*. Cambridge University Press, 1992.

[138] M. Sagar, D. Bullivant, G. Mallinson, and P. Hunter. A virtual environment and model of the eye for surgical simulation. *Computer Graphics and Interactive Techniques*, pages 205–212, 1994.

[139] R. Satava. Virtual reality surgical simulator: The first steps. *Surgical Endoscopy*, 7:203–205, 1993.

[140] I. F. Sbalzarini, J. H. Walther, M. Bergdorf, S. E. Hieber, E. M. Kotsalis, and P. Koumoutsakos. PPM – a highly efficient parallel particle-mesh library for the simulation of continuum systems. *J. Comput. Phys.*, 215:566–588, 2006.

[141] R. Scardovelli and S. Zaleski. Direct numerical simulation of free-surface and interfacial flow. *Annu. Rev. Fluid Mech.*, 31:567–603, 1999.

[142] M. Schill, C. Wagner, M. Hennen, H. Bender, and R. Maenner. Eyesi - A simulator for intra-ocular surgery. In *Proc. MICCAI'99*, pages 1166–1174, 1999.

[143] J. Sethian. Numerical algorithms for propagating interfaces: Hamilton-Jacobi equations and conservations laws. *J. Diff. Geom.*, 31:131–161, 1990.

[144] J. Sethian and J. Strain. Crystal growth and dendritic solidification. *J. Comput. Phys.*, 98:231–253, 1992.

[145] J. A. Sethian. A fast marching level set method for monotonically advancing fronts. *Proc. Natl. Acad. Sci. USA*, 93(4):1591–1595, 1996.

[146] J. A. Sethian. Fast marching methods. *SIAM Rev.*, 41(2):199–235, 1999.

[147] J. A. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, Cambridge, UK, 1999.

[148] J. A. Sethian. Evolution, implementation, and application of level set and fast marching methods for advancing fronts. *J. Comput. Phys.*, 169(2):503–555, 2001.

[149] J. A. Sethian and P. Smereka. Level set methods for fluid interfaces. *Annu. Rev. Fluid Mech.*, 35:341–372, 2003.

[150] K. Shariff, R. Verzicco, and P. Orlandi. A numerical study of three-dimensional vortex ring instabilities: viscous corrections and early nonlinear stage. *J. Fluid Mech.*, 279:351–375, 1994.

[151] C.-W. Shu and S. Osher. Efficient implementation of essentially nonoscillatory shock-capturing schemes. *J. Comput. Phys.*, 77(2):439–471, 1988.

[152] A. L. F. Silva, A. Silveira-Neto, and J. J. R. Damasceno. Numerical simulations of two-dimensional flows over a circular cylinder using the immersed boundary method. *J. Comput. Phys.*, 189:351–370, 2003.

[153] S. P. Singh and S. Mittal. Flow past a cylinder: shear layer instability and drag crisis. *J. Comput. Phys.*, 47:75–98, 2005.

[154] M. Souli, A. Ouahsine, and L. Lewin. ALE formulation for fluid-structure interaction problems. *Comp. Meth. Appl. Mech. & Engng.*, 190(5-7):659–675, 2000.

[155] V. Springel, N. Yoshida, and S. D. M. White. GADGET: a code for collisionless and gasdynamical cosmological simulations. *New Astronomy*, 6:79–117, 2001.

[156] J. Strain. A fast semi-Lagrangian contouring method for moving interfaces. *J. Comput. Phys.*, 161(2):512–536, 2001.

[157] M. Sussman and E. Fatemi. An efficient, interface-preserving level set redistancing algorithm and its application to interfacial incompressible fluid flow. *SIAM J. Sci. Comput.*, 20(4):1165–1191, 1999.

[158] M. Sussman, P. Smereka, and S. Osher. A level set approach for computing solutions to incompressible 2-phase flow. *J. Comput. Phys.*, 114(1):146–159, 1994.

[159] J. W. Swegle, D. L. Hicks, and S. W. Attaway. Smoothed particle hydrodynamics stability analysis. *J. Comput. Phys.*, 116(1):123–134, 1995.

[160] G. Szekely, M. Bajka, C. Brechbuhler, J. Dual, R. Enzeler, U. Haller, J. Hug, R. Hutter, N. Ionmenger, M. Kauer, P. Meier, P. Niederer, and G. Trister. Virtual reality based simulation of endoscopic surgery. *Presence*, 9(3):310–333, 2000.

[161] G. Szekely and R. Satava. Virtual reality in medicine. *British Medical Journal*, 319:1305–1315, 1999.

[162] H. Takeda, S. M. Miyama, and M. Sekiya. Numerical simulation of viscous flow by smoothed particle hydrodynamics. *Prog. Theor. Phys.*, 92(5):939–960, 1994.

[163] H.-Z. Tang, T. Tang, and P. Zhang. An adaptive mesh redistibution method for nonlinear Hamiltion- Jacobi equations in two- and three-dimensions. *J. Comput. Phys.*, 188:543–572, 2003.

[164] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. In *ACM SIGGRAPH Computer Graphics*, pages 205–214, 1987.

[165] D. Terzopoulous and K. Fleischer. Deformable models. *The Visual Computer*, 4:306–331, 1988.

[166] M. Teschner, S. Girod, and B. Girod. Direct computation of nonlinear soft-tissue deformation. In *Proc. Vision, Modeling, Visualization*, pages 383–390, 2000.

[167] M. Teschner, B. Heidelberger, M. Muller, and M. Gross. A versatile and robust model for geometrically complex deformable solids. In *Proceedings of Computer Graphics International*, pages 312–319, 2004.

[168] M. Teschner and M. Mueller. Volumetric meshes for real-time medical simulations. In *Bildverarbeitung fr die Medizin*, pages 279–283, 2003.

[169] U. Trottenberg, C. Oosterlee, and A. Schueller. *Multigrid*. Academic Press, San Diego, 2001.

[170] U. U. Kuehnapfel, H. Cakmak, and H. Maass. Endoscopic surgery training using virtual reality and deformable tissue simulation. *Computer & Graphics*, 24(5):671–682, 2000.

[171] D. Valtorta and E. Mazza. Dynamic measurements of soft tissue viscoelastic properties with a torsional resonator device. *Med. Image Anal.*, 9(5):481–490, 2005.

[172] L. Verlet. Computer experiments on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules. *Phys. Rev.*, 159(1):98–103, 1967.

[173] K. Verstreken, J. Van Cleynenbreugel, K. Martens, and et al. An image-guided planning system for endosseous oral implants. *IEEE Transactions on Medical Imaging*, 17:842–852, 1998.

[174] D. Vining. Virtual endoscopy: Is it reality. *Radiology*, pages 30–31, 1996.

[175] V. G. Vizing. On an estimate of the chromatic class of a p-graph. *Diskret. Anal.*, 3:25–30, 1964. in Russian.

[176] R. Wachter. *Evidence Report/Technology Assessment, No. 43, Making Health Care Safer: A Critical Analysis of Patient Safety Practices*. Agency for Healthcare Research and Quality, 2001.

[177] J. H. Walther and P. Koumoutsakos. Three-dimensional particle methods for particle laden flows with two-way coupling. *J. Comput. Phys.*, 167:39–71, 2001.

[178] Q. Wang. Variable order revised binary treecode. *J. Comput. Phys.*, 200:192–210, 2004.

[179] R. Webster, D. Zimmerman, B. Mohler, M. Melkonian, and R. Haluck. A prototype haptic suturing simulator. In J. Westwood and et al., editors, *Medicine Meets Virtual Reality*, pages 567–569, 2001.

[180] D. C. Wilcox. *Basic Fluid Mechanics*. DCW Industries, second edition edition, 2000.

[181] C. H. K. Williamson. Oblique and parallel modes of vortex shedding in the wake of a circular cylinder at low Reynolds numbers. *J. Fluid Mech.*, 206:579–627, 1989.

[182] J. H. Williamson. Low-storage Runge-Kutta schemes. *J. Comput. Phys.*, 35:48–56, 1980.

[183] WordNetSearch. `http://www.cogsci.princeton/cgi-win/webbwn`, 2006.

[184] B. Wu and X. Du. Finite element formulation of radial tires with variable constraint conditions. *Computers & Structures*, 55(5):871–875, 1993.

[185] X. Wu, M. C. Vargas, S. Nayak, V. Lotrich, and G. Scoles. Towards extending the applicability of density functional theory to weakly bound systems. *J. Chem. Phys.*, 115(19):8748–8757, 2001.

[186] H. Y. Yoon, S. Koshizuka, and Y. Oka. Particle-gridless hybrid method for incompressible flows. *Int. J. Numer. Meth. Fluids*, 30:407–424, 1999.

[187] N. Zabusky, M. Hughes, and K. Roberts. Contour dynamics for the euler equations in two dimensions. *J. Comput. Phys.*, 30:96–106, 1979.

[188] S. T. Zalesak. Fully multidimensional flux-corrected transport algorithms for fluids. *J. Comput. Phys.*, 31(3):335–362, 1979.

[189] D. Zeng, A. Ferrari, J. Ulmer, A. Veligodskiy, P. Fischer, J. Spatz, Y. Ventikos, D. Poulikakos, and R. Kroschewki. Three-dimensional modeling of mechanical forces in the extracellular matrix during epithelial lumen formation. 90:4380–4391, 2006.

[190] P. Zerfass, Z. Krol, B. von Rymon-Lipinski, T. Jansen, and E. K. Keeve. Towards a virtual environment for biomechanical simulation. In *Proceedings Computer Assisted Radiology and Surgery CARS'01, Berlin, Germany*, 2001.

[191] R. Ziegler, W. Müller, G. Fischer, and M. Goebel. A virtual reality medical training system. In *Proc. first In. Conf. on Comp. Vision, Virtual Reality and Robotics in Medicine, CVRMed'95*, pages 282–286, 1995.

[192] S. Zimmermann, P. Koumoutsakos, and W. Kinzelbach. Simulation of pollutant transport using a particle method. *J. Comput. Phys.*, 173:322–347, 2001.

Own Publications

Parts of this thesis have been published in the following papers

Refereed Journal Papers

1. S. E. Hieber, J. H. Walther, and P. Koumoutsakos. Remeshed smoothed particle hydrodynamics simulation of the mechanical behavior of human organs. *J. Technology and Health Care*, 12(4):305–314, 2004.

2. Simone Elke Hieber and Petros Koumoutsakos. A Lagrangian particle level set method. *J. Comput. Phys.*, 210:342–367, 2005.

3. I. F. Sbalzarini, J. H. Walther, M. Bergdorf, S. E. Hieber, E. M. Kotsalis, and P. Koumoutsakos. PPM – a highly efficient parallel particle-mesh library for the simulation of continuum systems. *J. Comput. Phys.*, 215:566–588, 2006.

Refereed Conference Papers

1. I. F. Sbalzarini, J. H. Walther, B. Polasek, P. Chatelain, M. Bergdorf, S. E. Hieber, E. M. Kotsalis, and P. Koumoutsakos. A software framework for portable parallelization of particle-mesh simulations. *Lect. Notes Comput. Sc.*, 4128:730–739, 2006.

Curriculum Vitae

Name	Simone Elke Hieber
Citizen of	Germany
Born	February 19^{th}, 1976, Geislingen/Steige, Germany
July 1995	Abitur (High school leaving certificate)
	Geislingen/Steige, Germany
1995 - 2001	University of Stuttgart, Germany
	Major: Engineering Cybernetics
1999 - 2001	Michigan Technological University, Houghton, MI, USA
	Major: Applied Mathematics
May 2001	Master of Science in Mathematics
	Michigan Technological University, Houghton, MI, USA
Oct 2001	Diploma in Engineering Cybernetics
	University of Stuttgart, Germany
2002 - 2006	Swiss Federal Institute of Technology, ETH Zurich
	Research and teaching assistant
	Institute of Computational Science (ICoS)