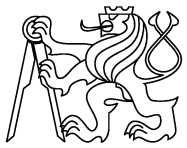
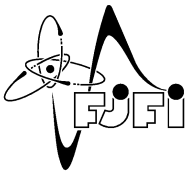


Czech Technical University in Prague
Faculty of Nuclear Sciences and Physical Engineering
Department of Physical Electronics



Arbitrary Lagrangian-Eulerian (ALE) Methods in Plasma Physics

Doctoral Thesis

Author: Ing. Milan Kuchařík
Supervisor: Doc. Ing. Richard Liska, CSc.
Adviser: Doc. Ing. Jiří Limpouch, CSc.
Date: April 13, 2006

Abstract

The complete Arbitrary Lagrangian-Eulerian (ALE) method, applicable to the fluid and laser plasma hydrodynamics, both in Cartesian and cylindrical geometries is presented. All parts of the ALE algorithm, i.e. Lagrangian solver, mesh smoothing, and conservative quantity remapping, in both geometries are fully described. The issues related to the laser plasma physics as plasma equation of state, thermal conductivity, interaction with laser beam, and sophisticated treatment of the boundary conditions necessary for the realistic laser plasma simulations are described. The complete developed ALE code is tested on a series of typical fluid problems to show its properties for the well known solutions. Finally, three sets of laser plasma simulations inspired by the real experiments are performed – the interaction of a laser beam with a massive target, ablative acceleration of small Aluminum disc flyer irradiated by a laser beam, and the high velocity impact of such accelerated disc onto a massive Aluminum target. The standard Lagrangian simulation of the last, high velocity impact problem fails, and the complete ALE methodology is required for this problem. Simulations of all types of problems show reasonable agreement with the experimental results.

Declaration

I declare that this thesis is my own work, based on my personal study and research. Information derived from the published and unpublished work of others has been acknowledged in the text, and a list of references is given.

Milan Kuchařík
Prague, April 13, 2006

Contents

Chapter 1

Introduction

The first note about the numerical solving of partial differential equations appeared in 1917, when the British scientist L. F. Richardson introduced his first talk about weather forecast using this technique, done by hand, and published in 1922 [81]. In general, this moment (although unsuccessful due to the CFL problem) is acknowledged as the beginning of the scientific discipline, which is nowadays called the Computational Fluid Dynamics (CFD). From the time of Richardson, not only the technical equipment allowing us to solve much more extensive and complex problems, but also theoretical knowledge essential for the computations, have evolved. Today, we know lots of mathematical theorems describing stability and accuracy of numerical methods, we know about the necessity of their conservativity and advantages of their symmetry. A wide spectrum of difference schemes based on the finite difference methods is used for numerical simulations. The numerical solving of the partial differential equations (PDEs) is today used not only in such traditional branches, as the weather forecast or oceanography. Connected with the technical development, many engineering and industrial applications, where the numerical modeling can be successfully used, came to view. On the other hand, the science provides many problems, where numerical simulations is the only approach to study them.

Although the field of numerical simulations is very large, our interest focuses on the simulations of the behavior of the compressible fluid, symbolized mathematically by a system of partial differential equations of hyperbolic type. Solution of such equations is represented as a density distribution of the conservative quantities in the computational domain, in the particular time of the simulation. In the case of compressible fluid dynamics, it is usual to neglect the fluid viscosity and the full set of the Navier-Stokes equations reduces to the form of system called the Euler equations. Here, the conservative quantities become the total mass, the total energy, and the components of the total momentum in each direction. In the compressible fluid dynamics, the phenomena of shock waves can appear. Shock wave is a severe change in the fluid density, velocity, pressure, and internal energy in a small distance in space and time. The shock waves have the form of discontinuity in the solutions and can cause serious troubles to the numerical methods – oscillations or diffusion can appear around the shock, when numerical treatment is not adequate. In reality, the shock waves thickness is proportional to the free mean path of the fluid particles and no real discontinuity exists. The numerical methods have to deal with such phenomena, and calculate the solution of the Euler equations close to the realistic fluid behavior.

For plasma simulations, two approaches are possible – the kinetic and the fluid models. In the kinetic model, the complete Boltzmann equation is solved either by a direct or statistical method (such as Particle in Cell or Monte Carlo method), and the solution has the form of the velocity distribution of electrons and ions. Kinetic approach may provide more complete information and use less assumptions than fluid approach, but it is computationally more expensive. Fluid approximation assumes that the scale lengths of intense quantities are large compared with electron and ion mean free paths. The temporal variations of these quantities are assumed slow compared with collision frequency. For these conditions, the difference of the particle distribution from Maxwellians is negligible. As electron ion relaxation is much slower than distribution relaxation to Maxwellian, electron and ion temperatures

often differ in the hot, low density corona. However, the difference is negligible in dense areas and thus we limit ourselves to one temperature approximation. Generally, these conditions can be violated in special cases, such as in extremely low pressure rarefied gas flows, where the fluid approach cannot be applied. On the other hand, the computational time of fluid simulations is reasonable also for large-scale simulations (when compared with the kinetic simulations), and allows simulations in higher dimensions. The first goal of this thesis is the development of an efficient, linearity and local-bound preserving remapping algorithm for recomputation of the conservative quantities between similar computational meshes, and its generalization to the cylindrical geometry. For future use, this algorithm must be generalizable to 3D and to meshes with changing topology. The next goal is the development of an efficient and reasonably accurate method for the simulations of laser-matter interaction and high velocity impacts, based on the fluid Arbitrary Lagrangian-Eulerian (ALE) methodology. This method have to treat severe computational mesh motion and produce results comparable with the experimental data. The method must be able to reproduce the experiments in both Cartesian and cylindrical geometries, and testing of the method on a set of experiment-inspired simulations is required. This thesis describes the construction of such method, and analyzes its properties. The last main goal of this thesis is the implementation of the presented ALE method, development of the computer code, and performing simulations of laser-matter and high velocity impact problems to model selected experiments.

In this chapter, we summarize the main contributions of this thesis to the field of ALE methods, and laser plasma simulations. The rest of the thesis is organized as follows. In the next chapter 2, we discuss two main approaches for solving the equations of compressible hydrodynamics – the Eulerian and the Lagrangian methods. These methods are suitable for different types of problems being solved. We summarize the main advantages and disadvantages of both approaches, and present the reasons for using their combinations – the Arbitrary Lagrangian-Eulerian (ALE) methods. We discuss the main types of the ALE methods, and briefly go through the history of the ALE methods. Several representative Eulerian, Lagrangian, and ALE codes for compressible fluid and plasma simulations are resumed.

In chapter 3, the complete Cartesian ALE method is presented. We go through all three parts of the ALE algorithm – Lagrangian solver, mesh smoothing techniques, and remapping. The compatible staggered Lagrangian method based on the evaluation of several types of forces affecting the nodal movement is reviewed, including several types of artificial viscosity formulation. Several both classical and modern methods for mesh smoothing and untangling are briefly summarized. Finally, the remapping method (the conservative interpolation of the conservative quantities) based on the approximate swept integration approach, is presented. This technique, which is the author mostly interested in, and which is his main contribution to the ALE methodology, is fully described and compared with the classical, computationally expensive method based on the exact integration. The swept integration remapping algorithm is also generalized to 3D, and to the computational meshes with changing connectivity.

In chapter 4, the ALE method is generalized into the cylindrical coordinates. It follows the previous chapter 3, and describes all points, where the cylindrical geometry changes the Cartesian formulas. The Area-Weighted Differencing (AWD) Lagrangian scheme is described and it is shown, that this approach cannot be used in our ALE formulation. Then, an alternative Control Volume (CV) Lagrangian method is presented, satisfying all needs for successful usage in the complete ALE algorithm. As for mesh smoothing techniques, no major changes according to the Cartesian geometry, are needed. The cylindrical formulas for our swept integration remapping method are derived in and easy-to-implement form. In the whole chapter 4, we emphasize the form of the formulas to be close to Cartesian ones, and thus an easy generalization of the Cartesian ALE code to the cylindrical geometry is possible.

In chapter 5, the main properties of the complete ALE method are commented. We discuss the conservativity of all parts of the ALE algorithm, and their linearity preservation, which in practice induces second order of accuracy. Satisfaction of these important properties is demonstrated for the typical tests of fluid dynamics.

In chapter 6, several physical aspects of the compressible fluid simulations are discussed, which arise when the ideal fluid is replaced by a laser-produced plasma. Properties of the quotidian equation of

state (QEOS) are briefly summarized, bringing our simulation closer to the real plasma behavior. The implementation of the thermal conductivity model with the classical Spitzer-Harm heat conductivity coefficient is briefly characterized, and its influence on different types of the laser plasma simulations is summarized. The process of laser beam absorption at the critical density is described, and we discuss the influence of the implementation of the laser plasma boundary conditions on the solution.
Chapter 7 includes numerical simulations of the laser-matter interactions, and their comparison with the experimental data. We present three approaches for expressing the maximal laser beam intensity from the experimental data. The code is applied to model several types of processes. At first, we perform simulations of the interaction of an intense laser beam with a massive Aluminum target, and compare the final crater formations with the craters obtained from experiments. Next, simulations of the laser beam irradiating and ablatively accelerating a thin Aluminum disc flyer are performed. And finally, simulations of the high velocity impact of this flyer on a massive target are shown. Two ways of construction of the initial data for the impact simulations are compared, average uniform values in the disc, and interpolated data from the disc acceleration simulations. All simulations are compared with the experimental results, and demonstrated on one particular example.

1.1 New Contributions of the Thesis

The main contributions of the thesis to the problematics of conservative interpolations, ALE methods, and computational laser plasma hydrodynamics, are listed below:

- Introducing the swept region remapping method in
  - 2D logically quadrilateral meshes in Cartesian and cylindrical geometries.
  - 3D general meshes.
  - 2D general meshes with changing connectivity.
- Introducing the complete remapping algorithm (for all state quantities) in cylindrical geometry.
- Development of the 2D ALE code on logically-orthogonal computational meshes working in both Cartesian and cylindrical geometries, applicable to the fluid and laser plasma simulations.
- Simulations of the small disc flyer ablative acceleration by an intense laser beam, and its impact to the massive target.

The given points are new, and can be useful for both the ALE and laser plasma societies. Parts of this thesis have been presented at the scientific conferences and published in the proceedings and in the specialized journals, for example “Journal of Computational Physics” [58], “Computers and Fluids” [36], or “Czechoslovak Journal of Physics” [55], other parts are submitted or in preparation for publication.

Chapter 2

Numerical Methods for Compressible Fluid Flows

In this chapter, we describe the main approaches for numerical solving of the compressible fluid Euler equations – the Eulerian and Lagrangian methods. We show the form of the system of the fluid Euler equations in both formulations and mention the basic methods for their solving, and selected codes implementing them. We also describe the ALE (Arbitrary Lagrangian-Eulerian) methods, the combination of both approaches. Finally, several codes for the plasma simulations are overviewed.

2.1 Eulerian Methods

The family of Eulerian methods solves the system of the compressible fluid Euler equations in the form

$$\begin{aligned} \frac{\partial \rho}{\partial t} + \operatorname{div}(\rho\,\mathbf{w}) &= 0 \\ \frac{\partial(\rho\,\mathbf{w})}{\partial t} + \operatorname{div}(\rho\,\mathbf{w}^2) + \mathbf{grad}\ p &= 0 \\ \frac{\partial E}{\partial t} + \operatorname{div}(\mathbf{w}\,(E+p)) &= 0, \end{aligned} \tag{2.1}$$

representing the laws of the conservation of mass, momentum in all directions, and the total energy. Here, ρ denotes the fluid density, $\mathbf{w} = (u, v)$ is the vector of fluid velocities in 2D, E is the total energy, and p is the fluid pressure given by the equation of state $p = p(\rho, \epsilon)$, where $\epsilon = E/\rho - |\mathbf{w}|^2/2$ is the specific internal energy of the fluid. In the Eulerian model, the system of equations is discretized on the static (in time) computational mesh. The conservative quantities (as mass, momentum, or total energy) are transfered between the computational cells in the form of the advective flux through their edges.

There exist a comprehensive theory describing many aspects of solving the Euler equations. It describes existence of shock waves, contact discontinuities, and rarefaction waves in the solution. Also dependence of the solution on the fluid velocity compared with the sound speed is characterized, and special cases of initial conditions where the solution is analytically expressible, are identified. There exist a big number of CFD monographs about this topic, we recommend [62, 61, 93] for details.

As for the numerical methods for solution of the hyperbolic Euler equations, there exist lots of them with different properties. There are techniques for analyzing the numerical methods as for their stability or order of accuracy. For an overview of the numerical Eulerian schemes and their analysis, see [93, 34]. Comparison of several Eulerian-type methods is done in [66].

Many Eulerian codes and simulation tools exist for modeling of different phenomena in the field of compressible fluid dynamics, for different purposes. Let us review several of them.

On the first place, we name the package `CLAWPACK` [63] developed by Randall LeVeque at the University of Washington. This package for solving time dependent conservation laws in 1D, 2D, and 3D (including

the system of Euler equations), uses sophisticated flux splitting scheme based on advection ideas. The code is adapted for the user to include his own numerical scheme, and use it in the `CLAWPACK`. Parallel version of the code is available, as well as the adaptive mesh refinement extension named `AMRCLAW`.
`ALLSPD-3D Combustor Code` [21] is a numerical tool developed in NASA for simulating the chemically reacting flows in the aerospace propulsion systems. The code can simulate multi-phase turbulences and flows with wide range of Mach-number in combustors of complex geometry. It uses 3D multi-zone structured grids with internal blockages and finite difference, compressible flow formulation with low Mach number preconditioning for both laminar and turbulent flows.
Another important simulations code `FLASH` [5], is being developed at the University of Chicago. It is a modular, adaptive, parallel simulation code capable of handling general compressible flow problems in astrophysics. Its aim is to solve the long-standing problem of thermonuclear flashes on the surfaces of compact stars such as neutron stars and white dwarf stars, and in the interior of white dwarfs.
Finally, we mention the `SAGE` [37] (Simple Adaptive Grid Eulerian) code developed in the Lawrence Livermore national Laboratory and the Los Alamos National Laboratory. It is a massively parallel multimaterial hydrodynamical code for solving of high deformation flow problems. By adding the temperature radiation to the code, `RAGE` code was developed. These codes are applicable for a variety of simulations, such as laser-target interaction, meteorite impact simulations, or tsunami wave spread simulations. The SAGE/RAGE codes are not available for public usage.

2.2 Lagrangian Methods

The philosophy of the Lagrangian-type methods is completely different from the Eulerian philosophy. In the Lagrangian model, the conservation system is solved in the form

$$\begin{aligned} \frac{1}{\rho}\frac{d\,\rho}{d\,t} &= -\nabla\cdot\mathbf{w} \\ \rho\,\frac{d\,\mathbf{w}}{d\,t} &= -\nabla\,p \\ \rho\,\frac{d\,\epsilon}{d\,t} &= -p\,\nabla\cdot\mathbf{w}, \end{aligned} \tag{2.2}$$

where

$$\frac{d}{d\,t} = \frac{\partial}{\partial\,t} + \frac{\partial\,x}{\partial\,t}\,\frac{\partial}{\partial\,x} + \frac{\partial\,y}{\partial\,t}\,\frac{\partial}{\partial\,y} = \frac{\partial}{\partial\,t} + \mathbf{w}\times\nabla \tag{2.3}$$

is the Lagrangian derivation in 2D Cartesian geometry. The Lagrangian method solves this system on the computational mesh moving with the fluid according to the ordinary differential equations

$$\dot{\mathbf{z}}_n = \mathbf{w}_n \tag{2.4}$$

for all nodes n, where $\mathbf{z}_n(t) = (x_n(t), y_n(t))$ is the position of node n and $\mathbf{w}_n$ its velocity. The symbol $\dot{\mathbf{z}}_n$ denotes temporal derivative of the nodal position. Masses in all cells remain constant. There is no mass flux (and thus no advective momentum or energy flux) through the cell edges as in the Eulerian model. For completeness, we demonstrate here, that the Eulerian (2.1) and Lagrangian (2.2) systems are equivalent. We show this equivalence for the case of 1D Cartesian equations with velocity $w = u$. The main point is substitution of the total derivatives in the Lagrangian equations by the partial ones (2.3). By this process, the first equation changes to

$$\left(\frac{\partial\rho}{\partial t} + \frac{\partial x}{\partial t}\,\frac{\partial\rho}{\partial x}\right) + \rho\,\left(\frac{\partial u}{\partial x}\right) = 0\,. \tag{2.5}$$

Here, the temporal partial derivative of the coordinate is equal to the actual fluid speed, that is

$$\frac{\partial\rho}{\partial t} + u\,\frac{\partial\rho}{\partial x} + \rho\,\frac{\partial u}{\partial x} = 0\,, \tag{2.6}$$

and after putting the derivatives together

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho\, u)}{\partial x} = 0\,, \tag{2.7}$$

the Eulerian equation for mass conservation is derived. By doing similar process with the momentum equation, we get

$$\rho \left(\frac{\partial u}{\partial t} + \frac{\partial x}{\partial t}\,\frac{\partial u}{\partial x} \right) + \frac{\partial p}{\partial x} = 0 \tag{2.8}$$

and after some algebra

$$\rho\,\frac{\partial u}{\partial t} + \rho\, u\,\frac{\partial u}{\partial x} + \frac{\partial p}{\partial x} = 0\,. \tag{2.9}$$

To get the Eulerian equation for momentum, let us rewrite (2.9) in the form

$$\frac{\partial (\rho\, u)}{\partial t} - u\,\frac{\partial \rho}{\partial t} + \frac{\partial (\rho\, u^2)}{\partial x} - u\,\frac{\partial (\rho\, u)}{\partial x} + \frac{\partial p}{\partial x} = 0\,. \tag{2.10}$$

The two terms with minus sign correspond to the mass equation (2.7) multiplied by the velocity factor, thus they disappear and momentum equation from (2.1) remains. By the similar process, the energy equation can be transformed to the Eulerian methodology. For this transformation, the definition of the specific internal energy ϵ from the previous section, and both mass and momentum equations in Eulerian form are required. Analogical process can be repeated to show the equality of the Lagrangian and the Eulerian formulation also in the case of multidimensional Euler equations, and the Euler equations in the cylindrical geometry.

The usual process in the Lagrangian method is following. At first, the forces due to the pressure gradient are evaluated from the momentum equations. The forces identify the velocities in each node defining the nodal movement. After moving the nodes according to the forces, a new mesh appears, the internal energy is updated from the energy equation, and density is updated from new mesh and constant cells masses Lagrangian assumption.

There exist many types of the Lagrangian methods. The beginning of the Lagrangian history is in late '40s, when the Lagrangian coordinates for the fluid equations were introduced by Courant and Friedrichs [23]. The classical methods reduce the system of the Lagrangian equations to the Hamiltonian system [33] corresponding to the transformation from the motion to the action equations, naturally preserving the total energy. The systematic approach of deriving the discrete operators with the same properties, as the continuous ones, has been developed by Shashkov and is presented in his book [84]. This approach is used in [18] to develop the conservative Lagrangian discretization in cylindrical coordinates.

Lagrangian methods are a standard tool in CFD and are widely used for fluid simulations. Especially, in the simulations of laser plasma, where huge changes of the computational domain during the simulations appear, the Lagrangian methods become eligible due to the natural treatment of the moving boundary conditions, which cause serious problems to the Eulerian methods.

Let us overview few Lagrangian codes. The first code is the classical 2D `HYDRO` code developed and used in the Los Alamos National Laboratory. This code is based on the algorithm described in [83], and represents a wide range of Lagrangian hydrodynamical codes quite similar to `HYDRO`.

Specialized Lagrangian code for the laser-plasma interactions `ATLANT` [43] was developed in the Russian Institute of Mathematical Modeling. This 2D Lagrangian code implements the hydrodynamical one-fluid model with two temperatures. Collisional laser absorption including the ray-tracing laser beam propagation algorithm is included. This code has been applied for many simulations of experiments of laser-plasma interactions. By including the radiation transport in the multi-group approximation, code `LATRANT` [4] was developed.

2.3 ALE Methods

Both the Eulerian and Lagrangian methods have their advantages and disadvantages, making them suitable for solving of different types of problems. The Eulerian methods are excellent for the simulations, where the computational domain does not change very much, and its size and shape remains more or less the same. Eulerian methods are general, applicable to solving of arbitrary system of hyperbolic equations, robust, and stable. On the other hand, the Lagrangian methods are excellent for solving of problems, where significant changes of the computational domain appear, or for problems of contacts of different materials. The changing Lagrangian mesh naturally covers the whole computational domain, the material interface is naturally kept sharp and in right direction, which needs additional complicated procedures in the Eulerian methods. On the other hand, several problems may appear in the Lagrangian process due to the mesh movement. The computational mesh can degenerate – cells with a very small volume can appear, opposite cell edges can intersect each other (hourglass-type cell motion), and negative volume cells can arise. In such situation, the Lagrangian method either fails, or (if treated by the method) quickly increases its numerical error. One way, how to deal with these problems, is to use the Arbitrary Lagrangian-Eulerian (ALE) methods (for more details, see chapter 3).

The ALE algorithm was firstly proposed by Hirt in 1974 [42] and at the beginning, it was used often for the simulations of the solid body deformations. Many authors contributed to this topic [27, 9, 10, 11, 79, 6, 28], but the CFD society did not pay enough attention to them. During the last several years, the situation has changed, and the ALE methods are becoming a modern tool for fluid simulations [47, 24, 3].

When talking about the ALE methods, one can distinguish two main types of the methods. In the first type, the movement of the computational mesh is predefined, using some information about the problem solution. Thus, no tangling of the mesh can appear [28]. In the second type (preferred in our approach) [79, 24, 3], the purely Lagrangian movement is performed. In the situation, when the mesh degenerates, or its quality is low, or just after each several Lagrangian steps, the mesh smoothing technique is applied and followed by the conservative recomputing (remapping) of the conservative quantities to the smoother mesh. Both types of ALE methods are used for real computations in modern ALE codes.

Let us briefly summarize several known ALE codes, used for the CFD simulations. Probably the first ALE code developed and applicable for practical usage is `YAQUI` [1], implementing the original ALE algorithm [42] by its authors. The programs computational scheme was later improved a little, and code `SALE-2D` [2] was released. These codes solve the Navier-Stokes equations on 2D computational mesh, which can either move with the fluid in a typical Lagrangian fashion, can be held fixed in an Eulerian manner, or can move in some arbitrarily specified way. 3D version of the code was also developed.

Another and much newer ALE code is called `ICF3D-Hydro` [88], and as clear from the name, it is a 3D ALE code for simulations of inertial confinement fusion plasmas. It handles general 3D unstructured meshes, using a second order FEM Godunov scheme with a 3D generalization of van Leer slope limiter process for shock stabilization. The code can work in a fully parallel mode on both SMP and MPP architectures.

Another ICF oriented code is `DRACO` [46], which is still under development. It is the ALE hydrodynamical code designated to run in 1D, 2D, and 3D in planar, spherical, and cylindrical geometries. The code uses Lagrangian hydro step, second order rezoning algorithm, and interface tracking method. The code uses SESAME equation of state [12], and works in parallel.

For simulations of the fluid behavior involving strong shock waves, the `ALEGRA` [79] (ALE General Research Application) code was developed in the Sandia National Laboratory. It uses the ALE algorithm on an unstructured mesh, combining the properties of modern fluid Eulerian shock codes with Lagrangian solid mechanics codes. Moreover, magnetohydrodynamics (MHD) is included into it, allowing to perform high energy density plasma simulations (as z-pinch simulations).

Another sophisticated ALE code `CALE` [92] was developed in the Lawrence Livermore National Laboratory, and uses a staggered 2D quadrilateral structured mesh Lagrange scheme followed by the mesh smoothing and remapping step. The code includes a multi-group, flux-limited, radiation diffusion model

in a wide variety of plasma models. The extension of the code CALEICF includes the thermonuclear burning mechanism, and is also extended to 3D HYDRA [73] code representing the state of the art of 3D ICF codes.
One of the top ALE codes is the code CORVUS [6] developed at the British Atomic Weapons Establishment (AWE). This robust and accurate multimaterial code works on the general unstructured 2D meshes. The main strength of the code is in its treatment of the material interfaces. Today, the code is used in different areas, such as the simulations of highly explosive materials explosions, shock/bubble interactions, or high velocity projectile impact to the tank of water. The code is currently being extended to the 3D fully parallel code PEGASUS.
Finally, we refer to the ALE code ALE Inc. [69], being currently under development in the Los Alamos National Laboratory. This 2D code uses exactly the same approach, as this thesis – Lagrangian step based on the compatible discretization [84, 18], several types of mesh rezoning techniques, and swept-region integration based remapping method [58]. The author of this thesis communicates with the authors of the ALE Inc. code a lot, collaborate in several fields, and many ideas and algorithms are similar. The main difference is in the purpose of the codes. The ALE Inc. code is the incubator – the testing environment for confirming of the properties of the developed methods in different aspects of the ALE field (a similar developmental code focusing on the combination of the ALE method with the adaptive mesh refinement approach (AMR) is described in [3]). On the other hand, our code described in this thesis, is a specialized ALE code for the laser plasma simulations, including many physical aspects of the simulations, such as heat conductivity, plasma equation of state, or the interaction of plasma with a laser beam.

2.4 Hydrodynamical Codes for Plasma Simulations

The field of plasma physics appeared at the turn of the 19th and 20th century. The reflection of radio waves led to the discovery of the Earth's ionosphere, a layer of partially ionized gas in the upper atmosphere. During the first half of the 20th century, many astrophysical discoveries produced a requirement of understanding the plasma physics. Around the year 1940, Hannes Alfvén formulated the theory of magnetohydrodynamics (MHD), in which plasma is treated as a conducting fluid, and its conclusions were confirmed by astrophysical observations. The creation of the hydrogen bomb in 1952 generated a great deal of interest in controlled thermonuclear fusion as a possible power source for the future in the late 1950's and the early 1960's. In these years, theoretical plasma physics first emerged as a mathematically rigorous discipline. The development of high powered lasers in the 1960's opened up the field of laser plasma physics. When an intense laser beam strikes a solid target, material is immediately ablated, and a plasma forms at the boundary between the beam and the target. The major application of laser plasma physics is the approach to fusion energy known as the inertial confinement fusion. The requirement of the simulation tools for plasma physics arises, leading to the development of many codes for numerical modeling of various plasma physics phenomena, based on both kinetic and hydrodynamic approaches. In this section, we summarize several hydrodynamical codes for the plasma simulations.
The state of the art of the plasma simulation is represented by 3D codes applied in the fusion research, typically used on high performance servers and clusters for simulations of the inertial confinement fusion (ICF) processes. Such codes usually solve the full set of the Maxwell equations by their direct integration, taking all electromagnetic effects into account, such as charges and currents generated by the charges in the plasma, or the connection of the full spectrum radiation and the plasma charges. In such critical ICF simulations, processes as the ion-ion and ion-wall collisions has to be taken into account. Such sophisticated and complex codes do not only simulate performed experiments, but are also used for investigating new and unknown phenomena in the plasma field.
In the previous three sections, we have already described several complex codes used for plasma simulations, Eulerian FLASH and SAGE/RAGE codes, the Lagrangian ATLANT code, and ALE ICF3D-Hydro, DRACO, ALEGRA, and CALE codes. All of them were based on the fluid models of plasma. Let us review

several more codes for fluid plasma simulations MEDUSA and ALPS, and the hybrid fluid/particle codes LASNEX and VORPAL.

Classical code MEDUSA [22] is a 1D Lagrangian hydrodynamical code, which was written to investigate some of the hydrodynamic and plasma processes that take place in a small pellet which is irradiated by laser light. Its main purpose is to perform simulations of processes appearing in ICF, and verify its feasibility. In the code's model, the Navier-Stokes equations are supplemented by separate heat conduction equations for the ion and electron temperatures, and a variety of additional effects are included.

Adaptive Laser Plasma Simulator (ALPS) [29] developed in the Lawrence Livermore National Laboratory is a 2D and 3D hydrodynamical code, using an Eulerian high-order accurate upwind technique combined with the adaptive mesh refinement (AMR) approach. The laser beam propagation is modeled using a paraxial wave equation. The main goal of this code is in modeling of laser plasma instabilities, and developing techniques to make laser beams more resistant to them.

The LASNEX code [40] was developed in the Lawrence Livermore National Laboratory to study inertial confinement fusion (ICF) processes, and to construct and analyze ICF experiments. It was first referred to in literature in 1972, but over time the code has evolved and has been greatly enhanced. It includes the Lagrangian (or simple ALE) hydrodynamics, the electron, ion, and radiation heat conduction, and the coupling among these energy fields. Thermonuclear reaction can be modeled by LASNEX, including the energy produced as well as the reaction products. It provides sufficient agreement between the calculations and the experiments, which makes this code eligible to predict and design future experiments, such as on the National Ignition Facility, for example.

Finally, we state the code developed at the University of Colorado – VORPAL [76], a versatile plasma simulations code. It includes both PIC and fluid plasma models, incorporating several models of plasma and electromagnetic field behavior in 1D, 2D, and 3D. Both modes (PIC and fluid) can run independently or in a hybrid regime. VORPAL is currently being used to study a number of plasma physics problems, such as generation of laser wake fields, optical injection of particles into wake fields, or radio frequency heating of fusion plasmas.

Chapter 3

ALE Method in Cartesian Geometry

In this section, we describe the basic principles of the ALE methods in the Cartesian geometry. We go through all three main parts of the ALE algorithm – the Lagrangian solver, mesh smoothing techniques, and conservative interpolation algorithms. We focus on the remapping techniques, which are the main contribution of the author to the field of ALE methods.
The ALE algorithm is based on the combination of the Eulerian and the Lagrangian methods. The computational mesh moves with the fluid (as in the Lagrangian approach), however the mass flux over cell boundaries is allowed (as in the Eulerian approach). The ALE method combines advantages of both approaches. Because of the moving mesh, it allows to simulate problems on moving or changing computational domain. The computational mesh is smoothed from time to time, causing less problems with negative cell volumes, hourglass-type motion, sharp edge angles, and other configurations, which are hardly to treat by pure Lagrangian methods.

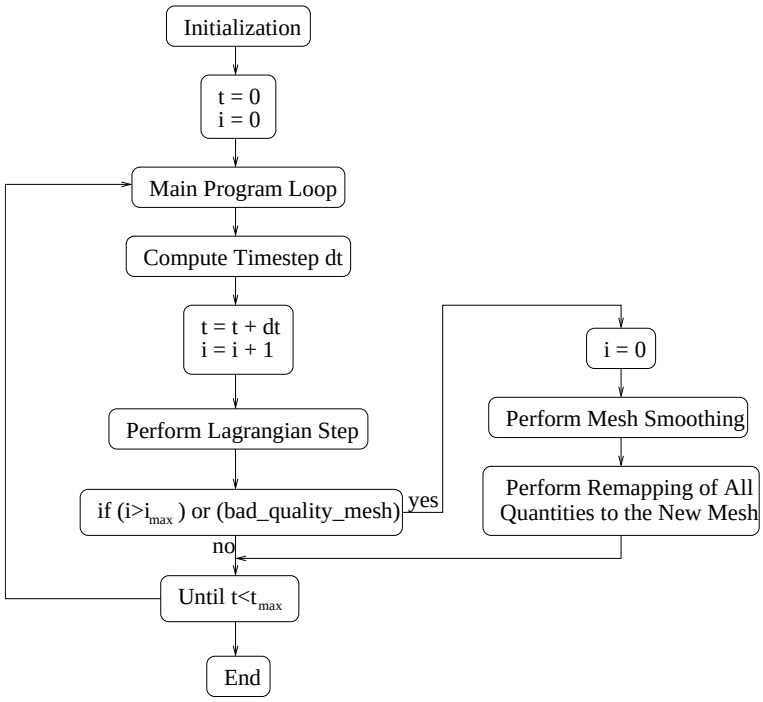


Figure 3.1: Structure of the ALE algorithm.

In the ALE algorithm, the Lagrangian-type and Eulerian-type steps are used. At first, several time steps of the Lagrangian solver are performed. After a selected number of such steps, or if the mesh becomes invalid or has bad quality, the Eulerian-type step is used. It includes a mesh smoothing sub-

step providing us a new mesh, which is already regular and better quality, but similar to the original Lagrangian mesh. The second part of the Eulerian-type step is the conservative interpolation substep, which conservatively remaps all conservative quantities from the Lagrangian mesh, to the new, smoothed one. The basic structure of the ALE algorithm is shown in Figure 3.1. Let us go through all the mentioned parts of the ALE method in detail.

3.1 Lagrangian Solver

There exist many approaches for formulating the Lagrangian schemes. For completeness we mention here the well known point-centered method with all quantities defined in the centers of the computational cells, reviewed in [18]. In general, this type of methods has problems with the total energy conservation. On the other hand, it is quite easy to incorporate such method into complete ALE code.
We employ the classical staggered Lagrangian methods with the scalar quantities (such as fluid pressure, density, or internal energy) defined in the centers of the computational mesh cells, and the vector quantities (such as positions or velocities) defined in the mesh nodes. We use discretization from [18, 19, 20, 16], which is one of the most suitable approaches for the Lagrangian hydrodynamical simulations of the fluid dynamics. This method is fully conservative, and allows a natural treatment of boundary conditions, which are very important for laser plasma simulations. On the other hand, due to the staggered discretization, it requires more sophisticated method for conservative quantity remapping to make the Lagrangian solver compatible with the rest of the ALE code.
The Lagrangian solver is based on the computation of three types of forces in each node of the computational mesh, and movement of the nodes according to these forces. They include the zonal pressure force, the subzonal pressure force, and the artificial viscosity force. The zonal pressure force represents the total force of the fluid caused by the pressure gradient, affecting the node from all cells around it. The subzonal pressure force arises from the finer discretization (subzones), and prevents the nodes from unphysical hourglass-type motion. The viscosity force adds artificial diffusion to the solution, which makes the Lagrangian solver able to perform also simulations including shock waves or contact discontinuities. Let us discuss the Lagrangian solver and the forces in more details.

3.1.1 Geometrical Structure

The Lagrangian solver uses staggered subzonal discretization on logically-orthogonal computational mesh characterized in Figure 3.2. Each cell is divided into four subzones, separated by four dashed lines (separators) connecting the cell center with the center of each edge of the cell. All cell edges, subzones, and dashed separators are enumerated from 1 to 4 in the particular cell, starting from the lower (edges, separators) or lower-left (subzones) ones, in the counter-clock wise order. For further integration, we also need orientation of all cell edges and separators, so let us define, that all separators s point from the cell center to the corresponding edge center, and all subzonal edges a^+, a^- have their directions from the nodes to the edge centers.
The centers of the edges are computed as the average of the coordinates of both end nodes. The original Lagrangian scheme uses simple averaging also for the computation of the cell centers. To incorporate the Lagrangian method into the ALE context, we changed the cell centers to centers of mass (centroids), which is needed by the remapping part of the algorithm. This change does not affect the order of accuracy of the Lagrangian scheme, as demonstrated later. The centroid of cell c is computed as

$$x_c = \frac{\int_c x \, dx \, dy}{V_c} , \qquad y_c = \frac{\int_c y \, dx \, dy}{V_c} , \tag{3.1}$$

where the formula for the volume of cell c is

$$V_c = \int_c 1 \, dx \, dy . \tag{3.2}$$

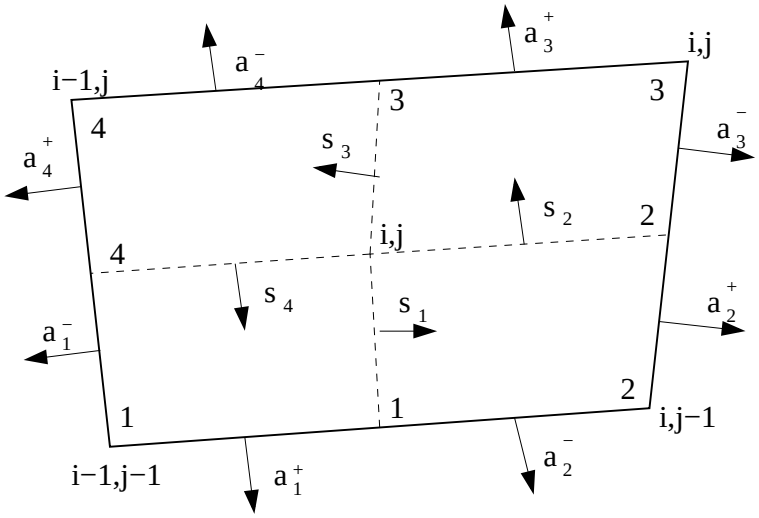


Figure 3.2: Structure of one computational cell i, j and enumeration of the nodes around it. It includes enumeration of cell edges and cell subzones, and orientation of $\mathbf{a_l^-}$, $\mathbf{a_l^+}$, and $\mathbf{s_l}$ vectors, $l = 1, \ldots, 4$.

By using the Green formula, the volume integrals can be reduced to the boundary integrals

$$V_c = \sum_{e \in E(c)} \int_e x \, dy \tag{3.3}$$

$$x_c = \frac{\frac{1}{2} \sum\limits_{e \in E(c)} \int_e x^2 \, dy}{V_c} \tag{3.4}$$

$$y_c = \frac{\frac{1}{2} \sum\limits_{e \in E(c)} \int_e x \, y \, dy}{V_c} \,, \tag{3.5}$$

where $E(c)$ is set of edges of cell c. After evaluating the line integrals, we get the final formula

$$V_c = \frac{1}{2} \left((y_2 - y_1)(x_1 + x_2) + (y_3 - y_2)(x_2 + x_3) + (y_4 - y_3)(x_3 + x_4) + (y_1 - y_4)(x_4 + x_1) \right) \tag{3.6}$$

for the volume of any quadrilateral cell with nodal coordinates (x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4) in counter-clock wise order. Analogically, we get the formula for the coordinates of the centroid of the computational cell. It can be shown that the final formulas are the same, if the opposite part of the Green formula is used, for example if the cell volume is computed as

$$V_c = \sum_{e \in E(c)} \int_e -y \, dx \,. \tag{3.7}$$

This is, of course, expected, but it gives us a simple test of correctness of our formulas.
Subzonal mass of subzone $l = 1 \ldots 4$ of cell c is naturally expressed as

$$m_c^l = \rho_c^l \, V_c^l \,, \tag{3.8}$$

where the subzonal volume V_c^l is computed using the same formula as for the zonal volume (3.6), ρ_c^l represents subzonal density defined by formula (3.8), and the subzonal mass m_c^l results from the initialization process and remains unchanged during the whole Lagrangian computation. Nodal and cell

(zonal) masses are naturally defined as

$$m_n = \sum_{c' \in C(n)} m_{c'}^{l_{c'}^n}, \quad m_c = \sum_{l=1}^{4} m_c^l \tag{3.9}$$

respectively. Here l_c^n means subzone of cell c corresponding to node n, and $C(n)$ is a set of all cells including node n[1]. This allows us to define the zonal (cell) density (and also nodal density similarly) naturally as

$$\rho_c = \frac{m_c}{V_c}, \tag{3.10}$$

because

$$V_c = \sum_{l=1}^{4} V_c^l. \tag{3.11}$$

3.1.2 Zonal Pressure Force

By integration of the momentum equation in x direction

$$\rho \frac{\partial u}{\partial t} = -\frac{\partial p}{\partial x} \tag{3.12}$$

over the nodal volume (four subzones from four different cells around a common node) characterized in Figure 3.3, we get

$$m_n \left(\frac{\partial u}{\partial t}\right)_n = -\int_{V_n} \frac{\partial p}{\partial x}\, dx\, dy, \tag{3.13}$$

where m_n is the nodal mass (3.9), V_n is the nodal volume defined as sum of the corresponding subzonal volumes, and $(\partial u/\partial t)_n$ is the average temporal derivative of the fluid velocity in V_n. We have assumed that the nodal velocity u_n is constant in the whole nodal volume V_n. The right hand side of equation (3.13) represents the total force affecting the nodal movement in x direction, which we denote by the symbol F_n^x. By applying the Green theorem to the right hand side, we get the boundary integral

$$F_n^x = -\oint_{\partial V_n} p\, dy, \tag{3.14}$$

which can be decomposed into a sum of edge integrals

$$F_n^x = -\sum_{l=1}^{4} \left(-\int_{s_l^{c(n,l)}} p\, dy + \int_{s_{l-1}^{c(n,l)}} p\, dy \right), \tag{3.15}$$

where by the symbol $c(n,l)$ we mean the cell, whose subzone with the index l corresponds to the node n, and s_l^c denotes the separator inside cell c with the index l. In this thesis, the l index is cyclic (as modulo of 4), in the sense that $s_0^c = s_4^c$, and similarly for all terms, where l appears. Here, we assume constant pressure p_c inside each cell c. Then, one can put pressures in front of the integrals and rewrite the formula as

$$F_n^x = \sum_{l=1}^{4} p_{c(n,l)} \left(\int_{s_l^{c(n,l)}} dy - \int_{s_{l-1}^{c(n,l)}} dy \right). \tag{3.16}$$

The edge integrals can be evaluated from the ending points of the separators, so let us define

$$I\left(s_{l'}^{c(n,l)}\right) = \int_{s_{l'}^{c(n,l)}} dy = y_{c(n,l)} - y_{e_{l'}^{c(n,l)}}, \tag{3.17}$$

[1]Generally, in this paper, we use capitals to note sets of objects, for example $C(c)$ means all cells in the neighborhood of cell c (in our logically rectangular mesh, $C(c)$ is a 3×3 patch of cells sharing an edge or vertex with c, including cell c), $E(n)$ are all edges connected to node n, or $N(c)$ represents all nodes of cell c.

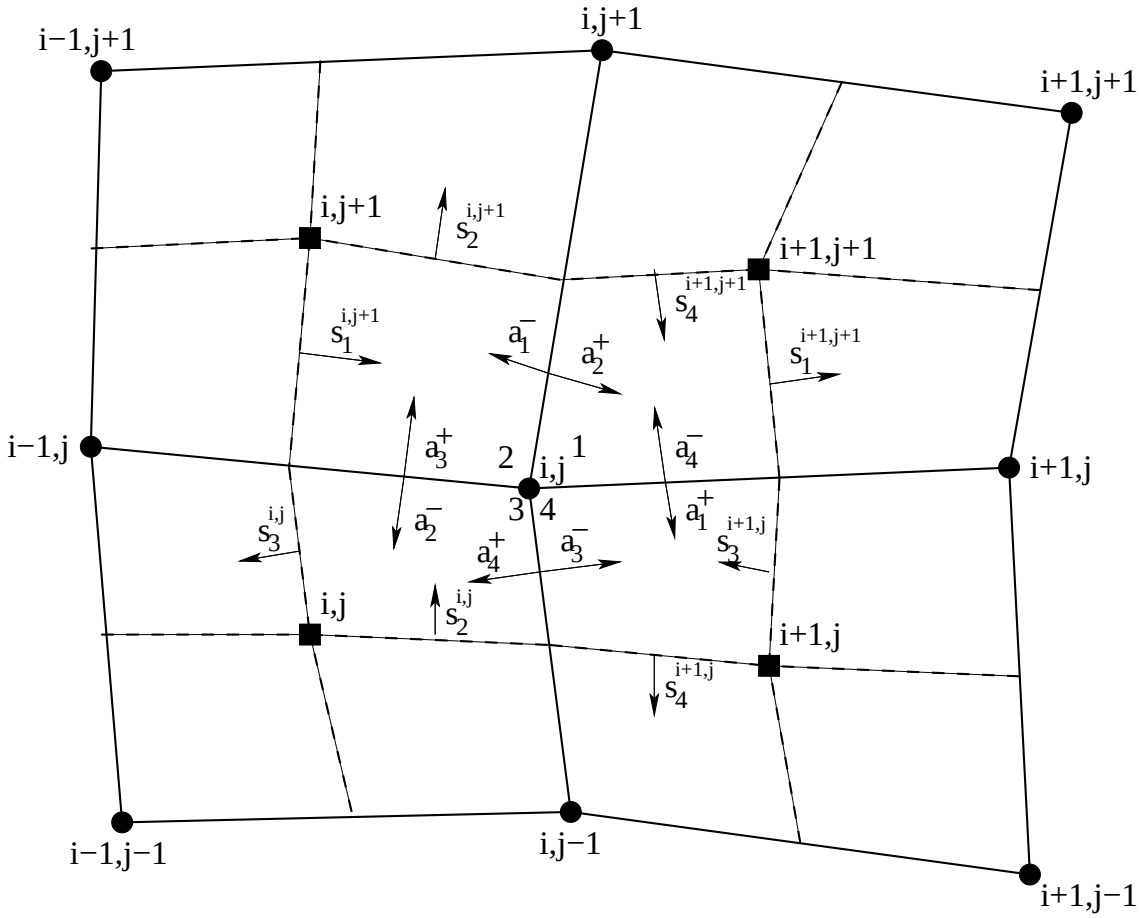


Figure 3.3: Situation in a nodal volume corresponding to node i, j and four cells c around: i, j, $i+1, j$, $i+1, j+1$, and $i, j+1$. Outer normals of cell sides (vectors $\mathbf{a}_l^{\pm}$) and normals of separators (vectors $\mathbf{s}_l^c$) are schematically shown.

where the symbol $e_{l'}^{c(n,l)}$ represents the edge of cell $c(n,l)$ with index l', and the formulas for the coordinate of the cell center y_c and edge center y_e are defined in the previous section 3.1.1. Now, one can rewrite the formula as

$$F_n^x = \sum_{l=1}^{4} p_{c(n,l)} \left(I(s_l^{c(n,l)}) - I(s_{l-1}^{c(n,l)}) \right) = \sum_{l=1}^{4} Fp_{c(n,l)}^{x\,l} \,, \tag{3.18}$$

or explicitly for the node $n = i, j$ as

$$\begin{aligned} F_{i,j}^x = {} & p_{i+1,j+1} \left(I(s_1^{i+1,j+1}) - I(s_4^{i+1,j+1}) \right) + p_{i,j+1} \left(I(s_2^{i,j+1}) - I(s_1^{i,j+1}) \right) \\ & + p_{i,j} \left(I(s_3^{i,j}) - I(s_2^{i,j}) \right) + p_{i+1,j} \left(I(s_4^{i+1,j}) - I(s_3^{i+1,j}) \right) . \end{aligned} \tag{3.19}$$

Every term of the previous formula corresponds to the corner pressure force $\mathbf{Fp}_{c(n,l)}^l$, influencing the node from the particular subzone. In the later text, we will denote the nodal pressure force by the symbol Fp_n^x.

For completeness, exactly the same formula can be used for the computation of the zonal pressure force in y direction

$$F_n^y = \sum_{l=1}^{4} p_{c(n,l)} \left(J(s_l^{c(n,l)}) - J(s_{l-1}^{c(n,l)}) \right) = \sum_{l=1}^{4} Fp_{c(n,l)}^{y\,l} \,, \tag{3.20}$$

just the integrals are computed as

$$J\left(s_{l'}^{c(n,l)}\right) = \int_{s_{l'}^{c(n,l)}} dx = -x_{c(n,l)} + x_{e_{l'}^{c(n,l)}}\,, \tag{3.21}$$

the different sign arises from the Green theorem. These relations give us the complete formulas for the computation of the zonal forces from each subzone to the particular node.

3.1.3 Subzonal Pressure Force

To explain the subzonal pressure force concept, let us suppose now that the pressure is not constant in each cell. Let us make finer discretization – let us define the subzonal pressures as

$$p_c^l = \frac{v_c^{*2}}{\gamma}\,\rho_c^l \tag{3.22}$$

or the subzonal pressure difference

$$\delta p_c^l = \frac{v_c^{*2}}{\gamma}\,(\rho_c^l - \rho_c)\,, \tag{3.23}$$

where the sound speed v^* is computed from the equation of state, and γ is the ratio of the specific heats. The zonal and subzonal densities are computed by dividing the particular mass by the corresponding volume. These zonal and subzonal masses are equal to the initial values from the first Lagrangian step, the densities change in time as the computational mesh moves. Easily, the following relation holds

$$p_c^l = p_c + \delta p_c^l\,, \tag{3.24}$$

which expresses the subzonal pressure as the variation of the zonal one.
Now, let us repeat the same process, as in the previous section. It continues till the formula (3.15) exactly the same way, as before, but now, one cannot put the pressure in front of the integrals, because there is no unique pressure value in the cell. Thus, one have to define a value of pressure along the separator lines, which can simply be done by averaging the adjacent subzonal pressures. Then, the formula can be rewritten as

$$F_n^x = -\sum_{l=1}^{4}\left(-\frac{p_{c(n,l)}^l + p_{c(n,l)}^{l+1}}{2}\int_{s_l^{c(n,l)}} dy + \frac{p_{c(n,l)}^l + p_{c(n,l)}^{l-1}}{2}\int_{s_{l-1}^{c(n,l)}} dy\right), \tag{3.25}$$

and after putting the formula (3.24) and the notation (3.17), we get

$$F_n^x = -\sum_{l=1}^{4}\left(-\left(p_{c(n,l)} + \frac{\delta p_{c(n,l)}^l + \delta p_{c(n,l)}^{l+1}}{2}\right) I(s_l^{c(n,l)}) + \left(p_{c(n,l)} + \frac{\delta p_{c(n,l)}^l + \delta p_{c(n,l)}^{l-1}}{2}\right) I(s_{l-1}^{c(n,l)})\right). \tag{3.26}$$

After regrouping the terms, one can write the formula in several different forms including the zonal pressure force Fp_n^x defined in (3.18),

$$\begin{aligned} F_n^x &= Fp_n^x + \frac{1}{2}\sum_{l=1}^{4}\left(\left(\delta p_{c(n,l)}^l + \delta p_{c(n,l)}^{l+1}\right) I(s_l^{c(n,l)}) - \left(\delta p_{c(n,l)}^l + \delta p_{c(n,l)}^{l-1}\right) I(s_{l-1}^{c(n,l)})\right) && (3.27)\\ &= Fp_n^x + \sum_{l=1}^{4}\delta p_{c(n,l)}^l\,\left(I(s_l^{c(n,l)}) - I(s_{l-1}^{c(n,l)})\right) && (3.28)\\ &\quad + \frac{1}{2}\sum_{l=1}^{4} I(s_l^{c(n,l)})\left(\delta p_{c(n,l)}^{l+1} - \delta p_{c(n,l)}^l\right) + \frac{1}{2}\sum_{l=1}^{4} I(s_{l-1}^{c(n,l)})\left(\delta p_{c(n,l)}^l - \delta p_{c(n,l)}^{l-1}\right)\\ &= Fp_n^x + \sum_{l=1}^{4} Fdp_{c(n,l)}^{x\,l}\,. \end{aligned}$$

Especially, the formulation (3.28) is important, because it demonstrates the geometrical meaning of the subzonal pressure force. The first term of the force corrects directly the zonal pressure force due to the different pressure in the zone and in the subzone. The second and the third terms are associated with the midpoints of the particular cell edges and play a very important role in reducing the artificial mesh movement. For more details about the subzonal discretization and different force formulations, see [19]. Let us also note, that (exactly as in the previous section 3.1.2) the subzonal pressure force in y direction can be computed using the same formulas, just the integrals must be defined as in (3.21).
Due to the fact, that the integral of 1 over the boundary of a subzone must be equal to zero (subzone is closed)

$$0 = \int_{\partial V_c^l} 1 \, dy = -I(s_l^c) + I(s_{l-1}^c) - I(a_l^{c-}) + I(a_l^{c+}) \,, \tag{3.29}$$

the integrals over the separators s can be replaced by the integrals over the outer half-edges of the cell a,

$$I(s_l^c) - I(s_{l-1}^c) = I(a_l^{c+}) - I(a_l^{c-}) \,, \tag{3.30}$$

where

$$I(a_l^{c\pm}) = \int_{a_l^{c\pm}} dy = y_{e(c\pm,l)} - y_{n(c,l)} \,. \tag{3.31}$$

This allows us to reformulate the force formulas (3.18), (3.28), which may be useful in the particular implementations.
In the later text, we use the term subzonal pressure force and notation Fdp_n^x for the part of the formula (3.28) not including the original zonal pressure force Fp_n^x. In the real computations, we multiply this subzonal pressure force by the merit factor $f_M \in \langle 0, 1 \rangle$. It is an important issue for setting the amount of the subzonal pressure force, which regulates the measure of removing the unreal hourglass-type motion of the computational mesh. It is not generally easy to say, how to select the merit factor, in real simulations we usually select it close to the center of the feasible interval, $f_M \approx 1/2$. Automatization of selecting the merit factor is analyzed in [19].

3.1.4 Viscosity Force

Viscosity is an important part of the total nodal force. Without the artificial viscosity, the Lagrangian solver is not able to simulate problems including shock waves and contact discontinuities. This is important particularly in the field of laser plasma target impacts, which we focus on.
There exist many approaches, how to incorporate the artificial viscosity to the solution. The original approach coming from [95] adds a nonlinear term in the form

$$q_c = C \, \rho_c \, (\Delta \mathbf{w})^2 \tag{3.32}$$

to the pressure p_c in cell c, where shock compression appears. Here, C is an unity order constant, and $\Delta \mathbf{w}$ is the velocity difference over this cell. Purpose of this term is preventing the cell c to collapse during the computational time step Δt satisfying the CFL stability condition $v_c^* \, \Delta t / \Delta x_c \leq 1$, where v_c^* is the sound speed in cell c appropriate to the fluid pressure $p_c + q_c$, and Δx_c is the characteristic length of the zone. To enforce the viscosity pressure to be dissipative, we define it non-zero only in the case of zonal compression. Then, the nodal velocity (and consequently kinetic energy) is via this mechanism transformed into the higher pressure (and consequently into the higher internal energy), so it acts as the real viscosity.
The previous formulation prevents the zone from collapsing, but does not remove the unphysical oscillations around the shock wave. To eliminate them, linear viscosity formulation was developed [60]

$$q_c = C \, \rho_c \, v_c^* \, |\Delta \mathbf{w}| \,, \tag{3.33}$$

which diminish slower around the shock wave and thus eliminates more compression in the contiguous cells. Many Lagrangian codes use the artificial viscosity in the form of combination of the previous two formulas,

$$q_c = C_1 \, \rho_c \, v_c^* \, |\Delta \mathbf{w}| + C_2 \, \rho_c \, (\Delta \mathbf{w})^2 \,, \tag{3.34}$$

and typically the following relation for the constants holds, $C_1 \leq C_2$.
Finally, we present here the Kuropatenko combination of the linear and non-linear viscosity combination [59, 96], which we will denote by q^{Kur}:

$$q_c^{\mathrm{Kur}} = \rho_c \left(C_2 \, \frac{\gamma+1}{4} \, |\Delta \mathbf{w}| \; + \sqrt{C_2^2 \left(\frac{\gamma+1}{4} \right)^2 (\Delta \mathbf{w})^2 + C_1^2 \, (v_c^*)^2} \right) |\Delta \mathbf{w}| \,. \tag{3.35}$$

Typically, the coefficients are selected as $C_1 = C_2 = 1$. Obviously, this formula reduces to the linear viscosity for $\Delta v \to 0$, and to the non-linear formula for $v_c^* \to 0$. Both in our practical tests and [20], this formulation produces high-quality results and we use it as a viscosity pressure for our simulations. However, there are several ways how to incorporate the viscosity pressure into the viscosity nodal forces which we describe later.

Bulk Viscosity

The easiest way to add the viscosity to the solution, is the bulk formulation [20]. The first possibility is to include the Kuropatenko viscosity pressure (3.35) directly into the pressure of each zone, and compute the zonal pressure forces Fp_n^x (3.18) in each node n from such modified pressures $p_c + q_c^{\mathrm{Kur}}$.
The second way is to construct the subzonal viscosity forces, and add them to the total nodal force. In fact, this approach is not needed in the case of bulk viscosity, but it is required for other viscosity formulations, where the easy way is not possible. In the case of the bulk viscosity, the viscosity force is

$$Fq_n^x = \sum_{l=1}^{4} q_{c(n,l)}^{\mathrm{Kur}} \left(I(s_l^{c(n,l)}) - I(s_{l-1}^{c(n,l)}) \right) = \sum_{l=1}^{4} Fq_{c(n,l)}^{x\,l} \tag{3.36}$$

and analogically in the y direction, just the pressure is replaced by the Kuropatenko viscosity pressure in equation (3.18). The equivalence of both approaches is apparent.

Edge Viscosity

The edge viscosity is also constructed to decrease the kinetic energy and transform it to the internal one. It was introduced in [20]. The basic principle of the edge-centered viscosity is the evaluation of the Kuropatenko viscosity (3.35) along every edge $e(c,l)$ of cell c,

$$q_{e(c,l)}^{\mathrm{Kur}} = \rho_{e(c,l)} \left(C_2 \, \frac{\gamma+1}{4} \, |\Delta \mathbf{w}_{e(c,l)}| \; + \sqrt{C_2^2 \left(\frac{\gamma+1}{4} \right)^2 (\Delta \mathbf{w}_{e(c,l)})^2 + C_1^2 \, (v_{e(c,l)}^*)^2} \right) |\Delta \mathbf{w}_{e(c,l)}| \,. \tag{3.37}$$

Here the edge density and sound speed is computed from the nodal densities and sound speeds on both sides of the edge

$$\rho_{e(c,l)} = 2 \, \frac{\rho_{n(e(c,l),-)} \, \rho_{n(e(c,l),+)}}{\rho_{n(e(c,l),-)} + \rho_{n(e(c,l),+)}} \,, \quad v_{e(c,l)}^* = \min(v_{n(e(c,l),-)}^* \,, v_{n(e(c,l),+)}^*) \,, \tag{3.38}$$

where by the symbols $n(e,-)$ and $n(e,+)$ represent the starting and ending nodes of edge e, the nodal density ρ_n is naturally computed as

$$\rho_n = \frac{\sum\limits_{c \in C(n)} \rho_c^{l_c^n} \, V_c^{l_c^n}}{\sum\limits_{c \in C(n)} V_c^{l_c^n}} \,. \tag{3.39}$$

The nodal sound speed v_n^* is obtained by the same formula, with the zonal volumes and sound speed instead of subzonal volumes and densities. The velocity difference $\Delta\mathbf{w}_e$ along the edge e is computed as

$$\Delta\mathbf{w}_e = \sqrt{\delta u_e^2 + \delta v_e^2}\,, \tag{3.40}$$

where

$$\delta u_e = u_{n(e,+)} - u_{n(e,-)} \tag{3.41}$$

$$\delta v_e = v_{n(e,+)} - v_{n(e,-)}\,. \tag{3.42}$$

This allows us to write the edge-centered viscosity force as

$$f^x_{e(c,l)} = (1 - \psi_{e(c,l)})\, q^{\mathrm{Kur}}_{e(c,l)}\, \delta u_{e(c,l)}\, \frac{\delta u_{e(c,l)}\, I\,(s_l^c) + \delta v_{e(c,l)}\, J\,(s_l^c)}{\Delta\mathbf{w}^2_{e(c,l)}}\,, \tag{3.43}$$

$$f^y_{e(c,l)} = (1 - \psi_{e(c,l)})\, q^{\mathrm{Kur}}_{e(c,l)}\, \delta v_{e(c,l)}\, \frac{\delta u_{e(c,l)}\, I\,(s_l^c) + \delta v_{e(c,l)}\, J\,(s_l^c)}{\Delta\mathbf{w}^2_{e(c,l)}}\,. \tag{3.44}$$

The previous formulas are used only in the case of cell compression, which means, that the following relation is true

$$\delta u_{e(c,l)}\, I\,(s_l^c) + \delta v_{e(c,l)}\, J\,(s_l^c) \le 0\,, \tag{3.45}$$

otherwise the edge forces are set to zero. Each edge-centered viscosity force is then added/subtracted to/from the total nodal viscosity force,

$$Fq^x_n = \sum_{l=1}^{4} (f^x_{e(c(n,l),l)} - f^x_{e(c(n,l),l-1)}) = \sum_{l=1}^{4} Fq^{x\,l}_{c(n,l)} \tag{3.46}$$

and similarly in the y direction.
We still did not comment the edge viscosity limiter $\psi_{e(c,l)}$. It helps to reduce the artificial viscosity also in regions of linear velocity gradients, where the viscosity is not required (case of shock-less compression). On the other hand, in some simulations from the field of laser plasma, the limiting process causes less diffusion and unsteady solution, so we kept the possibility in the code to switch the limiting off. It is necessary to test for the particular problem type, whether to use limiting or not. As for the formulas for the limiter evaluation, we refer to the original paper [20], including also more details about the complete edge viscosity process.

Tensor Viscosity

The tensor form of the artificial viscosity generalizes the previous edge formulation. Again, it is based on the scalar Kuropatenko viscosity term, multiplied by the tensor of velocity gradient. Analogically as in the edge viscosity case, the tensor viscosity can include a limiter term switching the viscosity off for the case of shock-less compression. The advantage of this formulation is the reduction of dependence of the solution on the computational mesh. It keeps the useful features of the edge viscosity, but it follows the real physical viscosity in a tensor form.
The tensor viscosity has the following form

$$\mathbf{Q} = \mu\, \mathbf{G}^\top\,, \tag{3.47}$$

where μ is a scalar viscosity coefficient, and $\mathbf{G} = \mathbf{grad}\,\mathbf{w}$ is the tensor of velocity gradient. In [17], the tensor viscosity formulas are derived both in continuous and discrete cases. When compared with the edge viscosity, the viscosity forces are not edge centered any more, but they directly affect the

computational mesh node from a particular cell subzone. In each subzone l of cell c we need to compute the following viscosity coefficient

$$\mu_c^l = (1-\psi_c^l)\,\rho_c^l \left(C_2\,\frac{\gamma+1}{4}\,|\Delta \mathbf{w}_c^l| \; + \sqrt{C_2^2 \left(\frac{\gamma+1}{4}\right)^2 (\Delta \mathbf{w}_c^l)^2 + C_1^2\,(v_c^*)^2} \right) L_c^l \tag{3.48}$$

based on the Kuropatenko modulus again. Here, ψ_c^l is the viscosity limiter, this time computed in each subzone, $\Delta \mathbf{w}_c^l$ is the subzonal velocity jump computed as the maximal velocity difference across the corner volume, and the characteristic subcell length L_c^l is required to keep the correct magnitude of the viscosity coefficient. There are many ways to select the characteristic length, we use the formula suggested in [17]

$$L_c^l = \begin{cases} 2\sqrt{V_c^l}\,\sqrt{\frac{|\mathbf{\Delta x_1}|}{|\mathbf{\Delta x_2}|}} & \text{for} \quad \mathbf{\Delta x_1}\cdot\mathbf{w}_{\text{avg}} > \mathbf{\Delta x_2}\cdot\mathbf{w}_{\text{avg}} \\ 2\sqrt{V_c^l}\,\sqrt{\frac{|\mathbf{\Delta x_2}|}{|\mathbf{\Delta x_1}|}} & \text{for} \quad \mathbf{\Delta x_1}\cdot\mathbf{w}_{\text{avg}} \le \mathbf{\Delta x_2}\cdot\mathbf{w}_{\text{avg}}\,, \end{cases} \tag{3.49}$$

where the average subzonal velocity $\mathbf{w}_{\text{avg}}$ is computed as the arithmetical average of the four velocity values in all subzone vertices. The vectors $\mathbf{x_1}$ and $\mathbf{x_2}$ start in the edge centers of the subzone corresponding to the cell edges, and point to the centers of the separators on the opposite side of the subzone.
The final complicated formula for the viscosity force Fq_c^l from each subzone to the corresponding node including the scalar viscosity coefficient μ_c^l is presented in [17]. As in the case of the other viscosity types, just by summing all these forces around a node, we get the total nodal viscosity force.

3.1.5 Conservative Internal Energy Update

In sections 3.1.2 and 3.1.3 we have shown, that the force is evaluated by integration of the momentum equation. Simply, by substituting the temporal velocity derivative by the central difference

$$m_n \left(\frac{\partial \mathbf{w}}{\partial t}\right)_n = m_n \left(\frac{\mathbf{w}_n^1 - \mathbf{w}_n}{\Delta t}\right) = \mathbf{F}_n\,, \tag{3.50}$$

and just by moving the old velocity $\mathbf{w}_n$ to the right hand side of the equation, we get the formula for the new velocity $\mathbf{w}_n^1$ computation

$$\mathbf{w}_n^1 = \mathbf{w}_n + \frac{\Delta t}{m_n}\,\mathbf{F}_n\,. \tag{3.51}$$

Thus, the momentum conservation in each cell is satisfied due to the force definition and the closeness of the cell.
Now, let us derive the formula for the new specific internal energy computation. At first, we have to define the total energy in cell c as

$$E_c = m_c\,\epsilon_c + \sum_{l=1}^{4} \frac{1}{2}\,m_c^l \left(u_{n(c,l)}^2 + v_{n(c,l)}^2\right)\,, \tag{3.52}$$

which corresponds well to the original formula for the total energy density defined in section 2.1. Due to the conservation of total energy in each cell, the time derivative of this quantity is equal to zero and can be written as

$$0 = \frac{\partial E_c}{\partial t} = m_c\,\frac{\partial \epsilon_c}{\partial t} + \sum_{l=1}^{4} \frac{1}{2}\,m_c^l \left(2\,u_{n(c,l)}\,\frac{\partial u_{n(c,l)}}{\partial t} + 2\,v_{n(c,l)}\,\frac{\partial v_{n(c,l)}}{\partial t}\right) \tag{3.53}$$

and due to equation (3.50), one can rewrite it as

$$0 = m_c\,\frac{\partial \epsilon_c}{\partial t} + \sum_{l=1}^{4} \left(u_{n(c,l)}\,F_c^{x\,l} + v_{n(c,l)}\,F_c^{y\,l}\right)\,. \tag{3.54}$$

Now, let us define the total work, performed in cell c due to the forces $\mathbf{F}_c$ as

$$E_c^{\text{work}} = -\sum_{l=1}^{4} \mathbf{w}_{n(c,l)} \cdot \mathbf{F}_c^l \,, \tag{3.55}$$

which allows us to rewrite the previous equation in the form

$$m_c \frac{\partial \epsilon_c}{\partial t} = E_c^{\text{work}} \,. \tag{3.56}$$

By substituting the specific internal energy derivative by the central difference, we get the final formula for the new specific internal energy

$$\epsilon_c^1 = \epsilon_c + \frac{\Delta t}{m_c} E_c^{\text{work}} \,, \tag{3.57}$$

which is fully conservative.
We have shown, that the conservation of momentum and total energy are naturally satisfied by definition of the corner forces. There is still the issue of the mass conservation. This is guaranteed by the Lagrangian nature of the step. The zonal, subzonal, and consequently the nodal masses are computed during the initialization stage, and then kept unchanged during the whole pure Lagrangian simulation. Only the zonal and subzonal densities are updated in each Lagrangian step according to the nodal movement changing the volumes, the total mass is conserved without any complications. Thus, we have completely conservative Lagrangian scheme for fluid simulations.

3.1.6 Complete Algorithm of the Lagrangian Method

In this section, we describe the complete algorithm of the Lagrangian step, allowing us to solve the system of fluid equations in Lagrangian form (2.2). It follows the process presented in [18, 19, 16].
When the Lagrangian step starts, the following quantities are known from the previous time step or initialization: time step Δt, nodal coordinates $\mathbf{z}_n = (x_n, y_n)$, nodal speeds $\mathbf{w}_n = (u_n, v_n)$, cell and subzone volumes V_c, V_c^n, cell and subzone densities ρ_c, ρ_c^n, cell pressures p_c, and specific internal energies ϵ_c. The complete algorithm can be described in the following way

1. For each subzone, compute the zonal, subzonal, and viscosity pressure forces $\mathbf{Fp}_c^l$, $\mathbf{Fdp}_c^l$, and $\mathbf{Fq}_c^l$ described in sections 3.1.2, 3.1.3, and 3.1.4 respectively.

2. Compute total corner and total nodal forces in each node

$$\mathbf{F}_c^l = \mathbf{Fp}_c^l + \mathbf{Fdp}_c^l + \mathbf{Fq}_c^l \tag{3.58}$$

$$\mathbf{F}_n = \sum_{l=1}^{4} \mathbf{F}_{c(n,l)}^l \tag{3.59}$$

   as a sum of all forces of all adjacent subzones.

3. According to the nodal forces, compute the new velocities

$$\mathbf{w}_n^1 = \mathbf{w}_n + \frac{\Delta t}{m_n} \mathbf{F}_n \tag{3.60}$$

   and apply the velocity boundary conditions to them, and compute the velocities in the half-time

$$\mathbf{w}_n^{1/2} = \frac{1}{2} \left(\mathbf{w}_n + \mathbf{w}_n^1 \right). \tag{3.61}$$

4. Move the nodes to their new positions

$$\mathbf{z}_n = \mathbf{z}_n + \Delta t \ \mathbf{w}_n^{1/2} \tag{3.62}$$

according to the velocities in half-time.

5. Update the geometry according to the new nodal position – compute new zonal and subzonal volumes, centers, and edge centers.

6. Compute the total work done in each cell c due to the forces affecting its nodes

$$E_c^{\text{work}} = -\sum_{l=1}^{4} \mathbf{F}_{n(c,l)} \cdot \mathbf{w}_{n(c,l)}^{1/2} \tag{3.63}$$

and compute the new specific internal energy due this work

$$\epsilon_c = \epsilon_c + \frac{\Delta t}{m_c} E_c^{\text{work}} . \tag{3.64}$$

7. Update cell and subzone densities in new cells as

$$\rho_c = \frac{m_c}{V_c} , \quad \rho_c^l = \frac{m_c^l}{V_c^l} , \tag{3.65}$$

and new cell pressure from the equation of state, and apply the pressure boundary conditions to them.

8. Finally, switch velocities to the new mesh $\mathbf{w}_n = \mathbf{w}_n^1$.

In fact, this algorithm uses the Euler method with step Δt for time integration, which is fast and easy to implement, but only first order accurate. To improve the algorithm, and due to the fact, that we wish all quantities entering the definition of total energy to be defined at the same time level, better time integration scheme should be used. We use the second-order Runge-Kutta method (RK2). The modification is quite easy – at the end of the process, the new quantities (positions $\mathbf{z}_n$ and the geometrical quantities as volumes, centers, ..., energies, zonal and subzonal densities) are transformed to the half-time $\Delta t/2$ by averaging the computed values with the original ones, and the described process is repeated once again to get the new values in the final time Δt. For more details about the time integration issues, see [18].

3.2 Mesh Smoothing Techniques

The second essential part of the ALE algorithm is the method for mesh smoothing and regularization. It regularizes (untangles and smoothes) the computational mesh and produces the new one, used for further calculation. There exist many types of mesh regularization techniques, due to our needs we focus only on the methods smoothing the meshes by node repositioning. There are several approaches for this task – methods based on plain or weighted averaging, or local and global optimization methods. Let us go through several methods for mesh optimization and discuss its properties.

3.2.1 Plain Averaging and Winslow Smoothing

The easiest methods for computational mesh regularization are based on some kind of average calculations. The method based on the plain average calculation can compute the new mesh nodes positions in the following way

$$\tilde{\mathbf{z}}_{i,j} = \frac{1}{8} \left(4\,\mathbf{z}_{i,j} + \mathbf{z}_{i-1,j} + \mathbf{z}_{i+1,j} + \mathbf{z}_{i,j-1} + \mathbf{z}_{i,j+1}\right) . \tag{3.66}$$

where $\mathbf{z}_{i,j} = (x_{i,j}, y_{i,j})$ denotes the nodal position of the node $n = (i, j)$, and the tilde symbol denotes the quantities connected to the new smoothed computational mesh. This simple formula is fast and smoothes the mesh properly. On the other hand, the mesh motion can be too severe, and the new mesh can be too different from the original one.
Another method using principles of averaging for mesh smoothing, is the classical Winslow smoothing introduced in [97] and extended in [91] and [15]. This smoothing scheme results from the solution of the inverted Laplace equation and can be written in the following form

$$\tilde{\mathbf{z}}_{i,j} = \frac{1}{2\,(\alpha+\gamma)} \Big(\alpha\,(\mathbf{z}_{i,j+1} + \mathbf{z}_{i,j-1}) + \gamma\,(\mathbf{z}_{i+1,j} + \mathbf{z}_{i-1,j}) - \frac{1}{2}\,\beta\,(\mathbf{z}_{i+1,j+1} - \mathbf{z}_{i-1,j+1} + \mathbf{z}_{i-1,j-1} - \mathbf{z}_{i+1,j-1}) \Big)\,,$$

where the coefficients α, β, γ are computed as

$$\alpha = x_\xi^2 + y_\xi^2 \tag{3.67}$$

$$\beta = x_\xi\, x_\eta + y_\xi\, y_\eta \tag{3.68}$$

$$\gamma = x_\eta^2 + y_\eta^2\,. \tag{3.69}$$

The derivatives of the nodal positions $\mathbf{z}_\xi$, $\mathbf{z}_\eta$ according to the logical coordinates ξ, η of node $n = (i, j)$ are computed as

$$\mathbf{z}_\xi = \frac{1}{2}\,(\mathbf{z}_{i+1,j} - \mathbf{z}_{i-1,j}) \tag{3.70}$$

$$\mathbf{z}_\eta = \frac{1}{2}\,(\mathbf{z}_{i,j+1} - \mathbf{z}_{i,j-1})\,. \tag{3.71}$$

This movement attempts to preserve mesh orthogonality and is used for mesh smoothing in many ALE codes [79]. Although, this method can produce meshes very different from the original Lagrangian meshes, on the other hand (unlike the plain averaging), the obtained results are reasonable and can be used for real ALE simulations. There exist some special cases, where this method is not advisable and more sophisticated method is necessary, but in general, the Winslow smoothing method can be employed to most ALE simulations.

3.2.2 Advanced Rezoning Methods

There exist many other methods for mesh regularization and smoothing, here we briefly describe two types of them closely connected to the ALE approach – the methods based on the combination of global and local mesh smoothing techniques, and the methods based on the Reference Jacobian smoothing.
Generally, the mesh smoothing techniques can be divided into two groups – global and local optimization methods. We focus here on the local feasible set method and the global method of the numerical optimization of mesh quality functional, and show their combination introduced in [94], which can be used for untangling of hardly distorted computational meshes. The feasible set method is a local method moving the nodal position just with respect to the local neighborhood of the node. Its biggest advantage is the fact, that it moves only the nodes, that are necessary to be moved. On the other hand, it is not guaranteed for a locally hardly distorted meshes that the local method will help. For each mesh node, its feasible set is constructed [78], which is the polygon, into which one can move the node to locally correct the problematic node and make all cells around the node valid (with positive subzonal volumes). The numerical functional optimization method is a global method moving all the mesh nodes at once. The main advantage is producing a valid mesh. On the other hand it moves all mesh nodes, also in smooth mesh regions, and thus increases the numerical error of the coming remapping stage. At first, a functional describing mesh quality must be constructed (for examples see [94]), which is then minimized by some optimization method, such as conjugate gradient method [77]. The combination

of both methods, introduced in [94], combines positives of both approaches. At first, the feasible set method locally correcting most of the problematic nodes is applied. The following global numerical optimization corrects all problems, but zero-volume cells can appear. Due to the previous feasible set step, the global mesh movement in the second step is not so strong. Then, the third step is applied – local feasible set method again, extending possible degenerated zero-volume cells. This 3-step method achieves acceptable mesh untangling while keeping the mesh close to the original one. It is suitable for sporadic usage in the case that computational mesh degenerates, but not for regular usage for mostly regular meshes. The best strategy in the ALE process is to keep the mesh smooth during the whole computation and do not allow such hardly degenerated meshes.
The last approach for mesh smoothing which is discussed in this study, is the reference Jacobian method introduced in [48], and extended and described in more details in [32]. It constructs reference positions of each node by local mesh quality functional (based on the local Jacobi determinants) minimization. Then, the reference Jacobi matrix depending on the reference nodal positions is constructed in each subzone. A global mesh quality functional based on the Jacobi matrices is constructed and minimized, which provides the new nodal positions for the new computational mesh. This final mesh is closer to the original Lagrangian mesh than the reference one, and can be directly applied in the ALE simulations.

3.3 Conservative Remapping Algorithm

The last essential part of the ALE algorithm is the conservative interpolation of all quantities from the Lagrangian computational mesh, to the new, smoothed one. We require the remapping method to be linearity-preserving (this condition seems in practical tests to imply second order of accuracy), conservative for all conservative quantities, and local-bound preserving (the method should not create new local extrema in any of the primitive quantities). Our approach results from [68, 70], and reduces the problem of remapping all conservative quantities to the problem of remapping of each of them by a single process, while satisfying the named properties for all these quantities.
In the next subsections, we describe the algorithm for remapping of arbitrary function (density of a conservative quantity) between computational meshes. In subsection 3.3.6, we go through the complete algorithm for remapping of all quantities, where we use the described remapping algorithm for each single conservative quantity. Then, in subsection 3.3.7, we demonstrate the properties of the remapping method on several numerical examples. Finally, for future use, we extend the idea of the remapping algorithm to 3D and to the case when the computational mesh changes its connectivity during the smoothing stage. This will allow the future development of the changing-topology and a 3D ALE codes.

3.3.1 Remapping of One Conservative Quantity

We have two computational meshes – the old one from the last Lagrangian step $\{c\}$, and the new one coming from the mesh smoothing process $\{\tilde{c}\}$. We will denote all quantities related to the new mesh by the tilde accent. Suppose, we have an arbitrary function $g(\mathbf{z})$, $\mathbf{z} = (x, y)$ defined in the computational domain. The function can be arbitrary. In the context of ALE methods, this function represents the density of mass $g = \rho$, density of momentum $g = \rho\, u$, $g = \rho\, v$, or density of total energy $g = \rho\,(\epsilon + |\mathbf{w}|^2/2)$. We do not know the explicit formula for the function evaluation, we know only the mean values of this function in the cells of the original computational mesh g_c. These values correspond to the integrals

$$g_c = \frac{\int_c g(x,y)\,dx\,dy}{V_c} \tag{3.72}$$

and by the definition of cell mass (or cell momentum or total cell energy),

$$g_c = \frac{G_c}{V_c}\,. \tag{3.73}$$

We call the integral of the function g over the cell as the mass of the function g (or cell mass) in cell c and denote it G_c (in the case $g(\mathbf{z}) = \rho(\mathbf{z})$, it corresponds to the real cell mass $G_c = m_c$). The total mass of the whole computational domain Ω corresponds to

$$G_\Omega = \int_\Omega g(\mathbf{z})\, dV = \sum_{\forall c} \int_c g(\mathbf{z})\, dV = \sum_{\forall c} G_c\,. \tag{3.74}$$

Our task is to compute the new mean values in the new mesh cells $g_{\tilde{c}} = G_{\tilde{c}}/V_{\tilde{c}}$. And we want to satisfy the previously mentioned conditions of good remapping method:

1. Accuracy – we want the new mean values and masses to be as close to the exact ones, as possible

$$G_{\tilde{c}} \approx G_{\tilde{c}}^{\text{exact}} = \int_{\tilde{c}} g(\mathbf{z})\, dV\,. \tag{3.75}$$

2. Conservativity – the new total mass must be the same, as the original one

$$\sum_{\forall \tilde{c}} G_{\tilde{c}} = G_\Omega\,. \tag{3.76}$$

3. Local-bound preservation – we require the new mean values not to create the new local extrema, they must remain in the original local bounds

$$g_c^{\max} \geq g_{\tilde{c}} \geq g_c^{\min}\,, \quad g_c^{\max} = \max_{c' \in C(c)} g_{c'}\,, \quad g_c^{\min} = \min_{c' \in C(c)} g_{c'} \tag{3.77}$$

   over the neighborhood $C(c)$ (patch of 3×3 cells) of the original cell c.

4. Linearity preservation – in the case, that the density function g corresponds to the global linear function, the new mean values and masses must be exact

$$G_{\tilde{c}} = G_{\tilde{c}}^{\text{exact}} = \int_{\tilde{c}} g(\mathbf{z})\, dV\,, \text{ if } g(\mathbf{z}) = a + b\,x + c\,y\,. \tag{3.78}$$

Our remapping method is based on the algorithm introduced in [71], [72] and extended in [58]. It consists of three parts:

1. Piecewise linear reconstruction inside old cells.
2. Integration (exact or approximate).
3. Repair.

In the first stage, the unknown function g is approximated by the piecewise linear function, which is exactly equal to the mean values g_c in the cell centers (centroids), and linear inside each old cell. In the second stage, the reconstructed function is integrated over the new cells $\tilde{c}$, which gives us the new cell masses $G_{\tilde{c}}$, and thus new mean values $g_{\tilde{c}}$. The most natural approach is the exact integration – direct integration of the reconstructed function over the new cell $\tilde{c}$. Unfortunately, it requires to find all the intersections of both meshes, which is computationally expensive in 2D. In 3D, this approach is almost uncodable and uncomputable. Thus, we have developed an approximate integration method – swept integration, which does not require any intersections, and is easily generalizable to 3D. On the other hand, this method is approximate, and it may happen, that the new mean values violate the local-bound preservation condition (3.77). Therefore, we have added the third stage – the repair, which corrects this problem and conservatively returns the values back to the local bounds. Finally, our algorithm satisfies all the conditions required for a high-quality remapping method stated above. Let us go through all the remapping stages.

3.3.2 Piecewise Linear Reconstruction

Piecewise linear reconstruction is the first stage of the remapping process. Its task is to compute the slopes of the unknown function g in each cell c. Suppose that the reconstructed function in cell c with cell center (x_c, y_c) has the form

$$g_c(x,y) = g_c + \left(\frac{\partial g}{\partial x}\right)_c (x - x_c) + \left(\frac{\partial g}{\partial y}\right)_c (y - y_c)\,. \tag{3.79}$$

To respect the linearity-preservation condition of the remapping algorithm, the reconstruction stage itself must be linearity-preserving.
There are several methods for function reconstruction. We discuss here three of them, the average difference method, the least-squares minimization process, and finally the Barth-Jespersen limiting process to enforce monotonicity preservation of the reconstructed function.

Average Derivative in Neighborhood The most natural approach for the unlimited slopes approximation, is averaging of the function derivative over the region $V_{i,j}^{\text{neigh}}$ defined by connecting the centers of the adjacent neighbors. The region is shown in Figure 3.4.

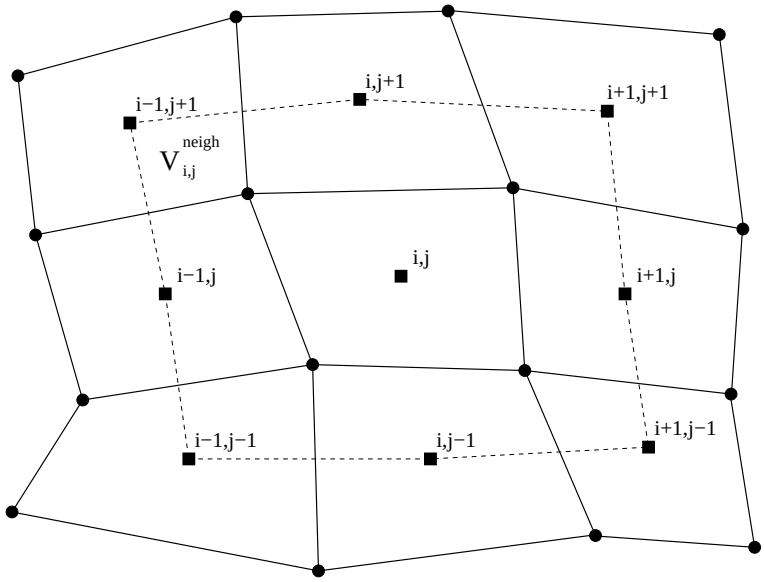


Figure 3.4: Patch of 3×3 cells and the region of integration $V_{i,j}^{\text{neigh}}$ (in dashed lines) for the approximation of the average derivative in cell i, j.

The unlimited slope approximation in cell $c = i, j$ can be written in the form

$$\left(\frac{\partial g}{\partial x}\right)_{i,j}^{\text{unlim}} = \frac{\int_{V_{i,j}^{\text{neigh}}} \frac{\partial g}{\partial x}\, dx\, dy}{\int_{V_{i,j}^{\text{neigh}}} 1\, dx\, dy}\,. \tag{3.80}$$

The denominator is equal to the neighborhood volume, and the numerator can be reduced by using the Green theorem to the boundary integral

$$\left(\frac{\partial g}{\partial x}\right)_{i,j}^{\text{unlim}} = \frac{\oint_{\partial V_{i,j}^{\text{neigh}}} g\, dy}{|V_{i,j}^{\text{neigh}}|}\,. \tag{3.81}$$

After defining the function value along each edge of the neighborhood by the average of the corresponding cell values, one can rewrite the whole formula to the form

$$\begin{aligned}\left(\frac{\partial g}{\partial x}\right)_{i,j}^{\text{unlim}} &= \frac{1}{|V_{i,j}^{\text{neigh}}|} \sum_{\forall e \in \partial V_{i,j}^{\text{neigh}}} \frac{1}{2} \left(g_{c(e,-)} + g_{c(e,+)}\right) \int_e dy \\ &= \frac{1}{|V_{i,j}^{\text{neigh}}|} \frac{1}{2} \sum_{\forall e \in \partial V_{i,j}^{\text{neigh}}} \left(g_{c(e,-)} + g_{c(e,+)}\right) \left(y_{c(e,+)} - y_{c(e,-)}\right) ,\end{aligned} \tag{3.82}$$

where the orientation of the neighborhood boundary edges is assumed in the counter-clock wise order. This formula can be explicitly expressed for the situation shown in Figure 3.4 as

$$\begin{aligned}\left(\frac{\partial g}{\partial x}\right)_{i,j}^{\text{unlim}} = \frac{1}{|V_{i,j}^{\text{neigh}}|} \frac{1}{2} \Big(&\left(g_{i-1,j-1} + g_{i,j-1}\right)\left(y_{i,j-1} - y_{i-1,j-1}\right) + \left(g_{i,j-1} + g_{i+1,j-1}\right)\left(y_{i+1,j-1} - y_{i,j-1}\right) + \\ &\left(g_{i+1,j-1} + g_{i+1,j}\right)\left(y_{i+1,j} - y_{i+1,j-1}\right) + \left(g_{i+1,j} + g_{i+1,j+1}\right)\left(y_{i+1,j+1} - y_{i+1,j}\right) + \\ &\left(g_{i+1,j+1} + g_{i,j+1}\right)\left(y_{i,j+1} - y_{i+1,j+1}\right) + \left(g_{i,j+1} + g_{i-1,j+1}\right)\left(y_{i-1,j+1} - y_{i,j+1}\right) + \\ &\left(g_{i-1,j+1} + g_{i-1,j}\right)\left(y_{i-1,j} - y_{i-1,j+1}\right) + \left(g_{i-1,j} + g_{i-1,j-1}\right)\left(y_{i-1,j-1} - y_{i-1,j}\right)\Big) ,\end{aligned} \tag{3.83}$$

where the symbols $y_{i\pm1,j\pm1}$ in this context denote the y coordinates of the centers of the corresponding cells. Here, the neighborhood volume $|V_{i,j}^{\text{neigh}}|$ is computed as the sum of four quadrilateral regions defined by connecting the neighboring cell centers with the center of the central cell i, j. The formula for the y slope $(\partial g/\partial y)_{i,j}^{\text{unlim}}$ is derived analogically and has the same form, just a minus sign coming from the Green theorem appears in front of it.

This method is straightforward and provides reasonable approximation for the function slopes in each cell. For more details about this approach, see [84], [71]. This method can be used in 2D, logically rectangular meshes. Unfortunately, this method is hardly generalizable to 3D, and several complications appear also for the case of general polyhedral cells. Thus, for future generalization of the code, we present here also another method based on minimization of the slope error functional, which is generalizable to the mentioned, more complicated cases.

Least-squares Minimization Process The second method is the least-squares method minimization of the error functional

$$F\left(\left(\frac{\partial g}{\partial x}\right)_c^{\text{unlim}}, \left(\frac{\partial g}{\partial y}\right)_c^{\text{unlim}}\right) = \sum_{c' \in C(c)} \left(g_{c'} - \frac{\int_{c'} g_c^{\text{unlim}}(x,y)\, dx\, dy}{V_c}\right)^2 \tag{3.84}$$

defining the measure of the difference of the neighboring mean values from the (unlimited) reconstructed values in the centers of the neighboring cells.

Now, let us find the minimum of this functional (which means to set the reconstructed function as close to the exact function as possible). Let us differentiate this functional according to its parameters and set it to zero. We present here the process of differentiating the functional according to the first parameter,

$(\partial g/\partial x)_c^{\text{unlim}}$, the second one is analogical. So we solve the following equation

$$0 = \frac{\partial F\left(\left(\frac{\partial g}{\partial x}\right)_c^{\text{unlim}}, \left(\frac{\partial g}{\partial y}\right)_c^{\text{unlim}}\right)}{\partial\left(\frac{\partial g}{\partial x}\right)_c^{\text{unlim}}} = \frac{\partial\left(\sum_{c'\in C(c)}\left(g_{c'} - \frac{\int_{c'} g_c^{\text{unlim}}(x,y)\,dx\,dy}{V_{c'}}\right)^2\right)}{\partial\left(\frac{\partial g}{\partial x}\right)_c^{\text{unlim}}} \tag{3.85}$$

$$= \sum_{c'\in C(c)} 2\left(g_{c'} - \frac{\int_{c'} g_c + \left(\frac{\partial g}{\partial x}\right)_c^{\text{unlim}}(x-x_c) + \left(\frac{\partial g}{\partial y}\right)_c^{\text{unlim}}(y-y_c)\,dx\,dy}{V_{c'}}\right)\left(-\frac{\int_{c'}(x-x_c)\,dx\,dy}{V_{c'}}\right) \tag{3.86}$$

and by substituting the definitions of cell volumes (3.3) and cell centers (3.4), (3.5), we get the following expression

$$0 = \sum_{c'\in C(c)} 2\left(g_{c'} - g_c - \left(\frac{\partial g}{\partial x}\right)_c^{\text{unlim}}(x_{c'} - x_c) - \left(\frac{\partial g}{\partial y}\right)_c^{\text{unlim}}(y_{c'} - y_c)\right)\left(-(x_{c'} - x_c)\right). \tag{3.87}$$

This equation can be rewritten into the linear form

$$0 = -b_x + a_{xx}\left(\frac{\partial g}{\partial x}\right)_c^{\text{unlim}} + a_{xy}\left(\frac{\partial g}{\partial y}\right)_c^{\text{unlim}}. \tag{3.88}$$

After repeating the same process for the y derivative, one can write the complete system as

$$\mathbb{A}\left(\frac{\partial g}{\partial \mathbf{z}}\right)_c = \mathbf{b} \tag{3.89}$$

or explicitly

$$\begin{pmatrix} a_{xx} & a_{xy} \\ a_{xy} & a_{yy} \end{pmatrix} \cdot \begin{pmatrix} \left(\frac{\partial g}{\partial x}\right)_c^{\text{unlim}} \\ \left(\frac{\partial g}{\partial y}\right)_c^{\text{unlim}} \end{pmatrix} = \begin{pmatrix} b_x \\ b_y \end{pmatrix} \tag{3.90}$$

for unknown partial derivatives, where

$$a_{\alpha\beta} = 2\sum_{c'\in C(c)}(\alpha_{c'} - \alpha_c)(\beta_{c'} - \beta_c) \tag{3.91a}$$

$$b_\alpha = 2\sum_{c'\in C(c)}(\alpha_{c'} - \alpha_c)(g_{c'} - g_c) \tag{3.91b}$$

for $\alpha, \beta \in \{x, y\}$. Solution of this linear system (3.90) is computed by direct method based on the inverse matrix calculation. The final solution is then

$$\begin{pmatrix} \left(\frac{\partial g}{\partial x}\right)_c^{\text{unlim}} \\ \left(\frac{\partial g}{\partial y}\right)_c^{\text{unlim}} \end{pmatrix} = \frac{1}{D}\begin{pmatrix} b_x\,a_{yy} - b_y\,a_{xy} \\ b_y\,a_{xx} - b_y\,a_{xy} \end{pmatrix}, \tag{3.92}$$

where the determinant of the matrix

$$D = \det(\mathbb{A}) = a_{xx}\,a_{yy} - a_{xy}^2 \tag{3.93}$$

also serves as an indicator of regularity of cell c. In some severe fluid motion types it may happen, that the particular cell degenerates to a line or a point (not exactly, but approximately). This causes the determinant D to be close to zero and this situation must be treated specially.

Barth-Jespersen Limiting Process Finally, to minimize the number of overshoots in the remapped values, we limit the reconstructed function. We have tested several most commonly used limiters [50], and the Barth-Jespersen (BJ) approach [8, 7] seems to us to be the most suitable for the reconstruction task. This limiter is constructed such that it preserves linear function, but it does not allow any overshoots in the reconstructed function. The cellular BJ limiter is defined as

$$\Phi_c = \min_{n \in N(c)} \Phi_c^n \,, \tag{3.94}$$

where the nodal limiters are defined as

$$\Phi_c^n = \begin{cases} \min\left(1, \frac{g_c^{\max} - g_c}{g_c^{\mathrm{unlim}}(n) - g_c}\right) & \text{for } g_c^{\mathrm{unlim}}(n) - g_c > 0 \\ \min\left(1, \frac{g_c^{\min} - g_c}{g_c^{\mathrm{unlim}}(n) - g_c}\right) & \text{for } g_c^{\mathrm{unlim}}(n) - g_c < 0 \\ 1 & \text{for } g_c^{\mathrm{unlim}}(n) - g_c = 0 \,. \end{cases} \tag{3.95}$$

Here $g_c^{\mathrm{unlim}}(n)$ is the value of the reconstructed function (3.79) computed with the unlimited slopes $(\partial g/\partial x, y)_c^{\mathrm{unlim}}$ and evaluated in the nodal position n. The local extrema $g_c^{\min}$, $g_c^{\max}$ are computed as the extrema of the mean values over the neighboring cells (3×3 patch of cells around the cell c) of the particular cell. The computation of the unlimited slopes has been presented in the previous paragraphs. The final slopes used for the numerical integration are then computed as

$$\left(\frac{\partial g}{\partial x}\right)_c = \Phi_c \left(\frac{\partial g}{\partial x}\right)_c^{\mathrm{unlim}} \,, \quad \left(\frac{\partial g}{\partial y}\right)_c = \Phi_c \left(\frac{\partial g}{\partial y}\right)_c^{\mathrm{unlim}} \,. \tag{3.96}$$

These slopes guarantee the global linearity and monotonicity preservation of the reconstructed function according to the original mean values g_c.
Formula (3.92) gives us the unlimited slopes, which is together with the limitation process (3.96) used for the final slopes evaluation. The main advantage of this approach is the fact that this method is generalizable to 3D meshes, and to meshes with the changing connectivity. As for the numerical errors of the final solution, there is no big difference between the unlimited method minimizing the error functional, and the previous unlimited method of average derivative in the neighborhood, resulting in formula (3.83).

3.3.3 Exact Numerical Integration

The first approach for getting the new density function masses and their mean values, is the exact integration of the reconstructed function over new cells. This method analytically integrates the reconstructed function over all overlapping elements of both meshes, which gives us the masses of these cells intersections. By summing the masses corresponding to some particular new cell, we get the new cell mass, and after dividing by the new volume, the new density mean value. Let us go through the method in more details.
At first suppose, that we have some reconstruction $g_c(x, y)$ as described in (3.79) in each cell c, obtained by some reconstruction method described in the previous section. Now, we go through the patch of the original mesh fully covering the new cell $\tilde{c}$. Usually, it is practical to suppose, that the mesh movement was not to severe during the mesh smoothing stage, and one can use the original cell c and its nearest neighbors $C(c)$. The situation is shown in Figure 3.5. We go through all cells in this patch $c' \in C(c)$, and compute the intersection of the cell c' with the new cell $\tilde{c}$. The examples of such intersection of the new cell with the neighborhood of the corresponding original cell, are shown in different shades of gray. Each intersection is a polygon, which can have arbitrary number of vertices from 0 (if the cell does not intersect the new one, for example cell $c' = i - 1, j + 1$ in Figure 3.5) up to 8 (this may happen, when the original cell rotates a little). For computation of this intersection region, a robust and precise intersection method is required. We use an intersection method described in [78], which is based on the

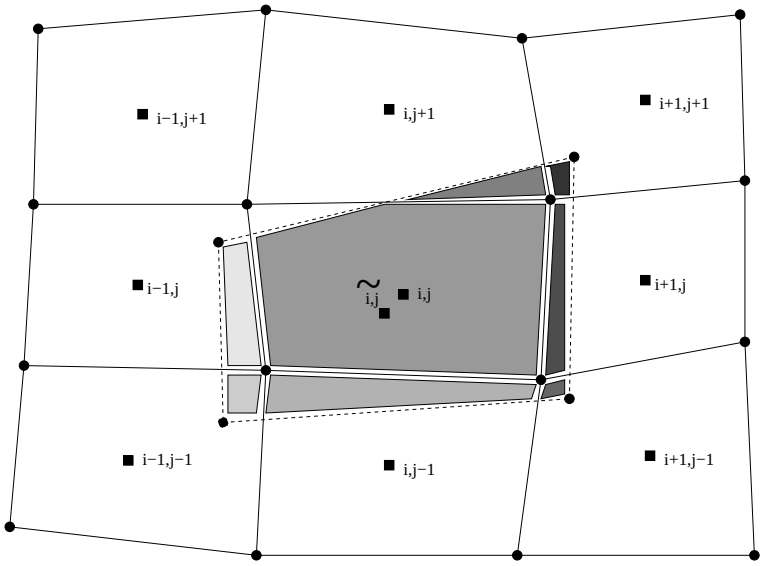


Figure 3.5: Original cell $c = i, j$ (solid lines) and its neighborhood $C(c)$, and the corresponding new cell $\tilde{c} = \widetilde{i, j}$ (dashed lines). All segments of intersection of the new cell with the original mesh are shown in different shades of gray.

computation of the intersections of all correctly oriented halfplanes assigned to all edges of both (old and new) cells. Some details about the exact integration approach are included in [71].
Suppose, we have computed an intersection of the old cell c and the new cell $\tilde{c}'$. We denote this polygon by the symbol $P_c^{\tilde{c}'}$. To compute the mass in this intersection region, we integrate the reconstructed function of the original cell $g_c(x, y)$ over this region

$$G_{P_c^{\tilde{c}'}} = \int_{P_c^{\tilde{c}'}} g_c(x, y)\, dx\, dy = \int_{P_c^{\tilde{c}'}} \left(g_c + \left(\frac{\partial g}{\partial x}\right)_c (x - x_c) + \left(\frac{\partial g}{\partial y}\right)_c (y - y_c) \right) dx\, dy\,, \tag{3.97}$$

which can be rewritten in the form

$$\begin{aligned} G_{P_c^{\tilde{c}'}} = g_c \int_{P_c^{\tilde{c}'}} 1\, dx\, dy + \left(\frac{\partial g}{\partial x}\right)_c \left(\int_{P_c^{\tilde{c}'}} x\, dx\, dy - x_c \int_{P_c^{\tilde{c}'}} 1\, dx\, dy \right) \\ + \left(\frac{\partial g}{\partial y}\right)_c \left(\int_{P_c^{\tilde{c}'}} y\, dx\, dy - y_c \int_{P_c^{\tilde{c}'}} 1\, dx\, dy \right) . \end{aligned} \tag{3.98}$$

Here, the integrals of 1, x, and y over the polygon can be evaluated analogically, as in cell volume and cell centers definitions, as we did in section 3.1.1. Using the Green theorem, they can explicitly be expressed from the intersection region vertices positions. Let us show the process on the example of the volume computation (integral of 1). The integral can be reduced to the boundary integral, which can be written as a sum of edge integrals

$$\int_{P_c^{\tilde{c}'}} 1\, dx\, dy = \oint_{\partial P_c^{\tilde{c}'}} x\, dy = \sum_{e \in \partial P_c^{\tilde{c}'}} \int_e x\, dy\,. \tag{3.99}$$

Now, suppose, that the particular edge $e = [e_1, e_2]$ with vertices $e_1 = (x_1, y_1)$ and $e_2 = (x_2, y_2)$ has the following equation

$$x = x_1 + \frac{x_2 - x_1}{y_2 - y_1}\, (y - y_1)\,, \tag{3.100}$$

which can be substituted into the integral and evaluated

$$\int_e x\, dy = x_1\, (y_2 - y_1) + \frac{x_2 - x_1}{y_2 - y_1}\, \frac{y_2^2 - y_1^2}{2} - \frac{x_2 - x_1}{y_2 - y_1}\, y_1\, (y_2 - y_1) \tag{3.101}$$

or after some algebra

$$\int_e x\,dy = \frac{1}{2}\,(x_1 + x_2)\,(y_2 - y_1)\,. \tag{3.102}$$

The situation of $y_1 = y_2$ is treated by analogical process with exchanged x and y coordinates in the edge equation (3.100). The final formula is exactly the same as formula (3.102), in which is the problematic $y_2 - y_1$ denominator missing, and can be used for arbitrary edge e. The other integrals are evaluated exactly the same way, and can be written as

$$\int_{P_c^{\tilde{c}'}} 1\,dx\,dy = \oint_{\partial P_c^{\tilde{c}'}} x\,dy = \sum_{e\in\partial P_c^{\tilde{c}'}} \frac{1}{2}\,(x_1 + x_2)\,(y_2 - y_1) \tag{3.103}$$

$$\int_{P_c^{\tilde{c}'}} x\,dx\,dy = \oint_{\partial P_c^{\tilde{c}'}} x^2\,dy = \sum_{e\in\partial P_c^{\tilde{c}'}} \frac{1}{6}\,\left(x_1^2 + x_1\,x_2 + x_2^2\right)\,(y_2 - y_1) \tag{3.104}$$

$$\int_{P_c^{\tilde{c}'}} y\,dx\,dy = -\oint_{\partial P_c^{\tilde{c}'}} y^2\,dx = -\sum_{e\in\partial P_c^{\tilde{c}'}} \frac{1}{6}\,\left(y_1^2 + y_1\,y_2 + y_2^2\right)\,(x_2 - x_1)\,. \tag{3.105}$$

This process gives us the final explicit formula for the mass of the intersection region $G_{P_c^{\tilde{c}'}}$ (3.98), and by summing them, we get the final formula for new cell mass and new cell density

$$G_{\tilde{c}} = \sum_{c'\in C(c)} G_{P_{c'}^{\tilde{c}}} \tag{3.106}$$

$$g_{\tilde{c}} = \frac{G_{\tilde{c}}}{V_{\tilde{c}}}\,. \tag{3.107}$$

By summing the intersections volumes (3.103), one can check, whether some part of the new cell $\tilde{c}$ was not forgotten to be included (the sum of the intersection volumes must in some positive ε range correspond to the new volume $V_{\tilde{c}}$). If we forget some piece, one can extend the intersection patch (level 2, 3, ... neighborhood), compute the intersection of the new cell $\tilde{c}$ with the larger patch, until we cover the complete cell $\tilde{c}$. This situation may occur, if the cell changes significantly during the smoothing process.
This method is applicable to our remapping problem and is conservative

$$\begin{aligned}\sum_{\forall\tilde{c}} G_{\tilde{c}} &= \sum_{\forall\tilde{c}} \left(\sum_{c'\in C(c)} G_{P_{c'}^{\tilde{c}}}\right) = \sum_{\forall\tilde{c}} \left(\sum_{\forall c} G_{P_c^{\tilde{c}}}\right) \\ &= \sum_{\forall c} \left(\sum_{\forall\tilde{c}} G_{P_c^{\tilde{c}}}\right) = \sum_{\forall c} \left(\sum_{\tilde{c}'\in C(\tilde{c})} G_{P_{c'}^{\tilde{c}}}\right) = \sum_{\forall c} G_c\,.\end{aligned} \tag{3.108}$$

Unfortunately, this method is computationally expensive due to the need to compute cells intersections. Moreover, this algorithm for cells intersections appears to have problems in cases, that the original and the new cell are almost the same, and their nodes differ just a little (which happens very often during smoothing in already smooth regions). This algorithm for cells intersections works only for convex cells. Another disadvantage of this method is its complicated generalization to 3D, where planar faces are not guaranteed. Although the method is applicable in 3D [38], the method is unfeasibly slow. Thus, we are forced to develop more sophisticated method, which does not require the intersections computation.

3.3.4 Approximate Swept Numerical Integration

This approach was developed to replace the exact integration method described in the previous section. This method does not need the computation of the intersections of the original and the smoothed meshes,

which makes this method much more efficient. This method was introduced in [72] and extended for full usage in [58]. Let us go through this method in more details.
Again, we suppose that we know some reconstruction $g_c(x, y)$ (3.79) in each old cell c, coming from some reconstruction method described in section 3.3.2. The swept-region integration method is based on the integration of the reconstructed function over swept regions, defined by the smooth movement of the cell edges from their old to the new positions. One cell with its swept regions is shown in Figure 3.6. Each quadrilateral cell has four swept regions corresponding to their four edges. Each swept region

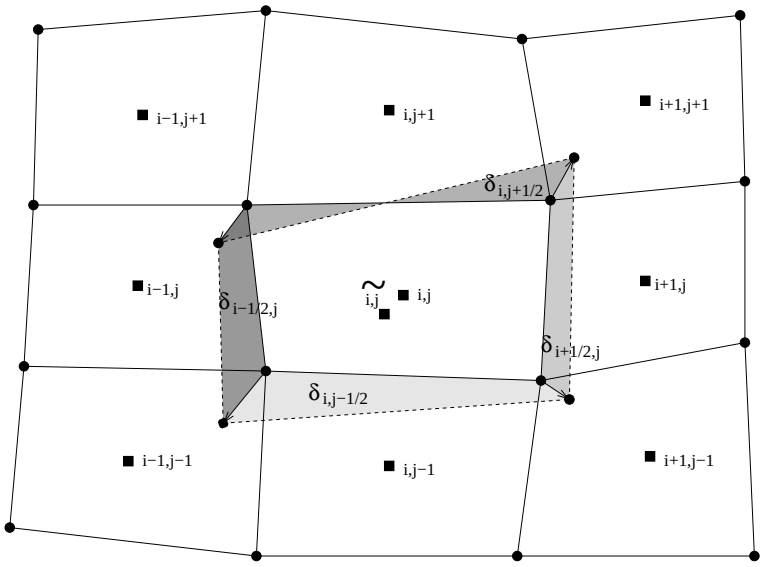


Figure 3.6: Original cell $c = i, j$ (solid lines) and its neighborhood $C(c)$, and the corresponding new cell $\tilde{c} = \widetilde{i, j}$ (dashed lines). Four swept regions are shown in different shades of gray.

is a quadrilateral region surrounded by the original edge, the corresponding new edge, and the lines connecting the corresponding old and new vertices.
The swept-region algorithm is based on the simple idea, that the mass of the new cell can be composed from the original cell mass and masses of all four swept regions

$$G_{\tilde{c}} = G_c + \sum_{e \in E(c)} \delta G_e \,, \tag{3.109}$$

or explicitly for the cells enumerated as in Figure 3.6,

$$G_{\widetilde{i,j}} = G_{i,j} + \delta G_{i-\frac{1}{2},j} + \delta G_{i+\frac{1}{2},j} + \delta G_{i,j-\frac{1}{2}} + \delta G_{i,j+\frac{1}{2}} \,. \tag{3.110}$$

Analogical equation can be written for the new cell volume. When using swept-region masses or swept-region volumes (or shortly swept masses, swept volumes), we always mean them in signed sense. By definition, if we are in a particular cell, the corresponding swept regions have positive volumes, if they move outwards from the cell, and negative, if moving inwards. The swept volume V_{δ_e} is computed using the same formula as (3.103), from the vertices of the swept region.
From the sign of each swept volume we detect, whether the corresponding edge moves inwards or outwards of the original cell. This approach naturally handles twisted swept regions (as the upper swept region $\delta_{i,j+1/2}$ in Figure 3.6), which may occur in case, that one edge vertex moves inwards and second one outwards of the cell. In this case, the sign of the swept volume corresponds to its larger segment, which overweights the smaller one. For computation of the swept mass, we integrate reconstruction from the cell, in which more of the region lies

$$c^* = \begin{cases} c & \text{for} \quad V_{\delta_e} \leq 0 \\ c' & \text{for} \quad V_{\delta_e} > 0 \,, \end{cases} \tag{3.111}$$

where the cell c' here is a neighbor of cell c, which have a common edge e with c. The integration is straightforward,

$$\begin{aligned}\delta G_e &= \int_{\delta_e} g_{c^*}(x,y)\,dx\,dy \\ &= \left(g_{c^*} - \left(\frac{\partial g}{\partial x}\right)_{c^*} x_{c^*} - \left(\frac{\partial g}{\partial y}\right)_{c^*} y_{c^*}\right) \int_{\delta_e} 1\,dx\,dy \\ &+ \left(\frac{\partial g}{\partial x}\right)_{c^*} \int_{\delta_e} x\,dx\,dy + \left(\frac{\partial g}{\partial y}\right)_{c^*} \int_{\delta_e} y\,dx\,dy\,,\end{aligned} \tag{3.112}$$

where the integrals are computed the same way as shown in equations (3.103), (3.104), (3.105).
Finally, we add this swept mass to the mass of the particular cell, and subtract it from the corresponding neighbor. Due to this edge-based nature of the algorithm, each swept volume and swept mass can be computed only once, and used twice for both adjacent cells, which increases its efficiency. Moreover, due to this fact, the algorithm is naturally conservative – the cell masses are interchanged between neighboring cells, what is added in one cell, is subtracted from other one. As for linearity preservation, this algorithm exactly integrates the linear reconstructed function (in case of global linear function, the reconstruction is also a global linear function), so the produced new masses (and thus new densities) are also exact. So the swept integration method preserves global linear function as well. This property, together with the smooth underlying function g, implies the second order of accuracy [72].
The swept region integration is an approximate method. In the regions of significant changes of the remapped quantities (typically, the regions close to shock waves), it may happen, that the new density value overjumps the local extrema of the original mean values. Thus, one more step must follow, to enforce the local-bound preservation condition.

3.3.5 Repair

Repair is the last stage of the remapping process, which corrects the possible local-bound violations caused by the integration process. This method can be described as the conservative redistribution of the conservative quantity. It was introduced in [58] and extended in [87], where its application to advection equation is shown.
The first step of the repair process should be finding the bound-determining neighborhood of cell c, the piece of the original mesh fully covering the new cell $\tilde{c}$, and computing local extrema of the underlying function from the original mean values in this bound-determining neighborhood. Finding the real bound-determining neighborhood would require computations of the intersections of the original and new mesh, which we want to avoid. In practical implementation, we use the same local extremes, which we computed during reconstruction limiting process in section 3.3.2. These extrema are computed over the cell neighborhood $C(c)$ (3×3 patch of cells), which covers the whole bound-determining neighborhood for reasonably strong mesh movement.
So, in each cell c, we know the original mean value g_c, the new mean value $g_{\tilde{c}}$, and the local extremes

$$g_c^{\min} = \min_{c' \in C(c)} g_{c'} \tag{3.113}$$

$$g_c^{\max} = \max_{c' \in C(c)} g_{c'}\,. \tag{3.114}$$

If the new mean value lies within the local extremes

$$g_c^{\min} \le g_{\tilde{c}} \le g_c^{\max}\,, \tag{3.115}$$

the condition is satisfied and no repair process is required. Suppose, that the local minimum is violated

$$g_{\tilde{c}} \le g_c^{\min}\,, \tag{3.116}$$

maximum case is treated analogically. At first, we compute the mass, which is missing in the new cell to bring the value back to the local minimum

$$\delta G_{\tilde{c}}^{\text{need}} = \left(g_c^{\min} - g_{\tilde{c}}\right) V(\tilde{c})\,. \tag{3.117}$$

The algorithm has to be conservative, so we cannot just add the mass to the cell $\tilde{c}$, we have to subtract it from the mass of the neighborhood. For each neighboring cell, we compute the mass

$$\delta G_{\tilde{c}'}^{\text{avail}} = \max\left(0, (g_{\tilde{c}'} - g_{c'}^{\min})\, V(\tilde{c}')\right)\,, \tag{3.118}$$

which can safely be taken from it without violating its local minimum. Then, the total available mass in the neighborhood is their sum

$$\delta G_{C(\tilde{c})}^{\text{avail}} = \sum_{c' \in C(c)} \delta G_{\tilde{c}'}^{\text{avail}}\,. \tag{3.119}$$

If the available mass is smaller than the needed mass

$$\delta G_{C(\tilde{c})}^{\text{avail}} < \delta G_{\tilde{c}}^{\text{need}}\,, \tag{3.120}$$

we have to extend the neighborhood stencil and search for the mass in a larger region. If we found enough mass

$$\delta G_{C(\tilde{c})}^{\text{avail}} \geq \delta G_{\tilde{c}}^{\text{need}}\,, \tag{3.121}$$

we can perform the repair process. We increase the wrong new value up to the local minimum

$$G_{\tilde{c}} = g_c^{\min}\, V(\tilde{c})\,, \tag{3.122}$$

and take the mass, which was just added to it, from the neighboring cells proportionally to their available masses

$$G_{\tilde{c}'} = G_{\tilde{c}'} - \frac{\delta G_{\tilde{c}'}^{\text{avail}}}{\delta G_{C(\tilde{c})}^{\text{avail}}}\, \delta G_{\tilde{c}}^{\text{need}} \tag{3.123}$$

from all cells $\tilde{c}'$ from the neighborhood $C(\tilde{c})$. This method repairs all local-bound violations coming from the integration process.

In [58], we have also proved, that this process successfully finishes the repair in a finite number of steps. It is not obvious, that the repair algorithm always finds enough mass in some (maybe larger) neighborhood, so let us go through the proof to clear it here. Suppose, that the new cell $\tilde{c}$ is fully covered by the neighborhood $C(c)$ of the original cell c,

$$\tilde{c} \cap c' \neq \emptyset \text{ if and only if } c' \in C(c)\,, \tag{3.124}$$

and suppose, that the local minimum is violated in cell c (as in equation (3.116)). Let us define two quantities,

$$\Delta M^- = \sum_{\tilde{c}' : g_{\tilde{c}'} < g_{c'}^{\min}} \left(g_{c'}^{\min} - g_{\tilde{c}'}\right) V(\tilde{c}') \tag{3.125}$$

$$\Delta M^+ = \sum_{\tilde{c}' : g_{\tilde{c}'} > g_{c'}^{\min}} \left(g_{\tilde{c}'} - g_{c'}^{\min}\right) V(\tilde{c}')\,. \tag{3.126}$$

Here, $\Delta M^- > 0$ is the total amount of mass, that is needed to bring all new densities in cells c', where they are lower that their local minima $g_{c'}^{\min}$, up to these local minima. On the other hand, $\Delta M^+ > 0$ is the total amount of mass, that can safely be taken from all other cells without violating their local minima. To show, that the repair process is successful, we have to prove the following relation

$$\Delta M^- \leq \Delta M^+\,. \tag{3.127}$$

Let us write the total mass in the entire mesh as

$$\begin{aligned} M &= \sum_{\forall \tilde{c}} g_{\tilde{c}}\, V_{\tilde{c}} \\ &= \sum_{\forall \tilde{c}} g_c^{\min}\, V(\tilde{c}) + \sum_{\tilde{c}' : g_{\tilde{c}'} > g_{c'}^{\min}} \left(g_{\tilde{c}'} - g_{c'}^{\min}\right)\, V(\tilde{c}') - \sum_{\tilde{c}' : g_{\tilde{c}'} < g_{c'}^{\min}} \left(g_{c'}^{\min} - g_{\tilde{c}'}\right)\, V(\tilde{c}') \\ &= \sum_{\forall \tilde{c}} g_c^{\min}\, V(\tilde{c}) + \Delta M^+ - \Delta M^- \,, \end{aligned} \tag{3.128}$$

which means the total mass of the local minimal masses, plus the total masses over this minimum, minus the local masses missing in these local minima. One can rewrite the formula as

$$\Delta M^+ - \Delta M^- = M - \sum_{\forall \tilde{c}} g_c^{\min}\, V(\tilde{c}) \tag{3.129}$$

and using the definition of $g_c^{\min}$ and assumption (3.124), one can rewrite it as

$$\begin{aligned} \Delta M^+ - \Delta M^- &= M - \sum_{\forall \tilde{c}} \sum_{c' \in C(c)} g_c^{\min}\, V(\tilde{c} \cap c') \geq M - \sum_{\forall \tilde{c}} \sum_{c' \in C(c)} g_{c'}\, V(\tilde{c} \cap c') \\ &= M - \sum_{\forall \tilde{c}} \sum_{\forall c'} g_{c'}\, V(\tilde{c} \cap c') = M - \sum_{\forall c'} g_{c'} \sum_{\forall \tilde{c}} V(\tilde{c} \cap c') \\ &= M - \sum_{\forall c'} g_{c'}\, V(c') = M - M = 0 \,. \end{aligned} \tag{3.130}$$

That is, $\Delta M^- \leq \Delta M^+$, and thus we proved, that it is always possible to finish the repair process by extending the repair stencil.
The presented repair process is naturally conservative – we are just exchanging the mass from one cell to another, no new mass is added or taken away. Moreover, this algorithm does not affect the linearity preservation of the complete remapping process. In the case of global linear function, the local extrema are situated in the neighboring cells, not in the actual one, and the repair process is not turned on at all. Problems may only appear in the regions close to the domain boundary, where the local extremes must be treated by suitable boundary conditions – by extrapolating the global underlying function to the external ghost cells.

3.3.6 Remapping of All Quantities

In this section we discuss the algorithm for remapping of all conservative quantities from the Lagrangian computational mesh, to the new, smoothed one. The complete algorithm is based on an older version of the remapping algorithm presented in [70]. The following properties are required from the algorithm:

1. Conservation – the total mass, momentum in both directions, and total energy must remain unchanged. This together with the Lagrangian step conservativity ensures the conservation of the complete ALE algorithm.

2. Local-bound preservation – the remapped mass density, velocities in both directions, and internal energy, must remain in the local bounds from the original values. This is important for reduction of possible generation of new peaks and oscillations during the remapping stage.

3. Accuracy – by accuracy, we mean in this context, that the remapping process preserves any linear function. This property corresponds in practical tests to the second order of accuracy. In [70], more accuracy issues are discussed.

4. Reversibility – this property says, that in the case of identical Lagrangian and new mesh, the remapped values must remain unchanged. This natural and important property is related to the continuous dependence of the change of primary quantities between the old and the new mesh [70].

We discuss here such remapping algorithm and describe all the important concerns of it. We use the remapping algorithm for conservative remapping of density of one arbitrary conservative quantity inside this section, which is described in previous sections. Let us remark, that satisfaction of the mentioned required features is mostly implied by their satisfaction by the method for remapping of one quantity.
The complete algorithm for remapping of all quantities to the new mesh can be described in the following way. At first, we define all the needed quantities in the mesh subzones. This can be understood as defining the quantities on a double-dense mesh (mesh of subzones), four values in each cell. This stage is called a gathering stage in [70]. It is important do define the subzonal quantities conservatively, such that their total value in the zones is equal to their sum over the corresponding subzones. The masses m_c^n (and thus densities ρ_c^n) are already defined in subzones, we need to define also subzonal momentum densities in both directions μ_c^n, ν_c^n (and thus subzonal velocities u_c^n, v_c^n), and the subzonal internal and kinetic energy densities E_c^n, K_c^n. Definitions of all mentioned subzonal quantities corresponding to the subzone of cell c at node n are presented as follows

$$\rho_c^n = \frac{m_c^n}{V_c^n} \tag{3.131}$$

$$\mu_c^n = \rho_c^n \, u_n \tag{3.132}$$

$$\nu_c^n = \rho_c^n \, v_n \tag{3.133}$$

$$E_c^n = \rho_c^n \, \epsilon_c \tag{3.134}$$

$$K_c^n = \frac{1}{2} \, \rho_c^n \, (u_n^2 + v_n^2) \, . \tag{3.135}$$

The definitions of subzonal velocities and specific internal and kinetic energies are obvious

$$\mathbf{w}_c^n = \mathbf{w}_n \tag{3.136}$$

$$\mathcal{E}_c^n = \epsilon_c \tag{3.137}$$

$$\mathcal{K}_c^n = \frac{1}{2} \, (u_n^2 + v_n^2) \, . \tag{3.138}$$

It is easy to check, that this distribution of the conservative quantities to subzones is conservative.
Now comes the remapping stage, in which we remap each subzonal conservative quantity from the Lagrangian mesh to the smoothed one. The first comes the subzonal density of mass ρ_c^n, which is remapped to the new values $\rho_{\tilde{c}}^{\tilde{n}}$, and then repaired (for explanation see section 3.3.5) according to the local extrema computed from subzonal densities in subzones neighboring the actual one. Now, one can update the new zonal, nodal, and subzonal masses, and zonal densities

$$m_{\tilde{c}}^{\tilde{n}} = \rho_{\tilde{c}}^{\tilde{n}} \, V_{\tilde{c}}^{\tilde{n}} \tag{3.139}$$

$$m_{\tilde{c}} = \sum_{\tilde{n} \in N(\tilde{c})} m_{\tilde{c}}^{\tilde{n}} \tag{3.140}$$

$$m_{\tilde{n}} = \sum_{\tilde{c} \in C(\tilde{n})} m_{\tilde{c}}^{\tilde{n}} \tag{3.141}$$

$$\rho_{\tilde{c}} = \frac{m_{\tilde{c}}}{V_{\tilde{c}}} \, . \tag{3.142}$$

Now, let us have a look at the momentum remapping. Using the same procedure for one conservative remapping, we remap both momentum densities μ_c^n, ν_c^n to the new mesh and get new momentum $\mu_{\tilde{c}}^{\tilde{n}}$,

$\nu_{\tilde{c}}^{\tilde{n}}$. This allows us to define new subzonal velocities

$$u_{\tilde{c}}^{\tilde{n}} = \frac{\mu_{\tilde{c}}^{\tilde{n}}}{\rho_{\tilde{c}}^{\tilde{n}}} \tag{3.143}$$

$$v_{\tilde{c}}^{\tilde{n}} = \frac{\nu_{\tilde{c}}^{\tilde{n}}}{\rho_{\tilde{c}}^{\tilde{n}}} \tag{3.144}$$

and the new nodal velocities

$$\mathbf{w}_{\tilde{n}} = \frac{1}{m_{\tilde{n}}} \sum_{\tilde{c} \in C(\tilde{n})} \left(m_{\tilde{c}}^{\tilde{n}} \, \mathbf{w}_{\tilde{c}}^{\tilde{n}} \right) . \tag{3.145}$$

Now, from the new nodal densities $\rho_{\tilde{n}} = m_{\tilde{n}}/V_{\tilde{n}}$ and the original nodal velocity local extremes, one can compute the nodal extremes of the momentum density

$$\mu_{\tilde{n}}^{\min} = \rho_{\tilde{n}} \, u_n^{\min} , \tag{3.146}$$

and analogically the others. Using these extremes, one can repair the nodal momentum densities

$$\mu_{\tilde{n}} = \rho_{\tilde{n}} \, u_{\tilde{n}} \tag{3.147}$$

$$\nu_{\tilde{n}} = \rho_{\tilde{n}} \, v_{\tilde{n}} \tag{3.148}$$

and then recompute the new velocities back

$$u_{\tilde{n}} = \frac{\mu_{\tilde{n}}}{\rho_{\tilde{n}}} \tag{3.149}$$

$$v_{\tilde{n}} = \frac{\nu_{\tilde{n}}}{\rho_{\tilde{n}}} . \tag{3.150}$$

This causes the new nodal velocities to satisfy the local-bound preservation condition.
The last quantity we need to remap, is the density of the subzonal total energy

$$T_c^n = E_c^n + K_c^n , \tag{3.151}$$

which gives us the new subzonal total energy $T_{\tilde{c}}^{\tilde{n}}$. From the new density and new velocities, we compute the new density of kinetic energy

$$K_{\tilde{c}}^{\tilde{n}} = \frac{1}{2} \, \rho_{\tilde{c}}^{\tilde{n}} \, \left(u_{\tilde{n}}^2 + v_{\tilde{n}}^2 \right) \tag{3.152}$$

and the new subzonal and zonal density of internal energy

$$E_{\tilde{c}}^{\tilde{n}} = T_{\tilde{c}}^{\tilde{n}} - K_{\tilde{c}}^{\tilde{n}} \tag{3.153}$$

$$E_{\tilde{c}} = \frac{1}{V_{\tilde{c}}} \sum_{\tilde{n} \in N(\tilde{c})} E_{\tilde{c}}^{\tilde{n}} \, V_{\tilde{c}}^{\tilde{n}} . \tag{3.154}$$

From the original cellular specific internal energy ϵ_c, we compute local extremes of the specific internal energy $\epsilon_c^{\min}$ and $\epsilon_c^{\max}$, and transform them to extremes of density of internal energy

$$E_{\tilde{c}}^{\min} = \rho_{\tilde{c}} \, \epsilon_c^{\min} \tag{3.155}$$

$$E_{\tilde{c}}^{\max} = \rho_{\tilde{c}} \, \epsilon_c^{\max} . \tag{3.156}$$

Then, we perform the repair of densities of the cell internal energy $E_{\tilde{c}}$ using to these local extremes. This procedure causes the final specific internal energy

$$\epsilon_{\tilde{c}} = \frac{E_{\tilde{c}}}{\rho_{\tilde{c}}} \tag{3.157}$$

to satisfy the local-bound preservation condition. Now, we can update the cell pressure field from the equation of state, apply the pressure boundary condition, and we have the complete algorithm for the remapping of all conservative quantities from the original Lagrangian mesh to the new, smoothed one. If the remapping method for one quantity satisfies all the required properties (which our method described in the previous sections does), this complete algorithm satisfies them also.
Compared with the method described in [70], there are several differences in our approach. In fact, we are using an older version of the method [70], which strictly separates three stages of the algorithm – gathering stage defining all subzonal quantities, remapping stage performing one by one subzonal quantity remapping and repair (all quantities are repaired in subzones), and the scattering stage defining the final nodal and zonal quantities. Next difference is in the fact, that the algorithm in [70] remaps the kinetic and internal energies separately, not their sum – the total energy. The separate energy remapping should be better, the algorithm does not mix two quantities, which may vary in an order of magnitude. We plan to add this property to the remapping algorithm in the future version. The last important difference occurs in the computation of the subzonal velocities. In our algorithm, there is no principal difference between nodal and subzonal velocities, which is correct in sense of conservativity, but it may decrease the accuracy of velocity remapping. In [70], there exist a transformation matrix in each cell, which converts the nodal velocities to subzonal ones, and vice versa. This requires computation of the inverse matrices, involving lot of additional computations. Moreover, this approach can generate consequent oscillations in the velocity field, and more repair steps would be necessary for correcting this problem.

3.3.7 Numerical Testing of Remapping Methods

In this section, we present several numerical examples showing properties of our complete remapping algorithm. Thus, we demonstrate all important features of the remapping algorithm – conservativity, linearity preservation and local-bound preservation. We focus on the importance of the limiting process in the reconstruction stage, and the repair process.
The first numerical example, we demonstrate here, is a linear function

$$g(x, y) = x + 2\,y \tag{3.158}$$

remapped over a series of 100 randomly changing quadrilateral meshes of 16^2 cells in $\langle 0, 1\rangle^2$ computational domain. In Figure 3.7, one can see the remapped values on the final mesh (a), the front view (b), and

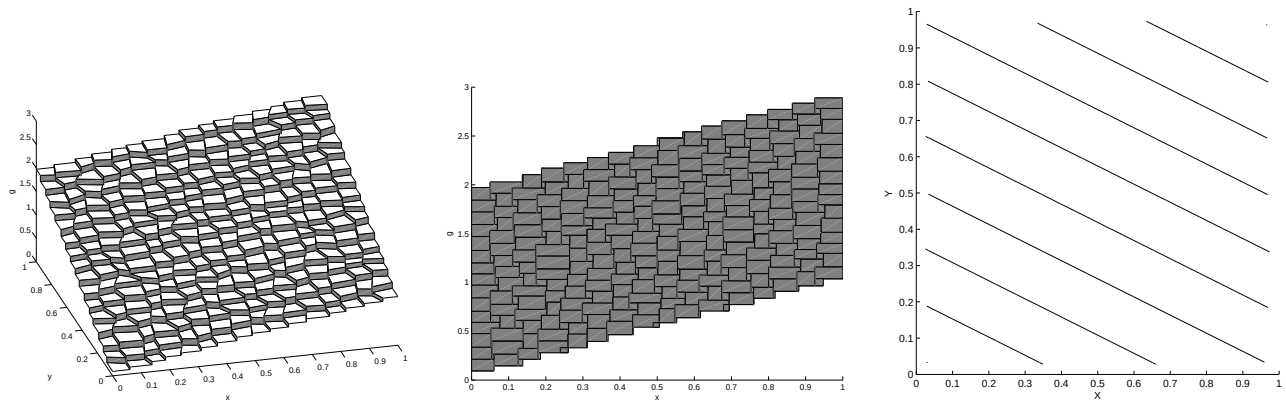


Figure 3.7: Linear function remapped over a series of randomly changing computational meshes. (a) – view to the final mean values, (b) – front view, (c) – contour map.

their contours (c). The described method, using Barth-Jespersen limited reconstruction, swept region

integration, and the final repair process, is used. This example demonstrates preservation of the global linear function, numerical error is zero up to the round off error.
In the second example, we use the following sine function

$$g(x, y) = 1 + \sin(2\,\pi\,x) + \sin(2\,\pi\,y) \tag{3.159}$$

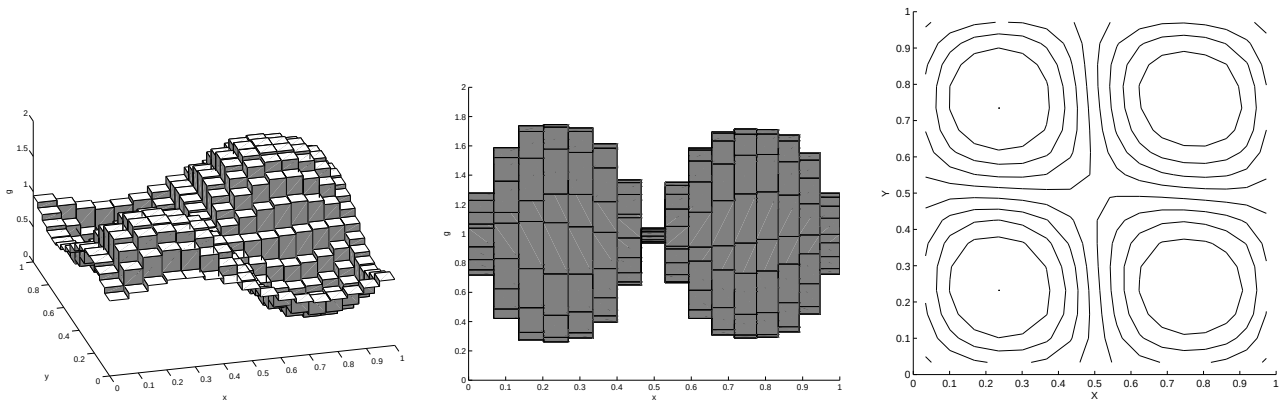
remapped over a series of 80 orthogonal, sine-like moving meshes described in [58]. The position of node $n = i, j$ in time t of the computation is described by the following formulas

$$x_{i,j}(t) = (1 - \alpha(t))\,\xi_i + \alpha(t)\,\xi_i^3 \tag{3.160}$$

$$y_{i,j}(t) = (1 - \alpha(t))\,\eta_j + \alpha(t)\,\eta_j^2 \tag{3.161}$$

$$\alpha(t) = \frac{1}{2}\,\sin\left(4\,\pi\,\frac{t}{t_{\max}}\right)\,, \tag{3.162}$$

where the logical coordinates $\xi_i = i/N_x$, and $\eta_j = j/N_y$. Clearly, for $t = 0$ and $t = t_{\max}$, the parameter $\alpha(t) = 0$, and the initial and final meshes are identical and uniformly rectangular. In Figure 3.8, the



(a) (b) (c)

Figure 3.8: Sine function remapped over a series of orthogonal, sine-like moving computational meshes. (a) – view to the final mean values, (b) – front view, (c) – contour map.

mean values (a) on the last computational mesh of 16^2 cells in $\langle 0, 1\rangle^2$ domain, the front view (b), and the contour maps (c) are shown, computed using the full limiting+swept integration+repair method. The initial and the final meshes are the same, which allows us compare the numerical errors of the remapping. In Table 3.1, relative numerical errors L_1 and $L_{\max}$ of the described problem, depending on the number

$N_x = N_y$	N_t	L_1	$\frac{L_1^{N_x}}{L_1^{2N_x}}$	$L_{\max}$	$\frac{L_{\max}^{N_x}}{L_{\max}^{2N_x}}$
16	80	6.85e-2	6.23	2.82e-1	2.52
32	160	1.10e-2	6.01	1.12e-1	2.76
64	320	1.83e-3	6.33	4.06e-2	2.78
128	640	2.89e-4	7.17	1.46e-2	2.82
256	1280	4.03e-5	—	5.17e-3	—

Table 3.1: Numerical errors of the remapping process including limited reconstruction, swept region integration, and the repair stage applied to the smooth sine function (3.159). L_1 and $L_{\max}$ errors for different mesh sizes and different number of remappings, and their ratio to the errors with half number of mesh cells in each direction, are presented.

of computational cells $N_x \times N_y$, and the number of remaps performed N_t, are presented. The L_1 error

shows at least second order of accuracy of the complete remapping method, the maximum $L_{\max}$ error slowly converges to the second order of accuracy too.
The last numerical test presented here is remapping of a non-smooth square function

$$g(x,y) = \begin{cases} 1 & \text{for} \quad \left((x-\frac{1}{2})^2 \leq 0.03\right) \wedge \left((y-\frac{1}{2})^2 \leq 0.03\right) \\ 0 & \text{else} \end{cases} \tag{3.163}$$

over a series of consequently smoothing meshes. This function includes a strong jump, and is hard for the repair algorithm to be treated in a good quality. The original mesh is a randomly distorted mesh, which is smoothed into the next step by plain averaging (3.66). The final mesh is obtained after 20 smoothing steps. In Figure 3.9, we show the mean values in the final, almost smooth mesh, using different combinations of limiting and repair methods with the approximate swept region integration. Relative numerical errors and the global extrema of all the methods on mesh with 16×16 cells, are presented in Table 3.2. From the presented numbers, one can see that all the numerical errors are

reconstruction and repair	L_1	$L_{\max}$	$g_{\min}$	$g_{\max}$	$T[s]$
swept, unlimited, no repair	9.97e-3	1.51e-1	-5.09e-2	1.058	0.032
swept, unlimited, repair	9.85e-3	1.91e-1	0	1	0.040
swept, BJ limiter, no repair	1.06e-2	1.93e-1	-3.46e-9	1.000006	0.024
swept, BJ limiter, repair	1.06e-2	1.93e-1	0	1	0.028
exact, BJ limiter, no repair	1.13e-2	1.98e-1	0	1	0.784

Table 3.2: Numerical errors L_1 and $L_{\max}$ and the global extrema $g_{\min}$ and $g_{\max}$ of the different remapping methods applied to the non-smooth square function (3.163) in 16×16 cells mesh after 20 steps. Time of computation T in seconds on a 2.4 GHz machine is also presented.

comparable, and in this particular case, the swept integration method is even better than the exact one. The limiting process dramatically decreases number of possible overshoots, but does not guarantee absolute bound preservation. The repair process helps in every situation, but due to the higher number of repairs, it causes the whole computation to need more time. From the times of computation, one can see that all swept integration simulations are much faster than the exact one.
Thus, all parts of the remapping method are essential. The swept integration is much more efficient than the exact one, the limiting process decreases the number of repairs applied, and the repair stage must be performed to guarantee bound preservation. Only combination of all these three stages guarantees optimal results for non-smooth underlying function. For more numerical tests of the 2D remapping algorithm, see [58].

3.3.8 Remapping in 3D

For the future development of the 3D ALE code, we have developed a 3D version of the complete remapping algorithm for arbitrarily polyhedral cells. The 3D algorithm was described and several preliminary examples presented in [35], and the full algorithm description with comprehensive numerical tests is presented in [36]. Here, we briefly describe the main differences of the 3D remapping process from the 2D case.
The main problem, which we have to deal with in 3D, is the fact that the faces in 3D do not have to be planar. Only the positions of the face nodes are known, the faces themselves (their shapes) are not defined. For each face f of a particular cell c, we find the face center by plain averaging all face vertices. By connecting all face vertices to this face center, we uniquely subdivide the face to triangles corresponding to each face edge. This way, every faces shape is uniquely defined.
In 3D, we use exactly the same process of error functional minimization for underlying function reconstruction, as described in section 3.3.2. In 3D, although the exact integration method was developed

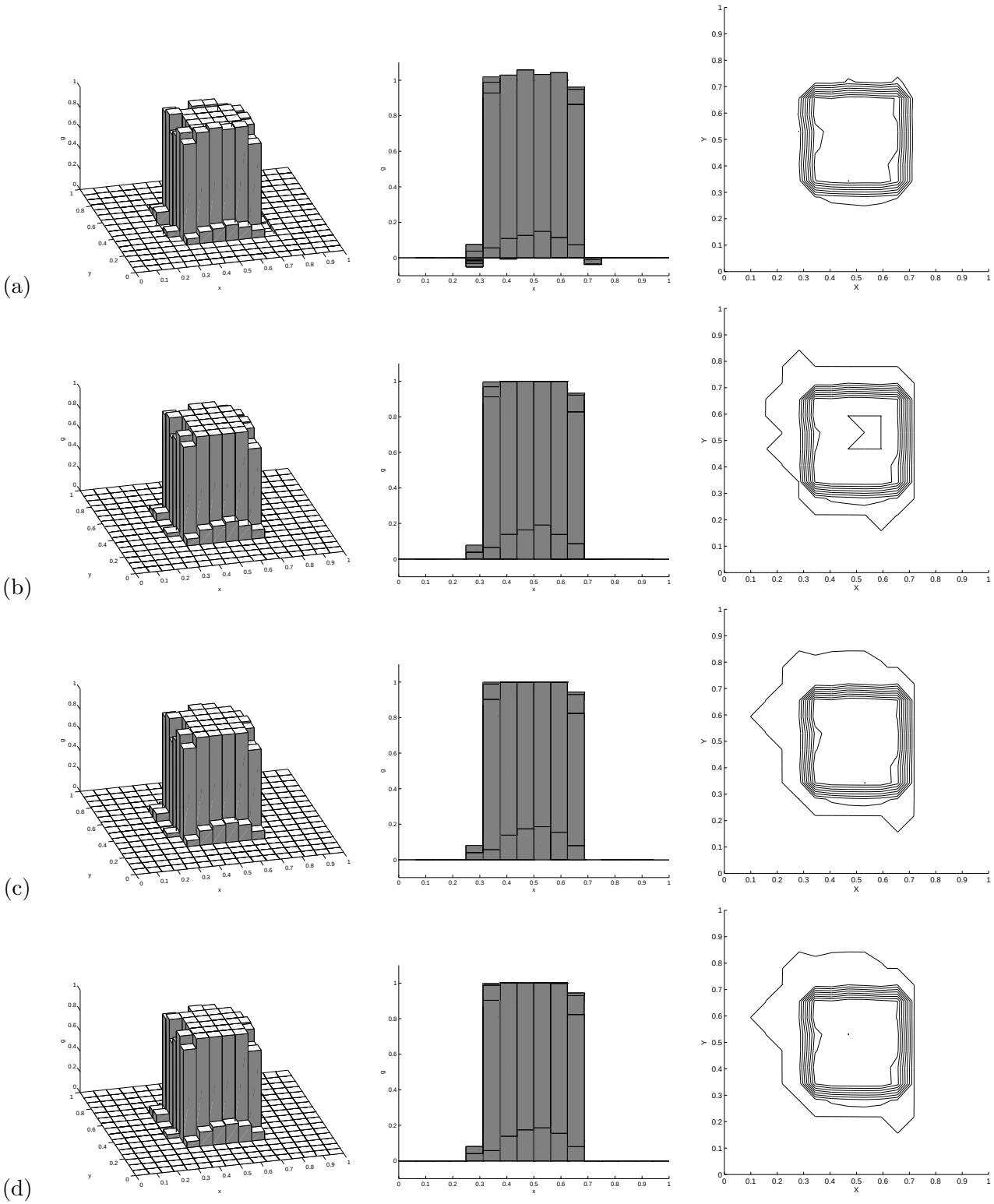


Figure 3.9: Non-smooth square function (3.163) with its front views and contour maps, remapped over a series of consequently smoothing meshes to the final mesh. Several combinations of reconstruction and repair, with the swept integration, are used – (a) unlimited slopes, (b) unlimited slopes + repair, (c) BJ limiter, and (d) BJ limiter + repair.

[38], it is quite a complicated and unfeasibly computationally expensive procedure. Thus, for practical use, we have only the swept integration method available. As for the swept integration and the repair algorithm, there is no major principle difference from the 2D case. The only really difficult part, is the integration of the linear function over arbitrary polyhedron, which is needed for the computation of the cell volumes

$$V(c) = \int_c 1 \, dx \, dy \, dz \,, \tag{3.164}$$

cell centers

$$x_c = \frac{1}{V(c)} \int_c x \, dx \, dy \, dz \,, \qquad y_c = \frac{1}{V(c)} \int_c y \, dx \, dy \, dz \,, \qquad z_c = \frac{1}{V(c)} \int_c z \, dx \, dy \, dz \,, \tag{3.165}$$

swept volumes, and masses analogical to 2D swept masses (3.112). This integration procedure is consuming most of the computational time. Let us describe the integration process briefly.
For the integration of the linear function over arbitrary polyhedron with faces split into triangles as described before (for example cell c), we use the method introduced in [74]. Let us demonstrate the integration procedure on the example of the computation of the cell volume, which can be written in the form

$$V(c) = \int_c 1 \, dV = \int_c \operatorname{div} \mathbf{\Phi}_1 \, dV \; = \oint_{\partial c} \mathbf{\Phi}_1 \cdot \mathbf{n}_c^+ \, dS \,, \tag{3.166}$$

where the vector function $\mathbf{\Phi}_1$ have the form

$$\mathbf{\Phi}_1 = \frac{1}{3} \, (x, y, z)^T \,, \tag{3.167}$$

and $\mathbf{n}_c^+$ is the outer normal of the cell c. The surface integral can be represented as sum of the face integrals, which can be composed as the sums of integrals over corresponding triangles. Therefore, the previous expression for volume can be rewritten in the following way

$$V(c) = \frac{1}{3} \sum_{f \in F(c)} \sum_{\Delta \in T(f)} \sum_{\kappa = x,y,z} n_c^{+,\kappa}(\Delta) \int_\Delta \kappa \, dS \,, \tag{3.168}$$

where $n_c^{+,x}$, $n_c^{+,y}$, $n_c^{+,z}$, are components of the normal $\mathbf{n}_c^+(\Delta)$ corresponding to the triangle Δ, whose orientation is consistent with outward normal to boundary of cell c. The symbol $F(c)$ denotes the set of all faces of cell c, and $T(f)$ denotes the set of all triangles of face f.
To evaluate the integrals

$$\int_\Delta \kappa \, dS \tag{3.169}$$

for $\kappa = x, y, z$, we project each triangle to one of the coordinate planes, as described in [74]. Suppose, that the triangle Δ belongs to the plane

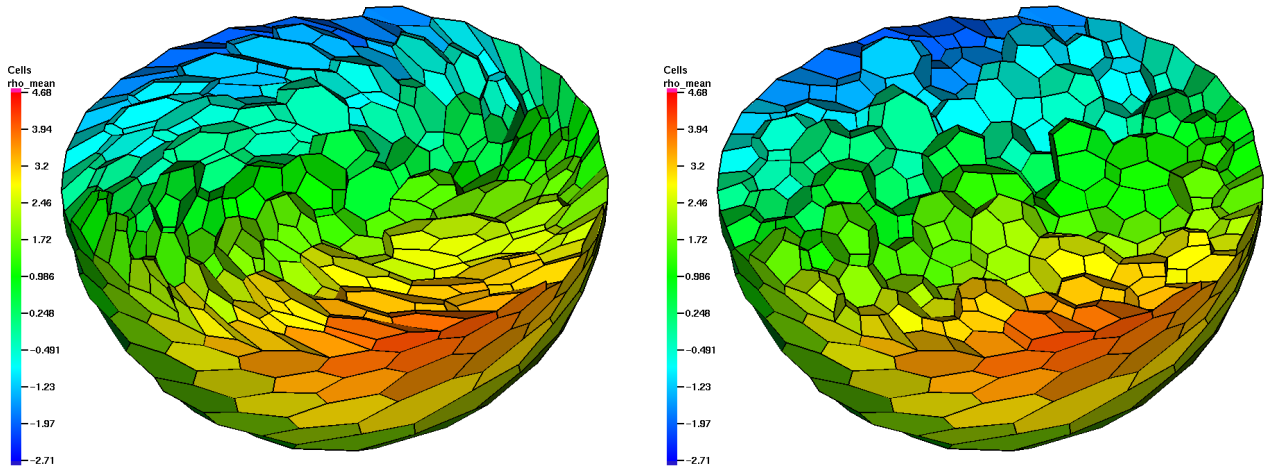
$$n_c^{+,x}(\Delta) \, x + n_c^{+,y}(\Delta) \, y + n_c^{+,z}(\Delta) \, z + \omega = 0 \,, \tag{3.170}$$

where $\omega = -\mathbf{n}_c^+ \cdot \mathbf{p}$ and $\mathbf{p}$ is an arbitrary point in the plane (for example, one of the vertices of the triangle). To reduce the error of computations, for the given triangle Δ we choose $\alpha - \beta - \gamma$ as right-handed permutation of $x - y - z$ coordinates, such that $|n_c^{+,\gamma}|$ is maximal. If we denote projection of the triangle Δ to (α, β) plane by Π_Δ, then integrals over Δ can be reduced to integrals over Π_Δ as follows

$$\int_\Delta \alpha \, dS = |n_c^{+,\gamma}|^{-1} \, J_\alpha^{\Pi_\Delta} \tag{3.171a}$$

$$\int_\Delta \beta \, dS = |n_c^{+,\gamma}|^{-1} \, J_\beta^{\Pi_\Delta} \tag{3.171b}$$

$$\int_\Delta \gamma \, dS = -|n_c^{+,\gamma}|^{-1} \, (n_c^{+,\gamma})^{-1} \, (n_c^{+,\alpha} \, J_\alpha^{\Pi_\Delta} + n_c^{+,\beta} \, J_\beta^{\Pi_\Delta} + \omega \, J_1^{\Pi_\Delta}) \,, \tag{3.171c}$$



(a) (b)

Figure 3.10: Distribution of the linear function (3.174) in the original twisted mesh (a), and the final RJ method smoothed mesh (b). In both pictures, cut in $z = 0$ plane is shown.

where the integral $J_g^{\Pi_\Delta}$ is defined as

$$J_g^{\Pi_\Delta} = \int_{\Pi_\Delta} g \, d\alpha \, d\beta \,. \tag{3.172}$$

Finally, using the Green theorem in (α, β) plane, the integrals $\int_{\Pi_\Delta} g \, d\alpha \, d\beta$ of polynomial function g are reduced to the 1D integrals over the edges and evaluated in the form

$$J_\alpha^{\Pi_\Delta} = \frac{1}{6} \operatorname{sign}(n^{+,\gamma}) \sum_{e \in \partial\Delta} \delta\beta_e \, (\alpha_{e_2}^2 + \alpha_{e_2} \, \alpha_{e_1} + \alpha_{e_1}^2) \tag{3.173a}$$

$$J_\beta^{\Pi_\Delta} = -\frac{1}{6} \operatorname{sign}(n^{+,\gamma}) \sum_{e \in \partial\Delta} \delta\alpha_e \, (\beta_{e_2}^2 + \beta_{e_2} \, \beta_{e_1} + \beta_{e_1}^2) \tag{3.173b}$$

$$J_1^{\Pi_\Delta} = \frac{1}{2} \operatorname{sign}(n^{+,\gamma}) \sum_{e \in \partial\Delta} \frac{1}{2} \, (\delta\beta_e \, (\alpha_{e_2} + \alpha_{e_1}) - \delta\alpha_e \, (\beta_{e_2} + \beta_{e_1})) \,, \tag{3.173c}$$

where each edge e has the end points $[e_1, e_2]$ in α, β coordinates, and $\delta\alpha_e = \alpha_{e_2} - \alpha_{e_1}$ and similarly $\delta\beta_e$. The final formula is then the combination of (3.173), (3.171), and (3.168).
Computation of integrals of arbitrary linear functions follows the same path, just the vector function $\mathbf{\Phi}$ has to be redefined. For example, in the case of the computation of the integral of x coordinate (x_c from equation (3.165)), we use the form of $\mathbf{\Phi}_x = (\frac{1}{2} \, x^2, 0, 0)^T$ suggested in [74]. Then, in the final stage one needs to compute 1D integrals of higher degree polynomial function, which still can be done explicitly.
For completeness, we present here one 3D numerical example of the remapping process. To show the linearity-preservation property of the algorithm, we have chosen the example of a linear underlying function

$$g(x, y, z) = 1 + 3 \, x + y + 2 \, z \tag{3.174}$$

remapped over a series of 3D polyhedral meshes including about 700 different polyhedrons with up to 21 faces in a unit sphere. Using the Reference Jacobian mesh smoothing method (RJM) described in section 3.2.2, we get a series of 19 consecutive meshes – the original twisted one, and 18 gradually smoothing meshes. Each mesh is obtained from the previous one by applying 10 steps of the reference Jacobian smoothing process. In Figure 3.10, we present the cut in $z = 0$ plane through the original twisted unit-sphere mesh, and the final RJM smoothed mesh. The remapped mean values of the linear

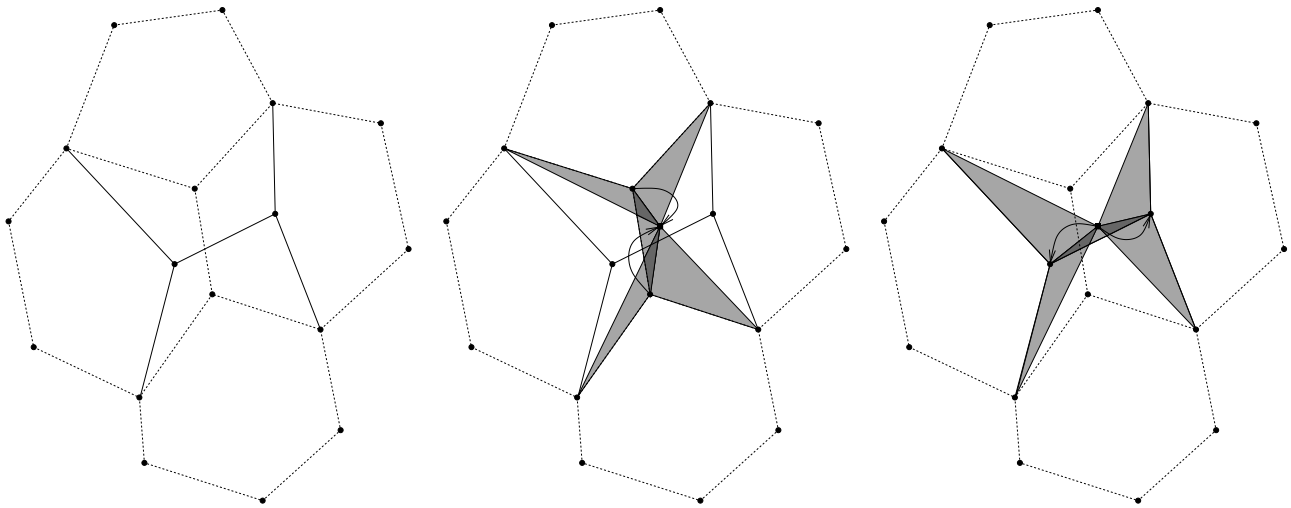


Figure 3.11: One reconnection in 2D mesh (a). Solving in two steps: shrinking old edges to one point (b) and expanding it to the new edges (c). Original mesh in dashed lines, new one in solid lines. Swept regions are shaded in gray.

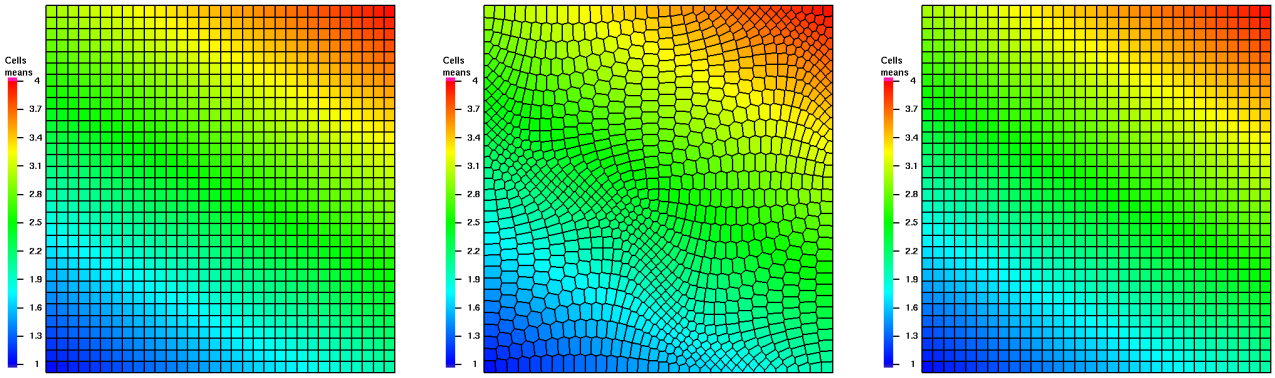
underlying function (3.174) correspond exactly to the values of the function in the new cell centers, the numerical error is (up to round-off error) zero. For more numerical examples of 3D remapping, see [36].

3.3.9 Remapping with Changing Topology

Next useful generalization of the remapping method for the future, is remapping from the Lagrangian mesh to the rezoned (smoothed) mesh with different topology. This is an important issue, because some of the mesh rezoning methods are not based on a pure nodal movement, they can also change the connections between them. We have developed generalization of the described remapping algorithm to the case of meshes with changing topology [57], which satisfies the same properties (conservativity, reversibility, linearity and local-bound preservation). Here, we only summarize the main ideas of this method.

In topology changing case in 2D, there is no difference in reconstruction and repair stages, there is no need to change anything. We have the exact integration method available without any change, there is only a problem with selecting the patch of the original mesh for intersection computation. The only requirement is, that both meshes must have consistent cells enumeration – corresponding cells in both meshes must have the same index to identify the mirror of each cell in the rezoned mesh. The same problem is with the mesh patch for reconstruction limiting, and for repair stage. Determining of such cell neighborhood (called good neighborhood) is quite a complicated task [90]. For this particular case, we switched from classical neighborhood identification (just neighbors of cell) to geometrical neighborhood identification. In this approach, we find the patch as cells, which are geometrically (not topologically) close to the original one.

The swept region integration must be revised for changing topology. The typical situation, when one reconnection appears, is shown in Figure 3.11. Detection of such reconnection is easy – we have to detect two cells, which were neighbors in the original mesh, and are not neighbors any more in the rezoned mesh. In the example shown in Figure 3.11 (a), these are the left and the right cell. We detect the edge, which is removed from the original mesh, and the edge, which is new in the new mesh. We perform the swept integration in two steps. In the first step, we shrink the disappearing edge into one central point computed as an average of the nodal positions of vertices of both edges as shown in Figure 3.11 (b). We have to perform five swept integrations corresponding to the disappearing edge, and four edges



(a) (b) (c)

Figure 3.12: Linear function on the initial quadrilateral mesh (a), remapped to the central polygonal mesh (b), and remapped to the final mesh.

connected to it. In the second step, we extend the central point into the new edge doing similar next five swept integrations shown in Figure 3.11 (c). This process treats the situation of one topology change in the cell. Special care must be taken, when the reconnection appears close to the domain boundary. If there should appear more reconnections in one cell, it can be solved by reducing the smoothing, and performing the mesh smoothing technique just to the point, when one reconnection appears, and then continue to the next one.

We present here one numerical example of remapping with changing topology. As the example, we demonstrate remapping of a linear function over a series of Voronoi meshes. The initial mesh is a quadrilateral mesh of 32^2 cells, which is in fact equal to the Voronoi mesh with equidistant generators. We move the generators according to the sine movement, described in [58]. Figure 3.12 (a), presents the linear function distribution on the initial quadrilateral mesh. In the middle of the computation, after a series of remappings, we get the situation shown in Figure 3.12 (b), where the mesh has completely different connectivity. The final mesh in Figure 3.12 (c) is the same, as the original one, and the remapped mean values correspond exactly (up to round off error) to linear function values in the cell centers. Massive changes of mesh topology were done during the process, the middle-time mesh (b) is completely different, from the initial (final) one, and remapping still works well. For more numerical tests, see [57].

Chapter 4

ALE Method in Cylindrical Geometry

In this chapter, we describe changes in the ALE method, which are needed to perform simulations in the 2D cylindrical geometry. By the 2D cylindrical geometry, we understand here, that the simulated problems have cylindrical symmetry, the solution depends on the distance r from the axis z, and the position in z direction only. There is no dependence on the cylindrical angle.

4.1 Lagrangian Solver

There exist several cylindrical Lagrangian approaches, we describe two methods from [18] – the Area-Weighted Differencing (AWD) scheme and the Control Volume (CV) method. Both these methods have the same staggered discretization, as in the Cartesian case. Let us describe both generalizations of the Lagrangian method to the cylindrical coordinates briefly in the following subsections 4.1.1, 4.1.2.
At this moment we note, that the AWD scheme is presented here just for completeness. Due to its geometrical nature, it is not possible to combine it into a complete **conservative** ALE algorithm. On the other hand, the scheme itself is quite simple and easy to understand, so we do not skip the AWD method here.

4.1.1 Area-Weighted Differencing Scheme

The complete derivation of the Area-Weighted Differencing (AWD) scheme in cylindrical coordinates is presented in details in [18] as an example of numerical method preserving spherical symmetry of the solution.
As for the geometry used for the AWD approach, the cell centers r_c, z_c are computed as the arithmetical averages of the corresponding cell vertices. Similarly, edge centers are computed by plain averaging of its vertices. Now, we compute Cartesian volumes of the dual triangles A_l corresponding to all edges in each cell, as shown in Figure 4.1. The subzonal volume of the lth subzone of cell c is defined as

$$V_c^l = r_{n(c,l)} \frac{5\,A_{l-1} + 5\,A_l + A_{l+1} + A_{l+2}}{12}\,, \tag{4.1}$$

where $r_{n(c,l)}$ is the r coordinate of the node $n(c,l)$, belonging to the particular subzone. The zonal volume can be written in the form

$$V_c = \sum_{l=1}^{4} V_c^l = \sum_{l=1}^{4} A_l \frac{r_{n(c,l)} + r_{n(c,l+1)} + r_c}{3}\,, \tag{4.2}$$

as a sum of dual triangles with the r factor (average of r coordinates of all vertices of the particular triangle). Cellular, nodal, and subzonal masses are computed from the initial subzonal densities and the described volumes, using the same formula as in the Cartesian case, (3.9), (3.8).

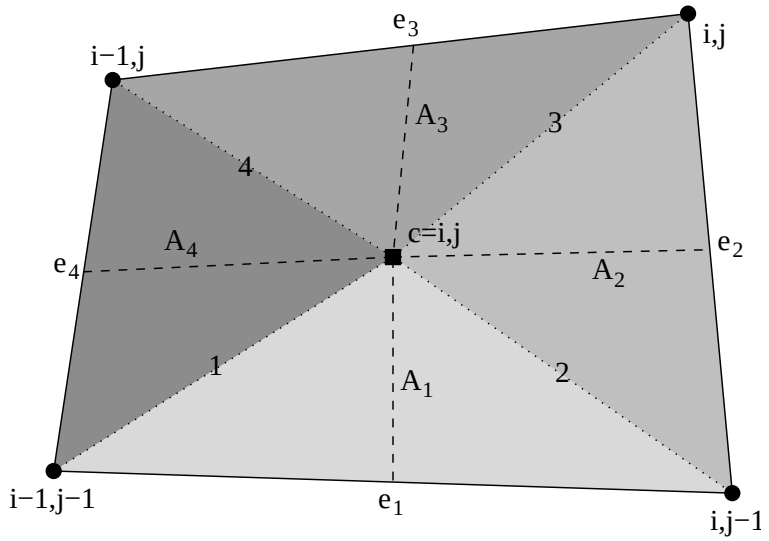


Figure 4.1: Cell c in AWD approach with dual triangles A_l (shaded in different grades of gray) corresponding to all cell edges e_l. Cell subzones are enumerated from 1 to 4.

In the AWD approach, the momentum equation has the discrete form

$$m_n \left(\frac{\partial \mathbf{w}}{\partial t}\right)_n = r_n \, \mathbf{F}_n \,, \tag{4.3}$$

and the energy equation has the form

$$m_c \left(\frac{\partial \epsilon}{\partial t}\right)_c = -\sum_{l=1}^{4} r_{n(c,l)} \, \left(u_{n(c,l)} \, F_c^{r\,l} + v_{n(c,l)} \, F_c^{z\,l}\right) \,, \tag{4.4}$$

where the velocity vector $\mathbf{w} = (u, v)$ have its direction in r, z coordinates, and the vector of corner and total nodal forces are computed using exactly the same process, as described in section 3.1.6 for Cartesian geometry. When we compare the equations with the Cartesian equations (3.50) and (3.54), one can see the obvious generalization of the Cartesian method to cylindrical geometry. The corner forces

$$\mathbf{F}_c^l = \mathbf{F}\mathbf{p}_c^l + \mathbf{F}\mathbf{d}\mathbf{p}_c^l + \mathbf{F}\mathbf{q}_c^l \tag{4.5}$$

are computed exactly the same way as in the Cartesian geometry, using the same formulas for zonal pressure (3.18), subzonal pressure (3.28), and viscosity forces. Then, just by multiplying these forces by the r coordinate of the corresponding node, we convert these forces to the cylindrical, AWD forces.
The AWD algorithm provides reasonable solutions in the cylindrical geometry, which are conservative, cylindrical symmetry preserving, and consistent with the fluid equations. Unfortunately, due to the volumes definitions, it is not possible to consistently combine this approach with the remapping stage into the complete ALE algorithm, such that the algorithm would be conservative. The reason is, that the remapping algorithm requires cell masses (and volumes) to be defined in such a way, that they can be computed as integrals of some quantity (density, 1) over the cell volume. In the AWD approach, the cell volume is defined as a sum of triangle volumes multiplied by the r factor, which cannot be transformed to any integral formula. Thus, we have to leave the AWD approach, and search for a different method, where the cell (and subzonal) volumes are defined as integrals.

4.1.2 Control Volume Method

The second method, which can be used for the cylindrical Lagrangian solver, is the Control Volume (CV) method shown in [18]. This method has one disadvantage – it does not preserve the spherical symmetry

(for more details see [18], some more details were provided by [67]). On the other hand, this property at the moment is not so important in our real calculations, it is fully overweighted by the fact, that the volume is "naturally" defined as the integral in cylindrical coordinates over the cell, which is needed for the combination with the conservative remapping method in the ALE code, as described in the last paragraph of the previous section.
As for the geometry (defined by the method of volume evaluation) in the CV method, the cell volumes are defined as cylindrical integrals of 1 over the cell

$$V_c = \int_c 1\, r\, dr\, dz\,, \tag{4.6}$$

which can be reduced by the Green theorem to the edge integrals and evaluated from the coordinates of the edge vertices

$$V_c = \sum_{e\in\partial c} \int_e \frac{r^2}{2}\, dz = \frac{1}{6} \sum_{l=1}^{4} (z_{l+1} - z_l)\, \left(r_l^2 + r_{l+1}^2 + r_l\, r_{l+1}\right)\,. \tag{4.7}$$

Here (r_l, z_l) are coordinates of node $n(c,l)$ in (r,z) coordinate system, where the nodes are enumerated as introduced in Figure 3.2. The subzonal volumes V_c^l are computed exactly the same way, and due to to the volume defined by the analytical integration, clearly

$$V_c = \sum_{l=1}^{4} V_c^l\,. \tag{4.8}$$

Analogically, the cell centers are defined as cylindrical integrals of the corresponding coordinate over the cell

$$r_c = \frac{1}{V_c} \int_c r\, r\, dr\, dz = \frac{1}{V_c} \sum_{e\in\partial c} \int_e \frac{r^3}{3}\, dz \tag{4.9}$$

$$z_c = \frac{1}{V_c} \int_c z\, r\, dr\, dz = \frac{1}{V_c} \sum_{e\in\partial c} \int_e \frac{r^2}{2}\, z\, dz\,, \tag{4.10}$$

which can be evaluated from the edges vertices similarly. In [18], arithmetical averages of cell vertices coordinates are used for the cell centers computation, but we need the centroids (4.9), (4.10) for the combination with the remapping stage in the ALE method. Our numerical tests show, that this algorithm change does not affect the numerical stability nor the order of accuracy of the cylindrical CV method.
Let us derive the cylindrical CV forces, as we did for the Cartesian geometry in section 3.1.1.
The momentum equations has the following general form

$$\rho\, \frac{d}{dt}\mathbf{w} = -\nabla\, p\,, \tag{4.11}$$

not depending on the coordinate system. Gradient of arbitrary function in the cylindrical coordinates can be written as

$$\nabla\, f = \mathbf{grad}\, f = \left(\frac{\partial\, f}{\partial\, r}, \frac{1}{r}\, \frac{\partial\, f}{\partial\, \varphi}, \frac{\partial\, f}{\partial\, z}\right)\,, \tag{4.12}$$

or applied to the pressure p in the cylindrical symmetry (cylindrical symmetry assumes, that the function does not depend on φ) we get

$$\mathbf{grad}\, p = \left(\frac{\partial\, p}{\partial\, r}, 0, \frac{\partial\, p}{\partial\, z}\right)\,, \tag{4.13}$$

where the derivatives can be written as

$$\frac{\partial\, p}{\partial\, z} = \frac{1}{r}\, \frac{\partial (p\, r)}{\partial\, z} \tag{4.14}$$

$$\frac{\partial\, p}{\partial\, r} = \frac{1}{r}\, \left(\frac{\partial (p\, r)}{\partial\, r} - p\right)\,. \tag{4.15}$$

The formulas are equivalent, the right hand side can be reduced to the left hand side form, we need it in this special form for further derivations. Now, the original system (4.11) with $\mathbf{w} = (u, v)$ can be rewritten as

$$\rho \frac{d\,u}{d\,t} = -\frac{1}{r}\left(\frac{\partial\,(p\,r)}{\partial\,r} - p\right) \tag{4.16}$$

$$\rho \frac{d\,v}{d\,t} = -\frac{1}{r}\frac{\partial\,(p\,r)}{\partial\,z}\,. \tag{4.17}$$

After multiplying the equations by r and integrating over the nodal region V_n (the same as the Cartesian nodal region from Figure 3.3), we get the cylindrical nodal forces on the right hand side of both equations. Let us focus on the force in the r direction at first. By assuming a constant pressure value in each subzone, it can be written as

$$Fp_n^r + Fdp_n^r = -\int_{V_n} \frac{\partial\,(p\,r)}{\partial\,r}\,dr\,dz + \int_{V_n} p\,dr\,dz = -\int_{V_n} \frac{\partial\,(p\,r)}{\partial\,r}\,dr\,dz + \sum_{l=1}^{4}\int_{V_n^l} p\,dr\,dz\,, \tag{4.18}$$

and using the Green formula, one can reduce both integrals to the boundary integrals to the form

$$Fp_n^r + Fdp_n^r = -\int_{\partial V_n} p\,r\,dz + \sum_{l=1}^{4} p^n_{c(n,l)} \int_{\partial V_n^l} r\,dz\,, \tag{4.19}$$

where the second integral was decomposed to the corresponding integrals over all subzones adjacent to the node n, and the subzonal pressure was moved in front of the integral. Both boundary integrals can be written as a sum of edge integrals, and the edge pressure is defined as the average of the subzonal pressures in both adjacent subzones (as in the Cartesian case). The formula is then

$$\begin{aligned} &Fp_n^r + Fdp_n^r = \\ &\quad -\sum_{l=1}^{4}\left(-\frac{p^l_{c(n,l)} + p^{l+1}_{c(n,l)}}{2}\int_{s_l^{c(n,l)}} r\,dz + \frac{p^l_{c(n,l)} + p^{l-1}_{c(n,l)}}{2}\int_{s_{l-1}^{c(n,l)}} r\,dz\right) \\ &\quad +\sum_{l=1}^{4} p^n_{c(n,l)}\left(-\int_{s_l^{c(n,l)}} r\,dz + \int_{s_{l-1}^{c(n,l)}} r\,dz - \int_{a_l^{c(n,l)-}} r\,dz + \int_{a_l^{c(n,l)+}} r\,dz\right) \end{aligned}\,. \tag{4.20}$$

After substituting the formula (3.24) for the subzonal pressure, and after evaluating the integrals

$$I\,(s_l^c) = \int_{s_l^c} r\,dz = \left(z_c - z_{e(c,l)}\right)\frac{r_{e(c,l)} + r_c}{2} \tag{4.21}$$

$$I\left(a_l^{c\pm}\right) = \int_{a_l^{c\pm}} r\,dz = \left(z_{e(c\pm,l)} - z_{n(c,l)}\right)\frac{r_{e(c\pm,l)} + r_{n(c,l)}}{2}\,, \tag{4.22}$$

one can write the force as

$$\begin{aligned} &Fp_n^r + Fdp_n^r = \\ &\quad -\sum_{l=1}^{4}\left(-\left(p_{c(n,l)} + \frac{\delta p^l_{c(n,l)} + \delta p^{l+1}_{c(n,l)}}{2}\right) I(s_l^{c(n,l)}) + \left(p_{c(n,l)} + \frac{\delta p^l_{c(n,l)} + \delta p^{l-1}_{c(n,l)}}{2}\right) I(s_{l-1}^{c(n,l)})\right) \\ &\quad +\sum_{l=1}^{4}\left(p_{c(n,l)} + \delta p^l_{c(n,l)}\right)\left(-I(s_l^{c(n,l)}) + I(s_{l-1}^{c(n,l)}) - I(a_l^{c(n,l)-}) + I(a_l^{c(n,l)+})\right) \end{aligned}\,. \tag{4.23}$$

By rearranging all the terms, we obtain the force in the form

$$\begin{aligned} Fp_n^r + Fdp_n^r = &\sum_{l=1}^{4} p_{c(n,l)} \left(I(a_l^{c(n,l)+}) - I(a_l^{c(n,l)-}) \right) \\ &+ \sum_{l=1}^{4} \delta p_{c(n,l)} \left(I(a_l^{c(n,l)+}) - I(a_l^{c(n,l)-}) \right) \\ &+ \frac{1}{2} \sum_{l=1}^{4} I(s_l^{c(n,l)}) \left(\delta p_{c(n,l)}^{l+1} - \delta p_{c(n,l)}^{l} \right) + \frac{1}{2} \sum_{l=1}^{4} I(s_{l-1}^{c(n,l)}) \left(\delta p_{c(n,l)}^{l} - \delta p_{c(n,l)}^{l-1} \right) , \end{aligned} \tag{4.24}$$

which corresponds exactly to the Cartesian formula (3.28). Obviously, the first term of the formula corresponds to the zonal pressure force in the cylindrical coordinates.
The nodal force in z direction is computed similarly. From (4.17), one can express the force in the form

$$Fp_n^z + Fdp_n^z = - \int_{V_n} \frac{\partial (p\, r)}{\partial z} \, dr \, dz \, , \tag{4.25}$$

and after applying the Green formula, one can rewrite it as a boundary integral

$$Fp_n^z + Fdp_n^z = \int_{\partial V_n} p\, r \, dr \, . \tag{4.26}$$

The sign change arises from the minus sign in the Green formula for the second coordinate. As in the previous case, we decompose the boundary integral to the sum of edge integrals, and define the value of pressure at the edge to the average of the pressures in the adjacent subzones, and rewrite the formula in the form

$$Fp_n^z + Fdp_n^z = \sum_{l=1}^{4} \left(- \frac{p_{c(n,l)}^{l} + p_{c(n,l)}^{l+1}}{2} \int_{s_l^{c(n,l)}} r \, dr + \frac{p_{c(n,l)}^{l} + p_{c(n,l)}^{l-1}}{2} \int_{s_{l-1}^{c(n,l)}} r \, dr \right) . \tag{4.27}$$

Again, we can rewrite the subzonal pressures as variations of the zonal pressures, and use the following symbols

$$J\left(s_l^c\right) = - \int_{s_l^c} r \, dr = - \left(r_c - r_{e(c,l)} \right) \frac{r_{e(c,l)} + r_c}{2} \tag{4.28}$$

$$J\left(a_l^{c\pm}\right) = - \int_{a_l^{c\pm}} r \, dr = - \left(r_{e(c\pm,l)} - r_{n(c,l)} \right) \frac{r_{e(c\pm,l)} + r_{n(c,l)}}{2} \tag{4.29}$$

for the edge integrals, which allows us to rewrite the force to the form

$$\begin{aligned} &Fp_n^z + Fdp_n^z = \\ &\sum_{l=1}^{4} \left(\left(p_{c(n,l)} + \frac{\delta p_{c(n,l)}^{l} + \delta p_{c(n,l)}^{l+1}}{2} \right) J\left(s_l^{c(n,l)}\right) - \left(p_{c(n,l)} + \frac{\delta p_{c(n,l)}^{l} + \delta p_{c(n,l)}^{l-1}}{2} \right) J\left(s_{l-1}^{c(n,l)}\right) \right) . \end{aligned} \tag{4.30}$$

For further changes, we need the assumption of the closed subzone l of cell $c(n,l)$, which implies the following equation

$$0 = \int_{\partial V_{c(n,l)}^{l}} r \, dr = - \left(-J\left(a_l^{c(n,l)-}\right) + J\left(a_l^{c(n,l)+}\right) - J\left(s_l^{c(n,l)}\right) + J\left(s_{l-1}^{c(n,l)}\right) \right) , \tag{4.31}$$

or

$$J\left(a_l^{c(n,l)+}\right) - J\left(a_l^{c(n,l)-}\right) = J\left(s_l^{c(n,l)}\right) - J\left(s_{l-1}^{c(n,l)}\right) . \tag{4.32}$$

This equality allows us to express the final z force (after some regrouping of terms) as

$$\begin{aligned} Fp_n^z + Fdp_n^z = &+ \sum_{l=1}^{4} p_{c(n,l)} \left(J(a_l^{c(n,l)+}) - J(a_l^{c(n,l)-}) \right) \\ &+ \sum_{l=1}^{4} \delta p_{c(n,l)} \left(J(a_l^{c(n,l)+}) - J(a_l^{c(n,l)-}) \right) \\ &+ \frac{1}{2} \sum_{l=1}^{4} J(s_l^{c(n,l)}) \left(\delta p_{c(n,l)}^{l+1} - \delta p_{c(n,l)}^{l} \right) + \frac{1}{2} \sum_{l=1}^{4} J(s_{l-1}^{c(n,l)}) \left(\delta p_{c(n,l)}^{l} - \delta p_{c(n,l)}^{l-1} \right) , \end{aligned} \tag{4.33}$$

which is the same formula as the one for the r force (4.24), with Js instead of Is.
The cylindrical forces are computed using exactly the same formulas, as the Cartesian forces (3.28). The only difference is in the evaluation of the I and J integrals. When we compare the Cartesian (3.17), (3.21) and cylindrical (4.21), (4.28) I, J integrals, there is the additional r factor equal to the average of the r coordinates of the edge vertices in the cylindrical case. The rest of the cylindrical Lagrangian step can remain unchanged, the r factor in the cylindrical integrals and volumes causes that the CV method approximates the cylindrical equations.
There is still the question of the cylindrical viscosity force unresolved. Our approach, inspired by [67, 85] is based on the evaluation of the viscosity forces using exactly the same Cartesian formulas, as described in section 3.1.4. Then, we add the r factor to these final forces

$$\mathbf{Fq}_c^l = r_c^l \, \mathbf{Fq}_c^{cart\, l} , \tag{4.34}$$

where the center of the subzone r_c^l is computed using formula (4.9) applied to the subzone. This modification is sufficient for the fully working CV algorithm in cylindrical coordinates.
The CV cylindrical method is used in our cylindrical ALE code, and conservatively cooperates with the rest of the ALE algorithm.

4.2 Mesh Smoothing Techniques

The mesh smoothing is considered from a purely geometrical point of view. In the cylindrical geometry, we use the same smoothing techniques, as described in section 3.2 for Cartesian geometry, there is almost no change needed. During the smoothing process, the Cartesian cell volumes and cell centers are used, so that smoothing the same mesh in both geometries provides the same final mesh. The only point, where the mesh smoothing in cylindrical geometry is different from the Cartesian one, is the $r = 0$ boundary condition. In cylindrical geometry, the corresponding boundary nodes are allowed to move only along the z axis, they cannot move in r direction. The rest of the smoothing methods remains unchanged.

4.3 Conservative Remapping Algorithm

For conservative cooperation of the remapping stage with the Lagrangian solver, we have to use the same cylindrical geometry, as described in section 4.1.2. Let us go through all three parts of the remapping method (reconstruction, integration, and repair), and point to the changes of the algorithm compared with the original Cartesian method.

4.3.1 Piecewise Linear Reconstruction

There is no need to change the reconstruction process (3.92) by minimization the error functional combined with the Barth-Jespersen limiter (3.96), (3.95), this method does not include any integration.

4.3.2 Numerical Integration – Exact

The derivation of the exact integration process is straightforward. As in Cartesian geometry, we intersect the new cell $\tilde{c}$ with all original cells c' in its neighborhood. We denote the intersection by the symbol $P_c^{\tilde{c}'}$. Analogically to the formula (3.98), we compute the mass of this intersection region as

$$\begin{aligned} G_{P_c^{\tilde{c}'}} = g_c \int_{P_c^{\tilde{c}'}} 1\, r\, dr\, dz + \left(\frac{\partial g}{\partial r}\right)_c \left(\int_{P_c^{\tilde{c}'}} r\, r\, dr\, dz - r_c \int_{P_c^{\tilde{c}'}} 1\, r\, dr\, dz\right) \\ + \left(\frac{\partial g}{\partial z}\right)_c \left(\int_{P_c^{\tilde{c}'}} z\, r\, dr\, dz - z_c \int_{P_c^{\tilde{c}'}} 1\, r\, dr\, dz\right), \end{aligned} \tag{4.35}$$

where the integrals of r, r^2, and $r\,z$ are reduced by the Green theorem to the boundary integrals, and evaluated from the coordinates of the region vertices, as shown in section 4.1.2. For completeness, we present here the final formulas for the required integrals over a general polygon P (cell, subzone, or as in this case, cell intersections) in the cylindrical geometry

$$\int_P r\, dr\, dz = \frac{1}{6} \sum_{e \in \partial P} (z_2 - z_1)\, \left(r_1^2 + r_2^2 + r_1\, r_2\right) \tag{4.36}$$

$$\int_P r^2\, dr\, dz = \frac{1}{12} \sum_{e \in \partial P} (z_2 - z_1)\, \left(r_1^2 + r_2^2\right)\, (r_1 + r_2) \tag{4.37}$$

$$\int_P z\, r\, dr\, dz = \frac{1}{24} \sum_{e \in \partial P} (z_2 - z_1)\, \left(r_1^2\,(3\,z_1 + z_2) + r_2^2\,(3\,z_2 + z_1) + 2\,r_1\,r_2\,(z_1 + z_2)\right), \tag{4.38}$$

where the vertices of the edge $e = [e_1, e_2]$ have coordinates $e_1 = (r_1, z_1)$ and $e_2 = (r_2, z_2)$. This is the only difference from the Cartesian geometry, the rest of the method remains unchanged, and all required numerical properties are satisfied.

4.3.3 Numerical Integration – Swept

As in the exact integration process, the swept region integration method follows the Cartesian algorithm presented in section 3.3.4. For each edge e of cell c, we construct swept region used for mass exchange between the cell and the corresponding neighbor. As in the Cartesian case, the swept mass δG_e is obtained by integration of the reconstructed function from cell c^*, in which more of the swept region lies (3.112), which is given by the sign of the swept region volume. This formula can be written in cylindrical coordinates in the form

$$\begin{aligned} \delta G_e &= \int_{\delta_e} g_{c^*}(r, z)\, r\, dr\, dz \\ &= \left(g_{c^*} - \left(\frac{\partial g}{\partial r}\right)_{c^*} r_{c^*} - \left(\frac{\partial g}{\partial z}\right)_{c^*} z_{c^*}\right) \int_{\delta_e} 1\, r\, dr\, dz \\ &+ \left(\frac{\partial g}{\partial r}\right)_{c^*} \int_{\delta_e} r\, r\, dr\, dz + \left(\frac{\partial g}{\partial z}\right)_{c^*} \int_{\delta_e} z\, r\, dr\, dz\,, \end{aligned} \tag{4.39}$$

where all the needed cylindrical integrals are evaluated in the previous subsection by formulas (4.36), (4.37), and (4.38). The rest of the swept region integration routine remains unchanged, as well as all properties of this method. And analogically as in the Cartesian case, the cylindrical remapping method is an approximate integration process which may cause the local-bounds to be violated. Thus, the additional conservative repair stage must be performed to correct it.

4.3.4 Repair

There is absolutely no need to change anything in the repair process presented in section 3.3.5 in cylindrical coordinates. During this stage, we are not performing any integration, we are just conservatively moving masses among cells in some neighborhood. We just have to note that, of course, for recomputation of masses to densities (a vice versa) in cylindrical coordinates, the cylindrical cell volumes (4.7) must be used.

Chapter 5

Properties of Numerical ALE Method

In this short chapter, we comment major properties of the described ALE algorithm and demonstrate them on a set of selected well known fluid dynamics tests. Both Cartesian and cylindrical geometries are discussed in this section. We also point here to possible violations of the required properties, which may appear in some situations.

5.1 Conservativity

The condition of conservativity is naturally satisfied by the presented ALE algorithm. We discuss this property separately for the Lagrangian solver and for the remapping process.

The Lagrangian solver naturally satisfies the conservation of mass, masses in all the computational cells remain unchanged during the Lagrangian computation. Concerning the conservation of momentum and energy, their conservation strongly depend on the fluid movement on the boundary of the computational domain. If the computational domain does not change during the simulation, and no shock waves, or other types of discontinuities appear on the boundary, the Lagrangian solver is conservative. For more details on the conservation issues of the Lagrangian method, see [18]. On the other hand, if the computational domain changes, the forces (and thus momentum and energy) are applied to the deformation of the boundary, which modifies the total momentum (energy), and they cannot be preserved any more. Other possibility, how the conservation conditions can be violated, is adding some outer force to the total nodal forces, for example the gravitation force in the Rayleigh-Taylor instability problem [24, 25]. Due to this outer force, the velocities are increased, which naturally affects the total momentum and energy in the domain. Let us conclude that for zero outer force, static computational domain, and no discontinuous waves on the boundary, the Lagrangian step is conservative for all conservative quantities, and it is always conservative for mass.

Concerning the remapping process, it is applied to each conservative quantity consecutively. We have already discussed in section 3.3.6, that the remapping process is conservative for all conservative quantities. This statement is correct, if the computational domain does not change during the mesh smoothing process. For simple computational domains, such as square, the boundary nodes during smoothing only move along the boundary, the computational domain does not change its shape or volume, and the remapping method is exactly conservative. This is caused due to the fact, that masses are interchanged between neighboring cells via the swept masses fluxes and the available masses in the repair stage, but no mass is created or removed. Unfortunately, it is not clear how to perform mesh smoothing on a general boundary, thus it may happen, that for the curved boundary, the smoothed mesh has different volume than the original one. Of course, if the volume of the domain is different, the mass is different too. The example of such situation is shown in Figure 5.1. It shows the Noh problem in 2D – one of the classical fluid dynamical problems, which is a rare case, when its analytical solution is known. The initial conditions are following. The fluid of density $\rho = 1$ and zero pressure ($p = 10^{-6}$ to avoid numerical problems) is moving with the velocity $|\mathbf{w}| = 1$ towards the origin. The circular shock wave is created, and

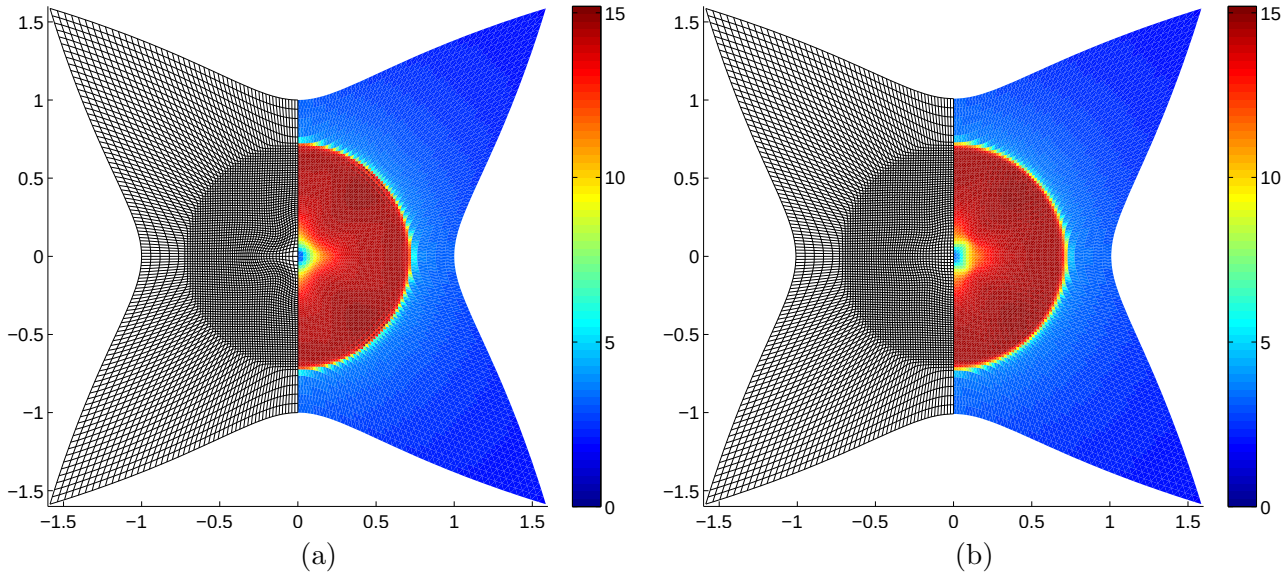


Figure 5.1: The Noh problem computed with purely Lagrangian method (a), and the simulation by complete ALE method (b) in the Cartesian coordinates. The density of fluid, and the computational mesh are shown.

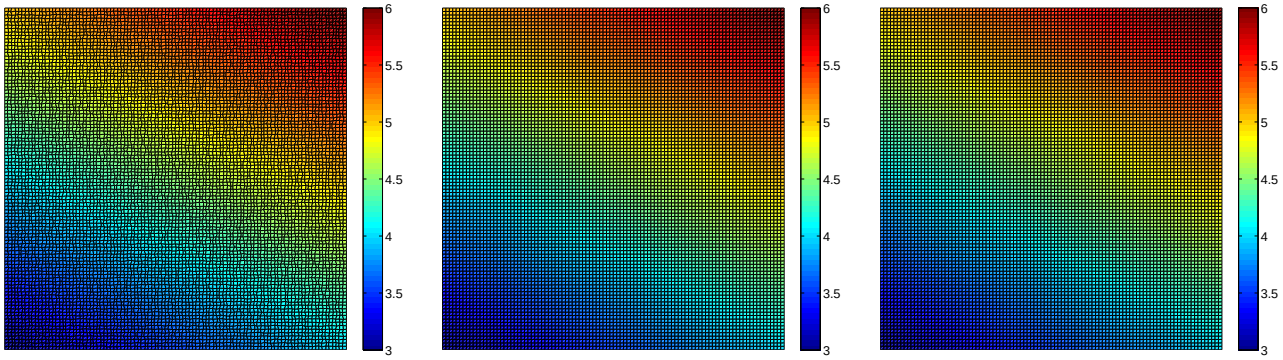
spreads with the velocity $1/3$ out of the origin. Inside the circular shock region, the fluid is at rest, and has the density $\rho = 16$. Outside of the circular region, the fluid is still moving with the initial velocity and has density $\rho = 1 + t/\sqrt{x^2 + y^2}$. In Figure 5.1, the solution computed by the pure Lagrangian, and complete ALE methods at time $t = 2.0$ is shown. Computational mesh with 100×100 cells, and bulk artificial viscosity (3.36) was used for the simulations. The solution is comparable with the Eulerian simulations presented in [49]. For the purely Lagrangian method, the total mass is exactly conservative, and the total energy is different by $0.17\,\%$ (due to movement of the boundary). For the complete ALE method, where mesh smoothing/quantity remapping is applied after every 10 Lagrangian steps (about 500 timesteps are needed till time $t = 2$), the mass conservativity is violated by $0.25\,\%$ (due to curvature of the boundary), and the total energy by $0.42\,\%$ (due to the combination of both effects). The error of the total momentum is close to the round off error in both cases thanks to the symmetry. As one can see, the cumulation of the error during the complete simulation due to the remapping non-conservativity on the boundary is lower than $0.5\,\%$ for all quantities for this particular example, and can be considered to be small enough.
In the cylindrical coordinates, the situation with conservativity of the ALE algorithm and possible problems is completely the same, and does not require further description.

5.2 Linearity Preservation, Order of Accuracy

The linearity preservation condition is satisfied in the Cartesian coordinates in the Lagrangian step, if the initial density values are set in the centers of the computational cells. We demonstrate this on a linear density problem, where the fluid density in the computational cells is initialized as a value of the following linear function

$$\rho(x, y) = x + 2\,y \tag{5.1}$$

in the cell centers and fluid pressure is set to $p = 1$. The initial mesh of 100×100 cells is a randomly distorted square mesh in the computational domain $\Omega = \langle 1, 2\rangle^2$ moving with velocity $\mathbf{w} = (1, 2)$. In Figure 5.2 (a), the purely Lagrangian simulation till time $t = 1$ is shown. The mesh is exactly the same



(a) (b) (c)

Figure 5.2: Linear density problem: (a) – solution by purely Lagrangian method; (b) – remapped from the original randomly distorted mesh over a series of consequently smoothing meshes; (c) – solution by the complete ALE algorithm.

as the initial one, it just moved with the specified velocity to the new location. The numerical error is zero up to the round off error.
The remapping process is linearity preserving in the case, that the initial density values are set in the centers of the cells subzones, and cell density is computed from the subzonal quantities. The situation after 300 remap steps is shown in Figure 5.2 (b). The final computational mesh is almost regular, and the numerical error is zero up to the round off error.
The simulation by the complete ALE method is presented in Figure 5.2 (c), where the mesh smoothing/quantity remapping step was performed after each Lagrangian step. In this case, the linearity preserving condition is not satisfied. On the other hand, the linearity preservation condition is not required by the complete ALE algorithm, only the second order of accuracy is desired. For the presented linear density problem, the numerical error converges with the second order of accuracy for refining meshes.
In the cylindrical geometry, one can just consider the linearity preservation property in z direction. During the movement in the r direction in the Lagrangian step, the function profile in the r direction is not preserved due to the geometry.

5.3 Numerical Tests

In the previous sections 5.1, 5.2, we have presented two numerical examples of fluid dynamical simulations, the Noh problem and the linear density problem. Let us note, that both problems can be computed by purely Lagrangian method, the complete ALE results are presented here for demonstration of its properties. Let us return to the Noh problem and present the corresponding cylindrical results. Then, we present one more classical test here, the Rayleigh-Taylor instability.
Figure 5.1 presents the Noh problem computed by the purely Lagrangian method (a), and the complete ALE algorithm (b) in the Cartesian geometry. The ALE algorithm does not affect the Lagrangian grid nor the solution, the density profiles are similar, and no significant diffusion appears. The Noh problem is difficult for Lagrangian-type methods – considerable density gap appears in the central region. This gap is smaller in the case of ALE computation, it is partially corrected by the Eulerian mass flux between cells. The numerical errors of the computations are similar, $L_1 = 15.2\,\%$ ($L_{max} = 79\,\%$) for the pure Lagrangian and $L_1 = 14.7\,\%$ ($L_{max} = 79\,\%$) for the simulation by the complete ALE method. The Noh problem simulation by our ALE method is described also in [51].
In the cylindrical coordinates, the 2D Noh problem can be simulated as a simple 1D problem in r direction. In Figure 5.3, one can compare plots of density computed by both methods in r direction in

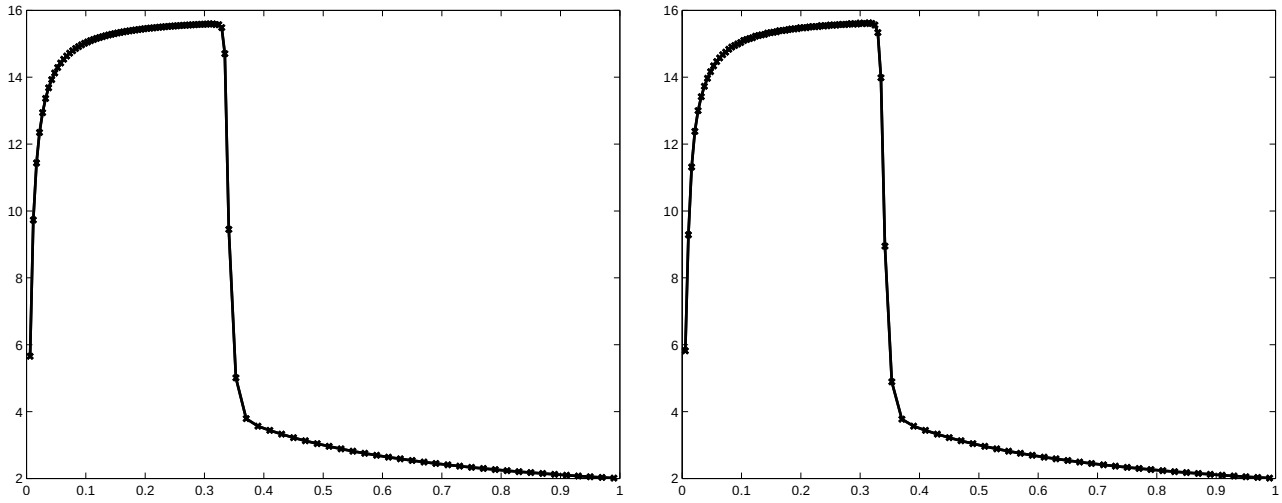


Figure 5.3: The cylindrical Noh problem computed by purely Lagrangian method (a), and the simulation by complete ALE method (b). The density of fluid in the computational cell centers is shown.

time $t = 1$. Minor dissipativity in the shock region, and the region around the central gap, can be seen, but in general, the mesh smoothing process did not affect the solution significantly. Also, the speed of the shock wave propagation is correct, which is often violated by the incorrectly constructed numerical methods.

The last numerical example which we present here, is the Rayleigh-Taylor instability (RTI) simulation in the Cartesian geometry. The initial conditions consist of two fluids in rest lying on each other, the heavier one with density $\rho = 2$ above a lightweight one with $\rho = 1$. The pressure is hydrostatic, and the fluid interface is disturbed a little by a sine like profile to start the instability evolution. The initial density on a 17×100 cells computational mesh is shown in Figure 5.4 (a). This problem requires adding of the gravitation force in the form

$$Fg_c^{y\,l} = -c_g\, m_c^l \tag{5.2}$$

in the vertical direction y to each total corner force of subzone l of cell c. Here, c_g denotes the gravitation constant $c_g = 0.1$. The gravitation and hydrodynamical pressure forces should keep the fluid interface in rest if it would be horizontally flat. Due to the interface perturbation, the equilibrium is violated, and the instability starts to evolve. The heavier upper fluid starts to stream down to the lighter one, and on the other hand, the lighter one starts to move up.

We performed several Rayleigh-Taylor instability (RTI) tests on a computational mesh with 17×100 cells, the results in time $t = 7.0$ are presented in Figure 5.4 (b), (c), and (d). The first test in Figure 5.4 (b) is obtained by the application of the purely Lagrangian method with a high merit factor $f_m = 1.0$. For more details on the merit factor, see section 3.1.3. The high merit factor causes full influence of the subzonal pressure force, which prevents the mesh cells to perform rotational motions. This is usually correct, but in the RTI problem, the cells close to the fluids interface must move this way to evolve the instability. Thus, in this case, the strong subzonal pressure force does not allow the instability to be created. On the other hand, if the merit factor is low $f_M = 0.1$ as in Figure 5.4 (c), the cells are allowed to deform more. Because we use purely Lagrangian method, no mixing of the fluids happens, and the rotated cells are very long and narrow. The instability does not evolve again, because the fluids interface follows the edges of these long cells. In the last plot in Figure 5.4 (d), the application of the complete ALE algorithm with the low merit factor and mesh smoothing/quantity remapping process after every 10 Lagrangian steps is shown. Some diffusion is added along the fluids interface, which represents their mixing. Due to the smoothing processes, all computational cells have reasonable shapes now, and the

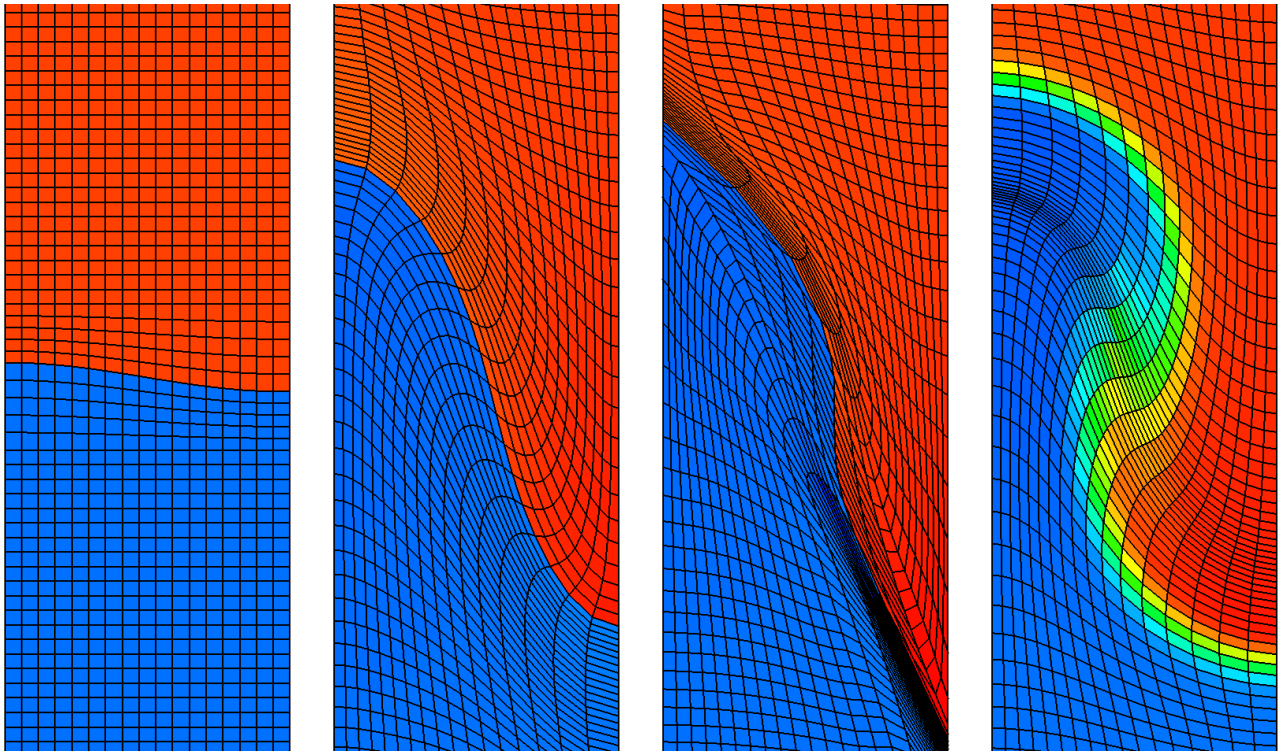


Figure 5.4: Density distribution of the Rayleigh-Taylor instability problem: (a) – initial data; (b) – Lagrangian simulation with high merit factor $f_M = 1.0$; (c) – Lagrangian simulation with low merit factor $f_M = 0.1$ (d) – ALE simulation with low merit factor $f_M = 0.1$ and smoothing/rezoning stage after each 10 Lagrangian steps. All (b), (c), (d) simulations at time $t = 7.0$.

instability starts to evolve into the mushroom shape. The simulation by the complete ALE method can continue without any problems till time $t = 8.5$, which is usually the final time of the RTI simulations. We used the shorter time, because the low merit Lagrangian simulation (c) fails shortly after $t = 7.0$. For further details about the ALE simulations of the RTI problem, see [24, 25].

In the case of the Noh problem, we have shown that the ALE method does not affect the positive properties of the Lagrangian mesh motion, both methods produce similar solutions not degraded by the used mesh smoothing techniques. On the other hand, in the case of the Rayleigh-Taylor instability problem, the Lagrangian method in our case was not able to produce reasonable results, which ALE method produces. This main advantage of the ALE approach is even more visible for the high velocity impact problems simulations, which are presented in chapter 7, where the purely Lagrangian approach fails due to severe mesh distortions, and no reasonable results are obtained without the complete ALE technique.

Chapter 6

Physical Aspects of ALE Simulations

In this chapter, we focus on several physical aspects of our ALE laser plasma simulation. In the first section 6.1, we describe the Quotidian equation of state used for the laser plasma simulations, which reasonably approximates plasma behavior in a broad range of plasma parameters. In the second section 6.2, we describe the method for solving the parabolic part of the energy equation representing the thermal conductivity in the laser plasma. Then, in section 6.3, we describe the process of the absorption of a laser beam in plasma at the critical density. Finally, in section 6.4, we describe the boundary conditions, which must be used to hold the cold Aluminum in a steady state, and to allow the expansion of melted and evaporated Aluminum. All these components are necessary for acceptable laser plasma simulations. The complete Lagrangian equations, including the thermal conductivity parabolic term, and the laser interaction source term, have the following form,

$$\frac{d\,\rho}{d\,t} + \rho\,\nabla\cdot\mathbf{w} = 0 \qquad\qquad \frac{d\mathbf{z}}{dt} = \mathbf{w} \tag{6.1}$$

$$\rho\,\frac{d\,\mathbf{w}}{d\,t} + \nabla\,p = 0 \tag{6.2}$$

$$\rho\,\frac{d\,\epsilon}{d\,t} + p\,\nabla\cdot\mathbf{w} = \nabla\cdot(\kappa\,\nabla T) - \nabla\cdot\mathbf{I}\,, \tag{6.3}$$

where κ is the thermal conductivity coefficient, T is the plasma temperature, and $\mathbf{I}$ is a vector of the laser intensity. In our model, plasma is specified as a mixture of the electron and ion gases. Due to the principle of quasineutrality, the total charge density is zero, and thus the following equation

$$n_e = Z\,n_i \tag{6.4}$$

must be satisfied, where n_e is the electron and n_i ion density, and Z is plasma mean ion charge. One-temperature model is employed, in which the electron and ion temperatures are the same, and only one common plasma temperature T is used. The two-temperature model with separate electron and ion temperatures is going to be included in the future.

6.1 Equation of State

Equation of state is a very important part of all fluid and plasma simulation codes. It is necessary for the computation of the actual fluid pressure and temperature from the actual fluid density and internal energy, and vice versa.
In the fluid dynamics, the ideal gas equation of state (IG EOS) is commonly used for this purpose. Let us describe the IG EOS for completeness. The fluid pressure is computed from the actual density and specific internal energy as

$$p = (\gamma - 1)\,\rho\,\epsilon\,, \tag{6.5}$$

where γ is the gas constant equal to the ratio of its specific heats. The sound speed is computed as

$$v^* = \sqrt{\gamma\,(\gamma - 1)\,\epsilon} \tag{6.6}$$

from the specific internal energy, and the temperature formula is

$$T = \frac{A}{Z+1}\,\frac{p}{c_{p/T}\,\rho}\,, \tag{6.7}$$

where A denotes the material atomic mass, Z is the material mean ion charge, and

$$c_{p/T} = \frac{k}{m_u} \tag{6.8}$$

is the constant for the recomputation of the pressure to the temperature. Its value (after multiplication by the conversion factor 11604 to transform the temperature in Kelvins to electronvolts) is presented in Table 6.1, together with the values of the Boltzmann constant k and the atomic mass unit m_u in

constant description	constant value
Boltzmann constant	$k = 1.3807 \cdot 10^{-16}$ erg/K
atomic mass unit	$m_u = 1.6605 \cdot 10^{-24}$ g
electron mass	$m_e = 9.1094 \cdot 10^{-28}$ g
electron charge	$e = 4.8032 \cdot 10^{-10}$ statcoul
light speed	$c = 3.0 \cdot 10^{10}$ cm/sec
electronvolt/Kelvin conversion factor	$1\,\mathrm{eV} = 11604\,\mathrm{K}$
pressure/temperature conversion factor	$c_{p/T} = 0.9648 \cdot 10^{12}$ erg/eV/g

Table 6.1: Values of the selected physical constants needed in this thesis in the CGS unit system.

the CGS unit system. The presented formulas are used for the recomputation of the state quantities between each other.
For the laser plasma simulations, the IG EOS does not provide realistic formulas for the relations among the state quantities. It can be used in hot, low density plasma of the expanding corona, but for the cold dense plasma close to the ablation surface and in the disc target with the spreading shock wave, the IG EOS is not applicable. We use the Quotidian equation of state (QEOS) introduced in [75]. It consists of three main parts:

1. The electron ionization-equilibrium equation of state based on a the Thomas-Fermi statistical cell model with the scaling property for the atomic number and the atomic weight.
2. Analytical ion equation of state combining the Debye, Gruneisen, Lindemann, and fluid-scaling laws.
3. Empirical term introducing the correction for chemical bonding, derived from physical properties of a given material.

This equation of state is valid in a broad range of laser plasma parameters, and its suitability, reasonable efficiency, accuracy, and consistency for such kind of simulations, was shown already in [31]. This EOS assumes the local thermodynamical equilibrium of electrons. The quantum shell effects, and the phase transitions are neglected. This EOS handles both the pure elements and compound materials. As the input parameters, it requires the material atomic weight, atomic number of every element, number of atoms of each element in the compound, solid state density, and the bulk modulus at solid state density describing how the total pressure changes with the changing plasma density.
Both mentioned equations of state are important. For fluid simulations, but also to first plasma tests, the IG EOS is used. This EOS is much simpler and faster, which makes it and ideal EOS for testing purposes. The final plasma simulations are then performed with the complex and slower QEOS, providing realistic results close to real plasma behavior.

6.2 Heat Conductivity

The importance of heat conductivity is different for each particular problem simulated. For some problems, such as the impact of the high-speed flyer to the massive target, the hydrodynamical effects are much stronger than the heat conductivity term, and the solutions of the same problem with and without the heat conductivity are close to each other. On the other hand, when the interaction of the laser beam with the target material is simulated, the solution without the thermal conductivity is clearly unrealistic, and the spreading shock wave is slower than the shock wave from the simulation including the thermal conductivity process. Moreover, we need a mechanism for propagating the energy from the critical surface inside the target. There are two such mechanisms possible – radiation transport and the electron heat conductivity. For the materials with low atomic number (such as Aluminum), the second mechanism is dominant, and we employ electron heat conductivity only.
Thermal conductivity effects are mathematically described by the parabolic part of the energy equation of the system (6.1),

$$\rho \frac{d\,\epsilon}{d\,t} = \nabla \cdot (\kappa \nabla T)\,. \tag{6.9}$$

This particular equation is solved separately by splitting from the hyperbolic part of the system. Generally, the specific internal energy depends on the the plasma temperature and density, and thus

$$\frac{d\,\epsilon}{d\,t} = \frac{\partial \epsilon}{\partial T}\frac{\partial T}{\partial t} + \frac{\partial \epsilon}{\partial \rho}\frac{\partial \rho}{\partial t}\,. \tag{6.10}$$

The plasma density changes only during the hyperbolic part of the splitting process, it is constant in the parabolic step, so the second term is equal to zero and can be omitted. Thus, the parabolic energy equation (6.9) can be transformed into the temperature equation

$$\rho \frac{\partial \epsilon}{\partial T}\frac{\partial T}{\partial t} = \nabla \cdot (\kappa \nabla T)\,, \tag{6.11}$$

or finally to the form

$$T_t = \frac{1}{\rho\,\epsilon_T}\,\mathrm{div}(\kappa\ \mathbf{grad}\,T)\,. \tag{6.12}$$

Here, the temperature derivative of the specific internal energy $\epsilon_T = \partial\epsilon/\partial T$ can be computed analytically for the IG EOS, or numerically for the QEOS formulation.
For the evaluation of the heat conductivity coefficient κ, the classical Spitzer-Harm [89] formula

$$\kappa = 20\,\left(\frac{2}{\pi}\right)^{3/2}\,\frac{k^{7/2}}{\sqrt{m_e}\,e^4}\,\delta_{ee}\,\frac{T^{5/2}}{Z\,\ln\Lambda} \tag{6.13}$$

corrected by the electron-electron collision term

$$\delta_{ee} = 0.095\,\frac{Z + 0.24}{1 + 0.24\,Z}\,, \tag{6.14}$$

where Z is the plasma mean ion charge and the values of the Boltzmann constant k, electron mass unit m_e, and the electron charge e are presented in Table 6.1. All quantities are supposed to be in the CGS units, except the plasma temperature T which is in electronvolts eV. Here, $\ln\Lambda$ denotes the Coulomb logarithm defined in [26] as

$$\ln\Lambda = 2.306\ \ln(\max(10, \min(\lambda_1, \lambda_2))) \tag{6.15}$$

$$\lambda_1 = 1.5526 \cdot 10^{10}\,\frac{T^{3/2}}{Z\,\sqrt{n_e}} \tag{6.16}$$

$$\lambda_2 = 8.07 \cdot 10^{10}\,\frac{T}{\sqrt{n_e}} \tag{6.17}$$

$$n_e = Z\,n_i = \frac{Z}{A}\,\frac{\rho}{m_u}\,, \tag{6.18}$$

where A is the atomic mass, n_e and n_i are densities of electrons and ions, and value of the atomic mass unit m_u is presented in Table 6.1. For more details about the heat conductivity coefficient, see [30]. Let us note, that also Rozmus-Offenberger heat conductivity coefficient [82] can used, but in practical test, we see only minor changes from the presented Spitzer-Harm formulation.
Due to the non-linear dependence of the heat conductivity coefficient on temperature, one can expect non-linear effects such as heat waves. The numerical method for solving the parabolic equation (6.11) have to be able to deal with them. The discretization of the heat conductivity equation (6.12) is treated by the mimetic method [86] using support operators [84]. The integral properties of the differential operators div and **grad** have to be satisfied also for their discrete approximations, such that the discrete operators mimic the properties of the continuous ones. We define the generalized gradient $\mathbf{G}$

$$\mathbf{Q} = \mathbf{G}T = -\kappa \, \mathbf{grad} \, T \,, \tag{6.19}$$

and extended divergence $\mathbf{D}$

$$\mathbf{DQ} = \begin{cases} \operatorname{div} \mathbf{Q} & \text{on} \quad V \\ -(\mathbf{Q}, \mathbf{n}) & \text{on} \quad \partial V \end{cases} \tag{6.20}$$

The mimetic operators have the same integral properties as the continuous ones, particularly the extended gradient is adjoint with the extended divergence, $\mathbf{G} = \mathbf{D}^*$.
Having both discrete operators D and G, a fully implicit scheme in time

$$\frac{(T^{n+1} - T^n)}{\Delta t} = a \, DGT^{n+1} \tag{6.21}$$

is used to approximate (6.12), where the coefficient $a = 1/(\rho \, e_T)$. The matrix of the global system

$$(I - \Delta t \, a \, DG)T^{n+1} = T^n \,, \tag{6.22}$$

as well as the matrix of the global system for the heat flux Q (6.19), remains symmetric and positive definite so the fast converging conjugate gradient method can be used to solve the implicit scheme (6.21). The described method is exact for piecewise linear solutions, otherwise it is second order accurate in space. It works well also on bad quality meshes appearing in Lagrangian simulations, and also for discontinuous diffusion coefficients κ.
The formulation allows to solve the global system for the heat flux Q (6.19), and simply add the heat flux limiter to the numerical scheme, reducing nonphysically large heat transfer in the material. For more details on the heat conductivity processes, see [54], [65].

6.3 Interaction with a Laser Beam

For employing the interaction of the laser beam with the material, we use the simplest model available – the laser beam penetrates the material till the critical density, which can in CGS units be written as

$$\begin{aligned} \rho_c = n_c^i \, A \, m_u = \frac{n_c^e}{Z} \, A \, m_u = \frac{1}{4\,\pi} \, \frac{m_e \, \omega^2}{Z \, e^2} \, A \, m_u = \frac{1}{4\,\pi} \, \frac{m_e \, (2\,\pi \, f)^2}{Z \, e^2} \, A \, m_u \\ = \frac{1}{4\,\pi} \, \frac{m_e \, (2\,\pi \, \frac{c}{\lambda})^2}{Z \, e^2} \, A \, m_u = \frac{\pi \, m_e \, m_u \, c^2}{e^2} \, \frac{A}{Z \, \lambda^2} \,, \end{aligned} \tag{6.23}$$

which can be written after substituting for all constants from Table 6.1 as

$$\rho_c = 1.85 \cdot 10^{-11} \, \frac{A}{Z \, \lambda^2} = 1.85 \cdot 10^{-3} \, \frac{A}{Z \, \lambda_\mu^2} \,, \tag{6.24}$$

where $f = \omega/(2\,\pi)$ is the laser frequency, c is the speed of light, A is the atomic mass, Z is the plasma mean ion charge, and λ_μ is the wavelength of the laser beam in μm. Here, at the critical density isoline, the laser beam is completely absorbed.

In the ALE code, laser beam incident normally from vacuum is implemented. Its direction in our code is from above, along the y (Cartesian) or z (cylindrical) axis. On the upper boundary, the incoming laser intensity vector has the form

$$\mathbf{I} = (0, -I_y(t,x))\,, \tag{6.25}$$

where the y intensity $I_y(t,x)$ has Gaussian profile both in time and x coordinate. Generally, the laser beam profile can be arbitrary, we use the Gaussian profile in our simulations, corresponding to the ideal laser beam. In real laser systems, the intensity shape often includes inhomogeneities, such as speckles with much higher intensity than in the surrounding regions. In the case of the PALS laser system, experiments of which we simulate in chapter 7, the laser beam profile is reasonably close to our Gaussian profile.

On each computational mesh edge, both x and y components of the laser intensity are projected to the normal of the edge. Because the isoline at critical density must move smoothly during the simulation, a special treatment must be taken to get values of laser intensity divergence at each cell. The nodal density is interpolated as a weighted (by the subzonal volumes) average of the neighboring subzonal densities. If we would use all four subzonal densities for this interpolation, we could obtain too high (supercritical) density for nodes close to the critical isoline, even if the node lies behind this isoline and the density should be subcritical. To correct this problem, we use only two neighboring subzonal densities closer to the laser source (for the laser beam incident from the upper direction, we use the upper two subzones). Now, if the densities in both these subzones are subcritical, the nodal density is then also subcritical, and the laser beam penetrates till this node, which is correct.

Now, we have density in each node interpolated from the neighboring subzones, and one can proceed to the evaluation of the laser intensities on each edge e of the particular cell c. If the density is supercritical in both nodes of the edge e, we set the edge intensity to zero $\mathbf{I}_e = \mathbf{0}$. If one density is subcritical and the second one is supercritical, the edge intersects the critical isoline. We suppose linear density profile along the edge and localize the position of the critical density ρ_c, and set the subcritical edge length $L^s(e)$ to the length of the segment with subcritical density. In the last case, when both densities are subcritical, we set the subcritical edge $L^s(e)$ to the actual length of the edge.

The divergence of the laser intensity in cell c is zero, if all four nodal densities are either subcritical or supercritical. If the values are mixed (some sub- and supercritical), we set the cell divergence to the value of the integral of the divergence over cell c divided by its volume, and after applying the Green formula, to the integral over the cell boundary ∂c. It is evaluated as the sum (over all cell edges) of the edge intensities $\mathbf{I}_e$ projected to the direction of the edge outer normal, multiplied by the subcritical edge length $L^s(e)$, and divided by the cell volume. This divergence of laser intensity is included in the internal energy equation (with laser absorption term on the right hand side)

$$\rho \frac{d\,\epsilon}{d\,t} + p\,\nabla\cdot\mathbf{w} = -C_a\,\nabla\cdot\mathbf{I}\,, \tag{6.26}$$

which transfers the laser energy to the internal energy of plasma. The laser absorption coefficient C_a is empirically estimated [64] to be equal to $C_a = 0.5$ for the first, and $C_a = 0.75$ for the third harmonic laser frequency. At the end, let us note, that the laser intensity divergence (and thus the right hand side of (6.26), causing the laser beam absorption) is non-zero only in the cells intersected by the critical density isoline.

6.4 Boundary Conditions

Implementation of the boundary conditions is an essential problem in all computational fluid dynamics and plasma physics codes. To describe this problematics, we suppose the simulation of the impact of the evaporated, ablatively accelerated Aluminum disc to the massive Aluminum target, as described in chapter 7. Naturally, purely free boundary condition cannot be applied, the solid material of the solid massive target would start to expand to the surrounding vacuum. On the other hand, we have to allow

the evaporated material to form the expanding corona freely. Let us describe the boundary conditions (BC) used in our code for the laser plasma simulations. We go through the boundary conditions for the plasma pressure, plasma velocity, BC for the computational mesh during the mesh smoothing process, and finally BC for the subzonal conservative quantities during the remapping stage.
The boundary conditions show us the main advantage of the staggered discretization. We just need one layer of zero-volume ghost cells around the whole computational domain, in which we set the pressure according to the particular BC. Then, from the pressure BC, we compute the nodal velocities during the Lagrangian step and modify them according to the velocity BC. The example of the situation on the boundary is shown in Figure 6.1. The ghost cells are shaded in gray and have non-zero volumes

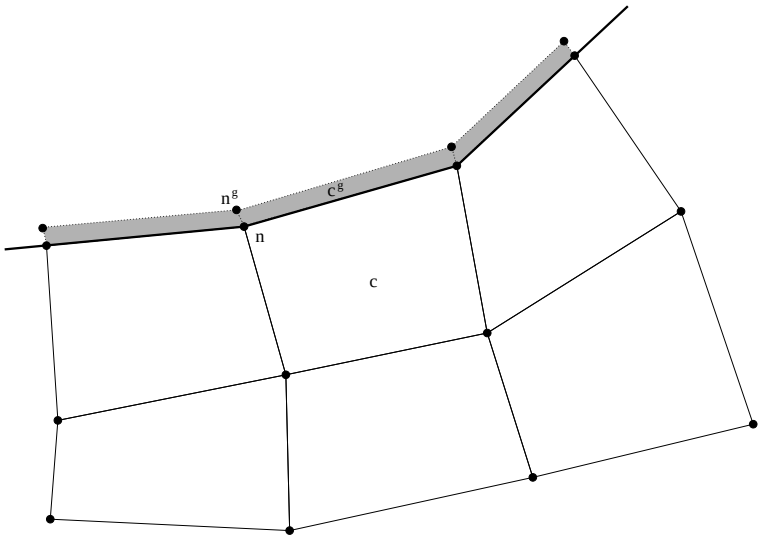


Figure 6.1: Domain boundary (thick line), a particular internal boundary cell c and the corresponding ghost cell c^g, the boundary node n and the ghost node n^g are shown. All zero-volume ghost cells are shaded in gray (due to visibility, they are depicted with non-zero volumes).

in the Figure 6.1, which is just for the visualization purposes. In fact, the fictitious ghost nodes n^g have the same coordinates as the real boundary nodes n, and the cells have zero volumes. In the cell centered Lagrangian methods (one of them is reviewed in [18]), the implementation of the BC is much more complicated, and for example the pressure boundary condition quite easy in the staggered discretization, is difficult to be implemented. Let us also note, that the BC calculation process is crucial for the simulations to correspond to the experimental results.

6.4.1 Pressure Boundary Conditions

The pressure boundary conditions determine the outer pressure, which will be used for the computation of the nodal forces, and thus new nodal velocities during the Lagrangian stage. It is a mechanism for repairing the incorrectnesses in the used equations of state, which do not give zero pressure in the solid target at the initial temperature. We use the following approach to compute the pressure in the boundary zero-volume ghost cells. To the particular ghost cell c^g, we assign the temperature from the adjacent internal cell c, which is a boundary cell. If this temperature T_{c^g} is lower than the melting temperature T^{melt} of the material ($T_{c^g} < T^{\mathrm{melt}} = 0.085\,\mathrm{eV}$ for Aluminum), we set the external ghost pressure to the adjacent internal one

$$p_{c^g} = p_c\,. \tag{6.27}$$

Thus, the pressure on boundary is constant, there is no pressure gradient across the boundary, and the nodes are not forced to move here. On the other hand, if the boundary temperature is bigger than

the boiling temperature T^{boil} of the material ($T_{c^g} > T^{\mathrm{boil}} = 0.25\,\mathrm{eV}$ for Aluminum), we set the ghost pressure to the atmospheric pressure

$$p_{c^g} = p_{\mathrm{atm}} = 101325\,\mathrm{Pa} = 1013250\,\mathrm{dyne/cm^2}\,, \tag{6.28}$$

or to a very low value

$$p_{c^g} = p_{\mathrm{vac}} = 1\,\mathrm{dyne/cm^2} \tag{6.29}$$

for vacuum $10^{-4}\,\mathrm{torr}$, according to the particular experiment. All our plasma simulations in section 7 reproduce experiments performed in vacuum, so the second value (6.29) is used. This BC allows the plasma to expand to the outer space. In the last case, when the material is liquid ($0.085\,\mathrm{eV} \leq T_{c^g} \leq 0.25\,\mathrm{eV}$ for Aluminum), we linearly interpolate the pressure value as a function of the actual temperature,

$$p_{c^g} = p_{\mathrm{vac}} + \frac{T_{c^g} - T^{\mathrm{boil}}}{T^{\mathrm{melt}} - T^{\mathrm{boil}}}\,(p_c - p_{\mathrm{vac}})\,. \tag{6.30}$$

This linear approximation causes the boundary condition to behave smoothly through the liquid plasma region, and no jumps appear. For completeness, let us note, that the described pressure BC is used for the irradiated/impacted domain boundary only. For all other boundary segments, solid boundary

$$p_{c^g} = p_c \tag{6.31}$$

is used, which does not allow the plasma to move outside the initial domain by a different process, than the impact corona. This is allowed by the fact, that the computational domain is always taken large enough, such that the generated shock wave does not reach other boundary segments, except on the z axis where solid and reflecting BC are the same.

6.4.2 Velocity Boundary Conditions

The velocity boundary conditions are applied whenever the nodal velocities are changed, mainly after computing the new nodal velocities during the Lagrangian step. The velocity BC sets the nodal velocities in the boundary nodes, i.e. nodes between the internal boundary cells and the ghost cells described in the previous section.
For the described disc impact problem, we suppose solid wall around the whole computational domain, excluding the upper boundary segment, where the accelerated disc impacts the massive target. For more details about the impact problem initial conditions, see chapter 7. The solid wall boundary conditions are implemented by setting the boundary nodal velocity in the normal direction to zero. The velocities computation takes care of the nodal movement along the boundary line, but this BC does not allow the nodes to move perpendicularly to the boundary. On the upper boundary, we again set the boundary condition according to the temperature in the internal cell c, which the particular node n belongs to. If the temperature is bigger than the melting temperature of the material ($0.085\,\mathrm{eV}$ for Aluminum), the material is allowed to move, the values in the boundary nodes are computed during the Lagrangian stage from the pressures in the ghost cells set during the pressure BC process described in the previous section. If the temperature is smaller, the material is still solid, and we overwrite the computed velocities by the zero normal velocity. This process successfully works for a wide range of laser plasma simulations, and produces reasonable behavior of all quantities close to the boundary.

6.4.3 Smoothing Mesh Boundary Conditions

During the mesh smoothing stage, there is no unique approach what to do on the boundary. Our approach, how to deal with the boundary during the smoothing stage, is its smoothing as there would be no boundary. In the first step, we extrapolate the external nodes of the zero-volume ghost cells, such that we extend the internal edge e connected to the boundary through the node n, and move the fictitious

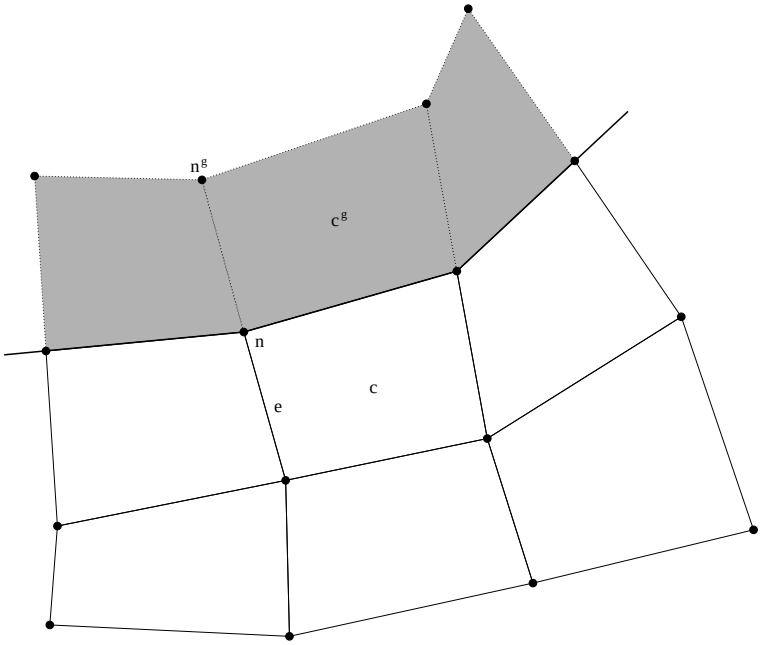


Figure 6.2: Domain boundary (thick line), a particular internal boundary cell c and the corresponding extended ghost cell c^g, the boundary node n and the ghost node n^g projected by extension of the internal edge e connected to boundary, are shown. All extended ghost cells are shaded in gray.

node n^g to the same distance from the boundary, as the node on the opposite side of the edge e. The final situation is shown in Figure 6.2. Now, the smoothing process moving the nodes can be applied to the mesh, including the boundary nodes n. It uses positions of the extrapolated nodes n^g, which are not moved by the smoothing process. The extrapolated ghost nodes n^g cause, that the situation out of the domain is similar to the situation inside, and the smoothing process does not move the boundary nodes too much perpendicularly to the boundary shape. However, they can move along the boundary, and the mesh is smoothed also in the boundary regions, the boundary shape remains more or less similar, no major roughness of the boundary is removed nor created. Finally, as the last step, the ghost nodes positions n^g are set to the positions of the smoothed nodes n to return the zero-volume ghost cells back. The described process preserves the notion of the boundary shape, and cooperates well with all types of mesh smoothing techniques described in section 3.2.

6.4.4 Boundary Conditions for Remapping

In the remapping process, the boundary values are important only for the computation of the local extrema $g_c^{\min}$, $g_c^{\max}$ in the boundary cells c, which are necessary for the reconstructions stage, and also for the repair redistribution stage. The mass movement through the boundary is clearly not allowed (the method would violate its conservativity), thus there is no problem with the boundary in the integration stage. The only issue about the boundary in the remapping process is the computation of the extremes. Let us focus on the reconstruction stage, the situation during the repair stage is similar. The local extrema have to be calculated from the function mean values in a good neighborhood of cell c, which according to [90] means, that the whole cell c (including all its vertices) must lie within the convex hull of the neighborhood cell centers. If some of the cell vertices lies outside of this hull, it may happen, that the computed local extrema are more restrictive than necessary. The limitation process is then applied, and linearity preservation condition is violated. For example in the Figure 6.1, this obviously happens for a global linear function $g(x, y) = y$, when we omit the ghost cells from the neighborhood. The node with the highest y coordinate has the highest function value, but the local maximum is computed from

the cell on right from cell c, which is smaller than the nodal value. Thus, the local maximum is lower than the value in the top node, the limiter is applied, and the linearity preservation is violated. In the case, that we include the gray shaded ghost cells to the neighborhood, the local maximum is reached in one of these ghost cells, no limiting is required, and no problems with the linearity preservation condition appear.

We still have to define, how to calculate the mean value of the function g in the center of the ghost cell c^g, from the mean values in the centers of the internal cells. We only require this ghost value to satisfy the global linearity preservation condition. Due to this fact, a simple linear extrapolation from the internal cells is satisfactory, the defined ghost mean value is exact for the global linear function and sufficient for the non-linear function. The described process successfully works in our fluid simulations, but also in the complicated laser plasma simulations, where correct BC play a crucial role to find a realistic solution.

Chapter 7

Numerical Simulations of Laser Plasma Experiments

We have implemented all described numerical methods into a complete ALE code, which can be used for the fluid and laser plasma hydrodynamical simulations. In this chapter, we present several numerical simulations of the laser plasma experiments performed at the PALS (Prague Asterix Laser System) laser facility [44]. Results of these experiments were published in [14, 13, 80, 45, 39]. In the first paper, the experiments of the interaction of the intense laser beam with a massive target are described. In the rest of the papers, laser interaction with small disc flyer are performed. The setup of this double target experiment is shown in Figure 7.1. A small Aluminum disc flyer is irradiated by a laser beam of intensity

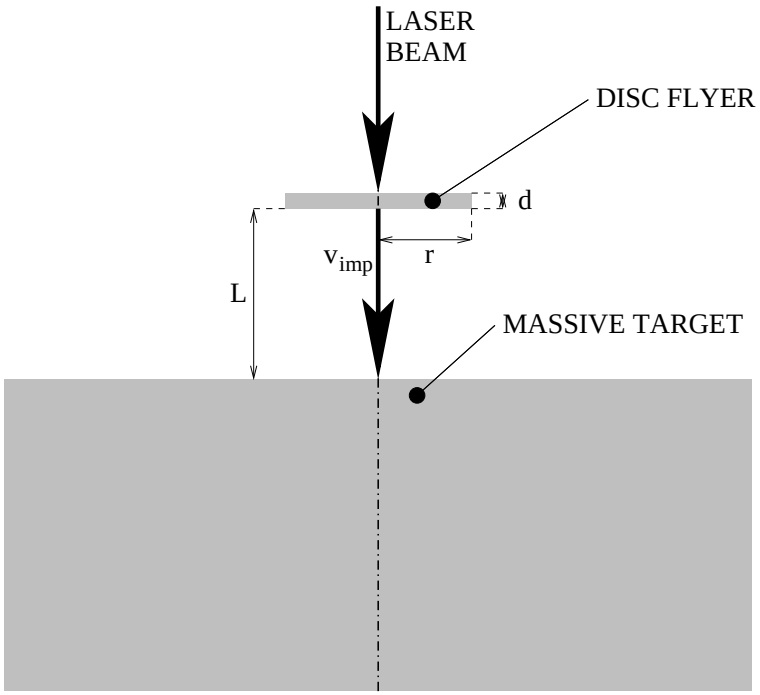


Figure 7.1: Experimental setup – laser beam irradiates a disc flyer of radius r and width d in distance L from the massive target, disc flyer is ablatively accelerated to the impact velocity v_{imp} and strikes the massive target.

I. The outer part of the disc is evaporated, it expands, and its density is decreased. The rest of the disc is ablatively accelerated up to the velocity v_{imp}, flies over the distance L, and strikes the massive

Aluminum target. By the impact, the target is heated, melted, and evaporated, and a crater is formed. The evaporated material expands in the form of corona out of the massive target.
This process is modeled in our simulations in two steps. In the first step, the interaction of the laser beam with the disc, causing its expansion and acceleration, is simulated. This simulation provides the density, temperature, and velocity distribution in the flyer disc in time of the impact. Then, this distributions are conservatively interpolated to the initial mesh of the second part of the simulation – the impact process. The flying disc strikes the massive target, a shock wave is formed moving from the impact region into the target, causing its heating, melting, and evaporation. Low density corona is formed, moving in high speed out of the impact region.
The experimental data include energy of laser pulse, pulse duration, diameter of the laser spot on target, and estimate of pulse temporal shape type and of focal intensity profile. In section 7.1, we derive effective laser intensity for 1D and 2D planar and cylindrical simulations from experimental data. Comparison of the derived intensities is also presented. In section 7.2, the simulations of the interaction of the laser beam with a massive Aluminum target are presented. In the following two sections 7.3, and 7.4, we describe the simulations of the double target experiments, which we perform in two phases. The simulations results are compared with the experimental data. We focus on one particular simulation and show its status in different phases of the simulation. Finally, in section 7.5 we summarize, how the absorbed laser energy is distributed into kinetic and internal energy, which is important for the relevance of the simulations.

7.1 Derivation of Maximum Laser Intensity from Experimental Data

In this section, we derive the formulas for the maximal laser beam intensity, which is used during the laser absorption process (6.25). The laser beam intensity $I(t)$ profile in time is shown in Figure 7.2. The

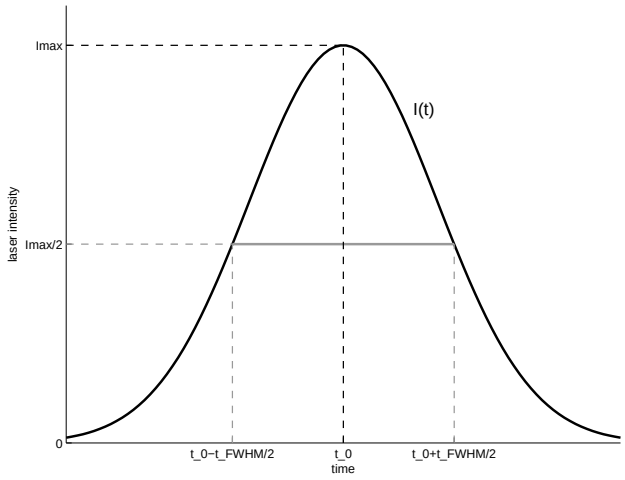
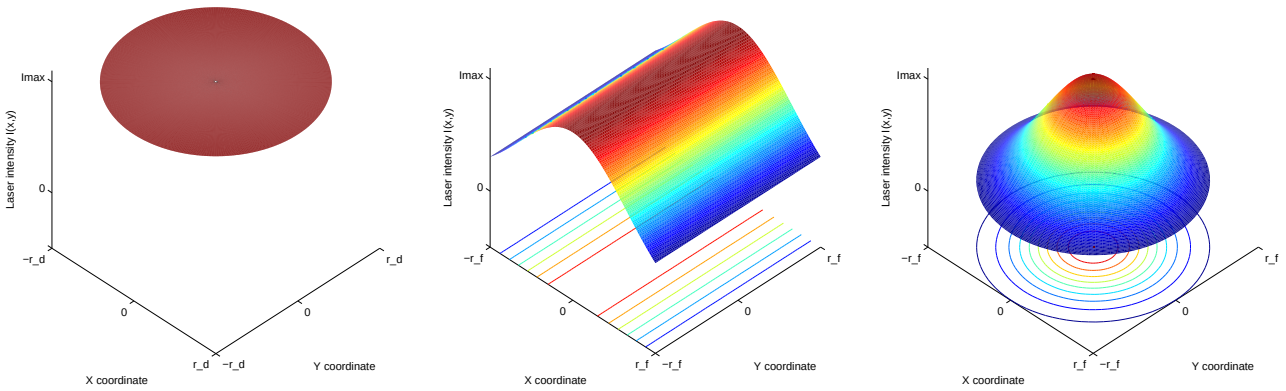


Figure 7.2: Laser beam intensity $I(t)$ Gaussian profile in time. Both important quantities are denoted – maximal laser beam intensity $I_{\max}$ and full-width at half-maximum pulse duration t_{FWHM}.

standard experimental data for pulse duration is thelength of time interval inside which the intensity is above one-half of the maximum intensity $I_{\max}$. This is called Full-Width at Half-Maximum pulse duration t_{FWHM}, and pulse duration means t_{FWHM} throughout this thesis.
It is assumed that the temporal profile of the laser pulse has shape near to Gaussian [44]. We also assume Gaussian spatial profile of the focused laser beam [44]. This is certainly an approximation, however, more details are not known as the exact profile of intense focused beam is difficult to measure. In Figure 7.3, we present the pulse profile in a particular time t shown in the whole disc region in 1D (a),



(a) (b) (c)

Figure 7.3: Gaussian profile of the laser beam intensity I in space, friezed at the particular time $t = t_0$. In 1D (a), the laser intensity is constant and equal to $I_{\max}$ in the whole disc region of diameter r_d. In 2D Cartesian geometry (b), the pulse profile depends on x coordinate only, so the maximal intensity $I_{\max}$ is reached on the line in y direction in the center of the laser spot on target defined as a square with edge length $2\,r_f$ in both directions, where r_f is the radius laser (focal) spot on target. In 2D cylindrical geometry (c), the intensity pulse profile in the laser spot on target depends on the distance from the focus center, where the maximal intensity value $I_{\max}$ is reached.

and in the laser spot on target in 2D Cartesian (b) and cylindrical (c) geometries. The maximal laser intensity $I_{\max}$ is reached in the whole disc in 1D, on one line in 2D Cartesian case, and in the center in 2D cylindrical case. The total energy of the laser beam (integral of the laser beam intensity profile) must remain the same, so the maximal intensity depends on the geometry used – it is lowest in 1D case and highest in the 2D cylindrical case.
In this section, we derive the formula for the $I_{\max}$ parameter evaluation in 1D, and in 2D Cartesian and cylindrical coordinates. Finally, we compare all three formulas and evaluate the maximal intensity values used for the particular laser plasma simulations.

7.1.1 Maximal Laser Intensity in 1D

The derivation of the formula for the maximal laser intensity $I_{\max}$ in 1D is shown here in more details. In the next two sections, similar formulas are derived for 2D Cartesian and cylindrical geometry by analogous process, where the details are omitted.
In 1D, the laser beam intensity depends on time only, there is no spatial dependence. The time profile is shown in Figure 7.2, and it is described by

$$I(t) = I_{\max}\, e^{-\left(\frac{t-t_0}{\tau}\right)^2}, \tag{7.1}$$

where the pulse position t_0 can be set to zero without the loss of generality. The total pulse energy in the disc region is equal to the integral of the intensity profile both in time and space. In 1D, the circular space integration is used (following the cylindrical shape of the laser beam),

$$E_d = \int_{-\infty}^{\infty} \int_{0}^{r_d} \int_{0}^{2\,\pi} r\, I(t)\, d\varphi\, dr\, dt = I_{\max}\, \tau\, \sqrt{\pi}\, \pi\, r_d^2\,, \tag{7.2}$$

where the symbol r_d denotes the disc radius. The parameter τ has to be determined from the condition $I\big(t_{\mathrm{FWHM}}/2\big) = I_{\max}/2$ defining t_{FWHM} and giving $\tau = t_{FWHM}/\big(2\,\sqrt{\ln(2)}\big)$.

We also need to estimate the ratio of how much laser energy strikes upon the disc. Let us suppose the Gaussian intensity profile in both space and time

$$I(r,t) = I_{\max}\, e^{-\left(\frac{t}{\tau}\right)^2}\, e^{-\left(\frac{r}{r_0}\right)^2}\,. \tag{7.3}$$

The total laser pulse energy can be computed as

$$E^{\mathrm{L}} = \int_{-\infty}^{\infty}\int_{0}^{\infty}\int_{0}^{2\pi} r\, I(r,t)\, d\varphi\, dr\, dt = I_{\max}\, t_{\mathrm{FWHM}}\, \sqrt{\pi}\, \pi\, r_0^2\,, \tag{7.4}$$

and the energy striking upon the laser spot circle of radius r_f as

$$E_f = \int_{-\infty}^{\infty}\int_{0}^{r_f}\int_{0}^{2\pi} r\, I(r,t)\, d\varphi\, dr\, dt = I_{\max}\, t_{\mathrm{FWHM}}\, \sqrt{\pi}\, \pi\, r_0^2 \left(1 - e^{\left(-\frac{r_f^2}{r_0^2}\right)}\right)\,. \tag{7.5}$$

The laser spot on target (focal spot) is defined as a circle including 80 % of the laser beam energy, $0.8\, E^{\mathrm{L}} = E_f$. After substituting the formulas for both energies to the equation, one can express the parameter $r_0 = r_f/\sqrt{\ln(5)}$. The coefficient of the energy included in the disc can be written as

$$C_d = \frac{E_d}{E^{\mathrm{L}}} = 1 - 5^{\left(-\frac{r_d^2}{r_f^2}\right)}\,. \tag{7.6}$$

If the laser spot radius is the same as the disc radius $r_f = r_d$, the coefficient C_d has the value 0.8. For the particular experimental values of $r_f = 125\,\mu$m, $r_d = 150\,\mu$m from [45], the coefficient C_d is equal to about 0.9. The final formula for the maximal laser intensity is then

$$I_{\max} = C_a\, C_d\, \frac{2\,\sqrt{\ln(2)}}{\sqrt{\pi}}\, \frac{E^{\mathrm{L}}}{t_{\mathrm{FWHM}}\, \pi\, r_d^2}\,. \tag{7.7}$$

We have added the absorption coefficient C_a for Aluminum, which is estimated from [64] to be $C_a = 0.5$ for the first harmonic and $C_a = 0.75$ for the third harmonic frequency.

7.1.2 Maximal Laser Intensity in 2D Cartesian Geometry

In 2D Cartesian coordinates, the laser pulse profile depends on time and x coordinate, and it does not depend on y coordinate

$$I(x,y,t) = I_{\max}\, e^{-\left(\frac{t}{\tau}\right)^2}\, e^{-\left(\frac{x}{x_0}\right)^2}\,, \tag{7.8}$$

where τ and x_0 parameters have to be defined again, similarly as in 1D. After the integration of the intensity formula, we get the energy inside the laser spot on target

$$E_f = \int_{-\infty}^{\infty}\int_{-r_f}^{r_f}\int_{-Y}^{Y} I(x,y,t)\, dy\, dx\, dt = 2\, I_{\max}\, \tau\, Y\, \pi\, x_0\, \mathrm{erf}\left(\frac{r_f}{x_0}\right)\,, \tag{7.9}$$

where $\mathrm{erf}(x) = \int_0^x e^{-\xi^2}\, d\xi$ is the error function. After evaluating of parameters $\tau = t_{FWHM}/\left(2\,\sqrt{\ln(2)}\right)$ and $x_0 = r_f/x_f$ (where $x_f \approx 0.906$ is the root of equation $5\,\mathrm{erf}(x_f) = 4$), and setting the Y limit to $Y = r_f$ to get the square laser spot on target, we get the final formula

$$I_{\max} = C_a\, x_f\, \sqrt{\ln(2)}\, \frac{E^{\mathrm{L}}}{t_{\mathrm{FWHM}}\, \pi\, r_f^2} \tag{7.10}$$

for the maximal laser beam intensity. As in 1D case, we added the absorption coefficient C_a equal either to 0.5 of 0.75, depending on the particular laser beam frequency.

7.1.3 Maximal Laser Intensity in 2D Cylindrical Geometry

In 2D cylindrical geometry, the laser beam intensity pulse has the following profile

$$I(r,\varphi,t) = I_{\max}\, e^{-\left(\frac{t}{\tau}\right)^2}\, e^{-\left(\frac{r}{r_0}\right)^2}\,. \tag{7.11}$$

The pulse does not depend on the angle φ, it only depends on time and the distance from the focus center. By integrating the intensity, we compute the energy in the laser spot on target

$$E_f = \int_{-\infty}^{\infty}\int_{0}^{r_f}\int_{0}^{2\pi} r\, I(r,\varphi,t)\, d\varphi\, dr\, dt = I_{\max}\,\tau\,\sqrt{\pi}\,\pi\, r_0^2 \left(1 - e^{\left(-\frac{r_f^2}{r_0^2}\right)}\right)\,. \tag{7.12}$$

As in the previous geometries, we have to calculate the $\tau = t_{FWHM}/\left(2\sqrt{\ln(2)}\right)$ and $r_0 = r_f/\sqrt{\ln(5)}$ parameters, and after their substitution to the previous formula, we get the final formula

$$I_{\max} = C_a\, \frac{2\,\ln(5)\,\sqrt{\ln(2)}}{\sqrt{\pi}}\, \frac{E^{\mathrm{L}}}{t_{\mathrm{FWHM}}\,\pi\, r_f^2} \tag{7.13}$$

for the laser beam maximal intensity in 2D cylindrical coordinates. Again, we have added the C_a coefficient of absorption depending on the particular harmonic of the laser system used.

7.1.4 Comparison of Maximal Laser Intensity Formulas

In this section, we compare the formulas for the maximal laser beam intensity $I_{\max}$ evaluated in all three cases – in 1D (7.7), in 2D Cartesian geometry (7.10), and in 2D cylindrical geometry (7.13).
The 2D Cartesian (7.10) and cylindrical (7.13) formulas are very similar to each other and include the same quantities, thus one can directly study their ratio to compare them. The following equation

$$\frac{I_{\max}^{\mathrm{cyl}}}{I_{\max}^{\mathrm{Cart}}} = 2.004 \tag{7.14}$$

says, that the cylindrical maximal intensity is about twice bigger, than the Cartesian one – universally, under all conditions. The 1D formula (7.7) includes the additional coefficient C_d of the total energy inside the disc region, depending on the ratio of the disc radius r_d and laser spot radius r_f. If we suppose their equality $r_d = r_f$, then the coefficient C_d is equal to 0.8, and the ratio of the cylindrical and 1D formulas is

$$\frac{I_{\max}^{\mathrm{cyl}}}{I_{\max}^{\mathrm{1D}}} = 2.012\,. \tag{7.15}$$

In the case of $r_f = 125\,\mu$m and $r_d = 150\,\mu$m from [45], the ratio is equal to $I_{\max}^{\mathrm{cyl}}/I_{\max}^{\mathrm{1D}} = 2.571$. These ratios confirm our demand of the lowest 1D and highest 2D cylindrical laser beam intensity. Let us present values of the maximal laser intensities corresponding to the specific values of laser beam energies, radius of laser spot on target (focal spot), and pulse duration of the particular experiments. In the following sections, we use the presented values for our laser irradiation simulations.
In [45], the laser beam of $t_{\mathrm{FWHM}} = 400\,$ps pulse duration and $r_f = 125\,\mu$m laser spot radius operating in the first and third harmonics (wavelengths $\lambda_1 = 1.315\,\mu$m and $\lambda_3 = 0.438\,\mu$m) irradiates the Aluminum disc of radius $r_d = 150\,\mu$m. The laser beam has the total energy $E^{\mathrm{L}} = 120\,$J, $E^{\mathrm{L}} = 240\,$J, or $E^{\mathrm{L}} = 390\,$J, depending on the particular experiment. In [45], [80], several experiments with slightly modified parameters are described. The laser beam operates with 130 J energy in the pulse. In Table 7.1, we present maximal laser intensities in all three geometries, which we use for the particular simulations.
Finally in [14], several experiments with laser irradiated massive target are described. The laser beam operates with the energy either 100 J or 600 J, depending on the experiment. Both first and third

first harmonic: $\lambda_1 = 1.315\,\mu$m

E^{L}	$I^{\mathrm{1D}}_{\mathrm{max}}$	$I^{\mathrm{Cart}}_{\mathrm{max}}$	$I^{\mathrm{cyl}}_{\mathrm{max}}$
120 J	$0.1797\,10^{15}$	$0.2305\,10^{15}$	$0.4620\,10^{15}$
240 J	$0.3594\,10^{15}$	$0.4611\,10^{15}$	$0.9240\,10^{15}$
390 J	$0.5841\,10^{15}$	$0.7493\,10^{15}$	$1.5016\,10^{15}$
130 J	$0.1947\,10^{15}$	$0.2498\,10^{15}$	$0.5005\,10^{15}$

third harmonic: $\lambda_3 = 0.438\,\mu$m

E^{L}	$I^{\mathrm{1D}}_{\mathrm{max}}$	$I^{\mathrm{Cart}}_{\mathrm{max}}$	$I^{\mathrm{cyl}}_{\mathrm{max}}$
120 J	$0.2696\,10^{15}$	$0.3458\,10^{15}$	$0.6930\,10^{15}$
240 J	$0.5391\,10^{15}$	$0.6916\,10^{15}$	$1.3861\,10^{15}$
390 J	$0.8761\,10^{15}$	$1.1239\,10^{15}$	$2.2524\,10^{15}$
130 J	$0.2920\,10^{15}$	$0.3746\,10^{15}$	$0.7508\,10^{15}$

Table 7.1: Maximal laser intensities (in W/cm^2) in 1D, and in 2D Cartesian and cylindrical geometry for 120 J, 130 J, 240 J, and 390 J laser beam operating in the first and third harmonics. The pulse duration is 400 ps, laser spot radius 125 μm, and disc flyer radius 150 μm.

No.	laser mode	r_f	E^{L}	$I^{\mathrm{e}}_{\mathrm{max}}$	$I^{\mathrm{cyl}}_{\mathrm{max}}/C_a$	$I^{\mathrm{Cart}}_{\mathrm{max}}$	$I^{\mathrm{cyl}}_{\mathrm{max}}$
1	first harmonic	35	100	$0.65\,10^{16}$	$0.982\,10^{16}$	$0.245\,10^{16}$	$0.491\,10^{16}$
2	first harmonic	100	100	$0.80\,10^{15}$	$0.120\,10^{16}$	$0.300\,10^{15}$	$0.602\,10^{15}$
3	first harmonic	300	100	$0.88\,10^{14}$	$0.134\,10^{15}$	$0.333\,10^{14}$	$0.668\,10^{14}$
4	first harmonic	600	100	$0.22\,10^{14}$	$0.334\,10^{14}$	$0.834\,10^{13}$	$0.167\,10^{14}$
5	first harmonic	600	600	$0.13\,10^{15}$	$0.200\,10^{15}$	$0.500\,10^{14}$	$0.100\,10^{15}$
6	third harmonic	35	100	$0.65\,10^{16}$	$0.982\,10^{16}$	$0.368\,10^{16}$	$0.737\,10^{16}$
7	third harmonic	100	100	$0.80\,10^{15}$	$0.120\,10^{16}$	$0.450\,10^{15}$	$0.902\,10^{15}$
8	third harmonic	300	100	$0.88\,10^{14}$	$0.134\,10^{15}$	$0.500\,10^{14}$	$0.100\,10^{15}$
9	third harmonic	600	100	$0.22\,10^{14}$	$0.334\,10^{14}$	$0.125\,10^{14}$	$0.251\,10^{14}$

Table 7.2: Maximal laser intensities (in W/cm^2) in 2D Cartesian and cylindrical geometry for 100 J and 600 J laser beam operating in the first ($\lambda_1 = 1.315\,\mu$m) and third ($\lambda_3 = 0.438\,\mu$m) harmonic. The pulse duration is 400 ps, laser spot radius varies from 35 μm to 600 μm. Enumeration of experiments from [14] is used. Comparison with the experimental intensities $I^{\mathrm{e}}_{\mathrm{max}}$ and the cylindrical intensity without the absorption factor C_a, $I^{\mathrm{cyl}}_{\mathrm{max}}/C_a$, is presented. Laser energies E^{L} are in Joules, laser spot radius r_f in μm.

harmonic frequencies are used, and the radius of laser spot on target varies from 35 μm to 600 μm. The article also includes the measured maximal laser beam intensities, which we include into Table 7.2. The value of the cylindrical intensity without the absorption coefficient have been also included into the Table 7.2. This computed maximal intensity is much bigger than the experimental data. This is caused by the fact, that in [14] it was assumed, that the pulse has constant intensity value $I^{\mathrm{e}}_{\mathrm{max}}$ in the whole laser spot on target and pulse duration time. The pulse energy in the laser spot on target is equal to 80 % of the total laser beam energy for all computed values, and 100 % in the case of the experimental value. It can be derived that the ratio of the experimental and computed intensity values is constant $I^{\mathrm{cyl}}_{\mathrm{max}}/(C_a\, I^{\mathrm{e}}_{\mathrm{max}}) \approx 1.5$. In the simulations, we of course use the values including the C_a coefficient to take the partial reflection of the laser beam into account.

7.2 Massive Target Irradiation by Laser Beam

In this section, we present several simulations of a massive target irradiation by an intense laser beam with the following parameters – pulse duration (full-width at half-maximum (FWHM) time) $t_{\mathrm{FWHM}} = 400$ ps, laser beam energy is either $E^{\mathrm{L}} = 100$ J or $E^{\mathrm{L}} = 600$ J, and the laser system is operating either in the first harmonic ($\lambda_1 = 1.315\,\mu$m) or in the third harmonic ($\lambda_3 = 0.438\,\mu$m), depending on the particular experiment. Nine sets of parameters corresponding to parameters performed on the PALS iodine laser facility are described in [14], these parameters are presented in Table 7.2. The radius of laser spot on target varies from 35 μm to 600 μm. The laser beam melts and evaporates the target material (Aluminum

is used as the target), shock wave is moving from the irradiated region inside the target. In [14], the experimental crater width and depth for all experiments is shown. Here, we compare our results from the ALE simulations, with the experimental results.

For the simulations, we use the ALE algorithm described in the previous sections, including the interaction of the laser beam with the material and Spitzer-Harm formulation of the thermal conductivity process. The mesh smoothing/remapping process is not performed regularly, it is used only if "something wrong" happens – negative volume subzone appears, or cell density is too low, or similar problem appears.

We present here the simulation of problem 2 (according to Table 7.2 and [14]), starting 400 ps before the intensity maximum till time 70 ns after the laser pulse. In Figure 7.4 (a), (b), density colormap in the whole computational domain is shown. The computational mesh is projected to the negative r halfplane

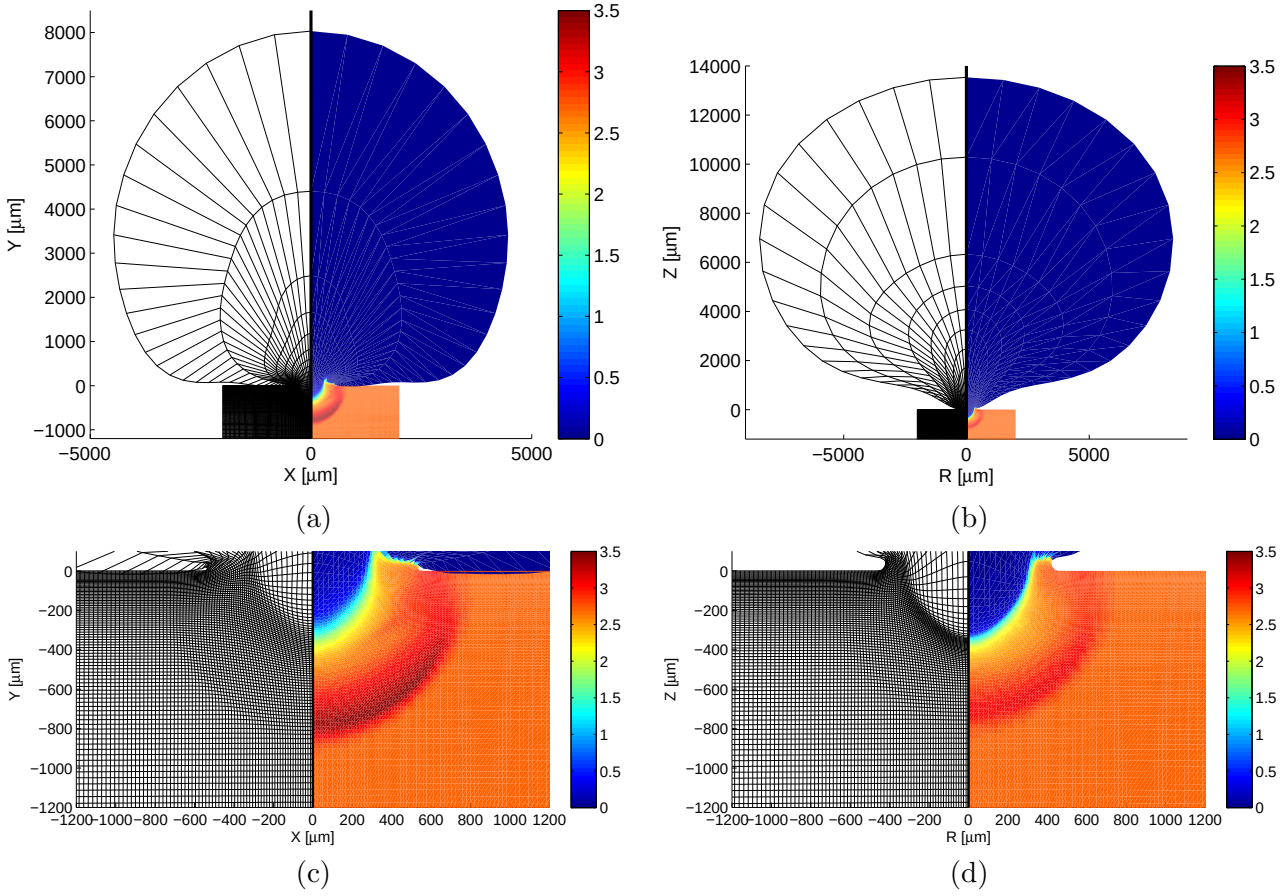


Figure 7.4: Computational mesh and density colormap (in g/cm^3) for laser interaction with massive Aluminum target (problem 2, basic laser frequency) in time 70 ns after the laser pulse. Views to the whole computational domain (a), (b) and details of the crater region (c), (d) are presented, obtained by the Cartesian (a), (c) and cylindrical (b), (d) simulations by complete ALE method.

for better insight. After the laser pulse, shock wave is created moving inside the Aluminum material. One can see it as a high density circular structure in the lower part of the Figure 7.4 (a), (b). In the region, where the shock wave did not come yet, the density is equal to the density of solid Aluminum $\rho = 2.7\,\mathrm{g/cm^3}$. Low density and high temperature corona of evaporated Aluminum is streaming from the region of irradiation (upper part in Figure 7.4 (a), (b)). The corona structure is enormous, when compared with the original computational domain. This is the reason, why Eulerian computational method is not suitable here. The Lagrangian method naturally treats the moving corona boundary.

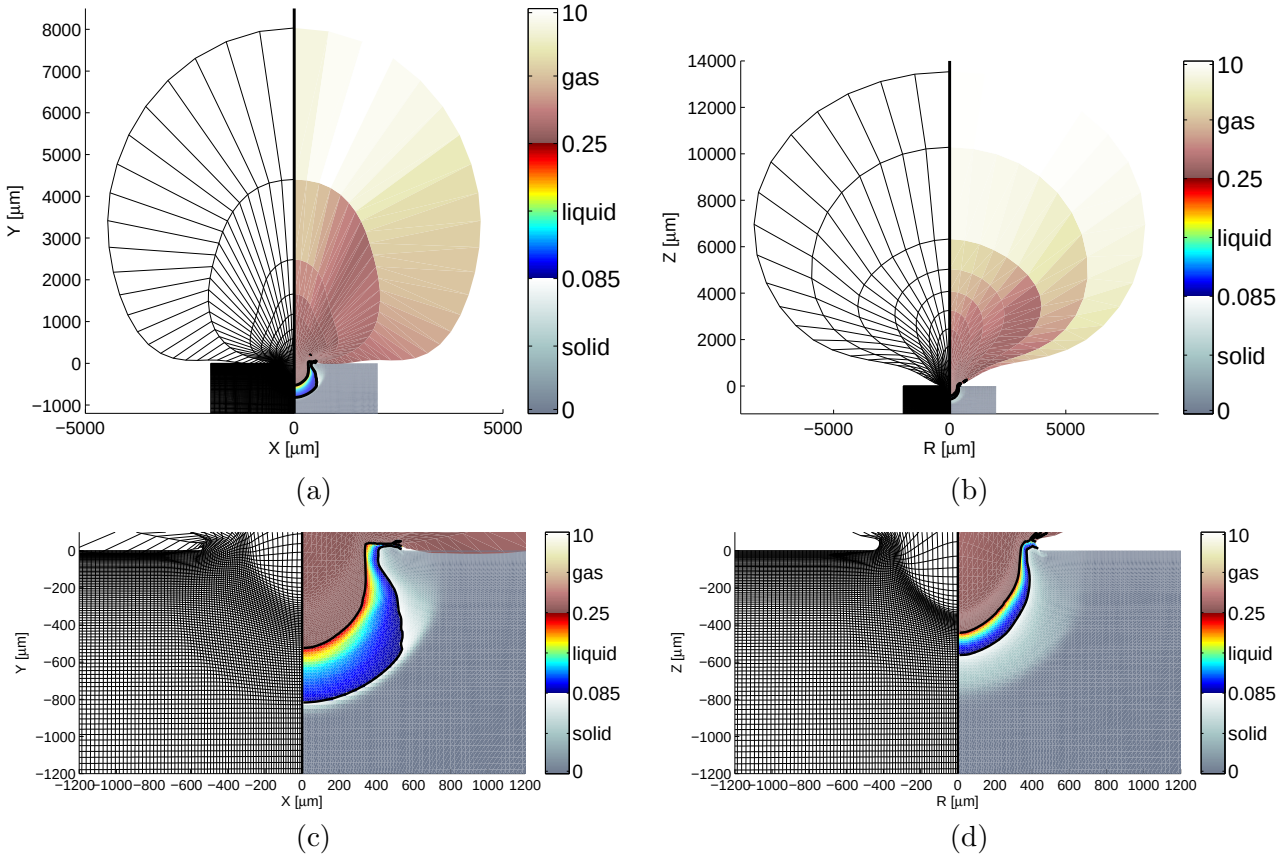


Figure 7.5: Computational mesh and temperature colormap (in eV) for laser interaction with massive Aluminum target (problem 2, basic laser frequency) in time 70 ns after the laser pulse. Views to the whole computational domain (a), (b) and details of the crater region (c), (d) are presented, obtained by the Cartesian (a), (c) and cylindrical (b), (d) simulations by complete ALE method. Solid, liquid, and gas phases are separated by the temperature isolines of melting $T^m_{Al} = 0.085\,\mathrm{eV}$ and evaporation $T^v_{Al} = 0.25\,\mathrm{eV}$ of Aluminum, different material phases are shown in different colormaps.

Zoom to the crater region is shown in Figure 7.4 (c), (d), where the shock wave is visible both in density and in the computational mesh. In the Cartesian geometry, the shock wave is moving faster, and is stronger (the density peak is higher), than in the cylindrical geometry.
In Figure 7.5 (a), (b), temperature colormap of the same problem is shown. Isolines in the temperature of Aluminum melting $T^m_{Al} = 933\,\mathrm{K} = 0.085\,\mathrm{eV}$ and boiling $T^v_{Al} = 2792\,\mathrm{K} = 0.25\,\mathrm{eV}$ separate solid, liquid, and gas phases of the material. For better insight, Aluminum phases are depicted in different colormaps. In the region below the shock wave, the temperature is equal the room temperature $T^r = 0.03\,\mathrm{eV} \approx 300\,\mathrm{K}$, the material is not heated yet. The temperature inside the low density corona is much hotter than the evaporation temperature of Aluminum. In Figure 7.5 (c), (d), zoom to the crater region is shown again. By the crater, we mean in this context the interface between the liquid and gas material phases. The evaporated gas escapes in the form of corona away, the melted material is cooled down after some time, becomes solid again, and forms the crater shape. In the experiment described in [14], the crater of problem 2 is 340 μm deep and has radius 280 μm. From the gas/liquid isoline, we estimate our simulated crater depth to 520 μm in Cartesian and 440 μm in cylindrical coordinates, and its radius to 340 μm in both geometries. As one can see, the cylindrical simulation approximates the real experiment better, than the Cartesian one.
On the other hand, both simulated craters are bigger (deeper) than the experimental craters. This

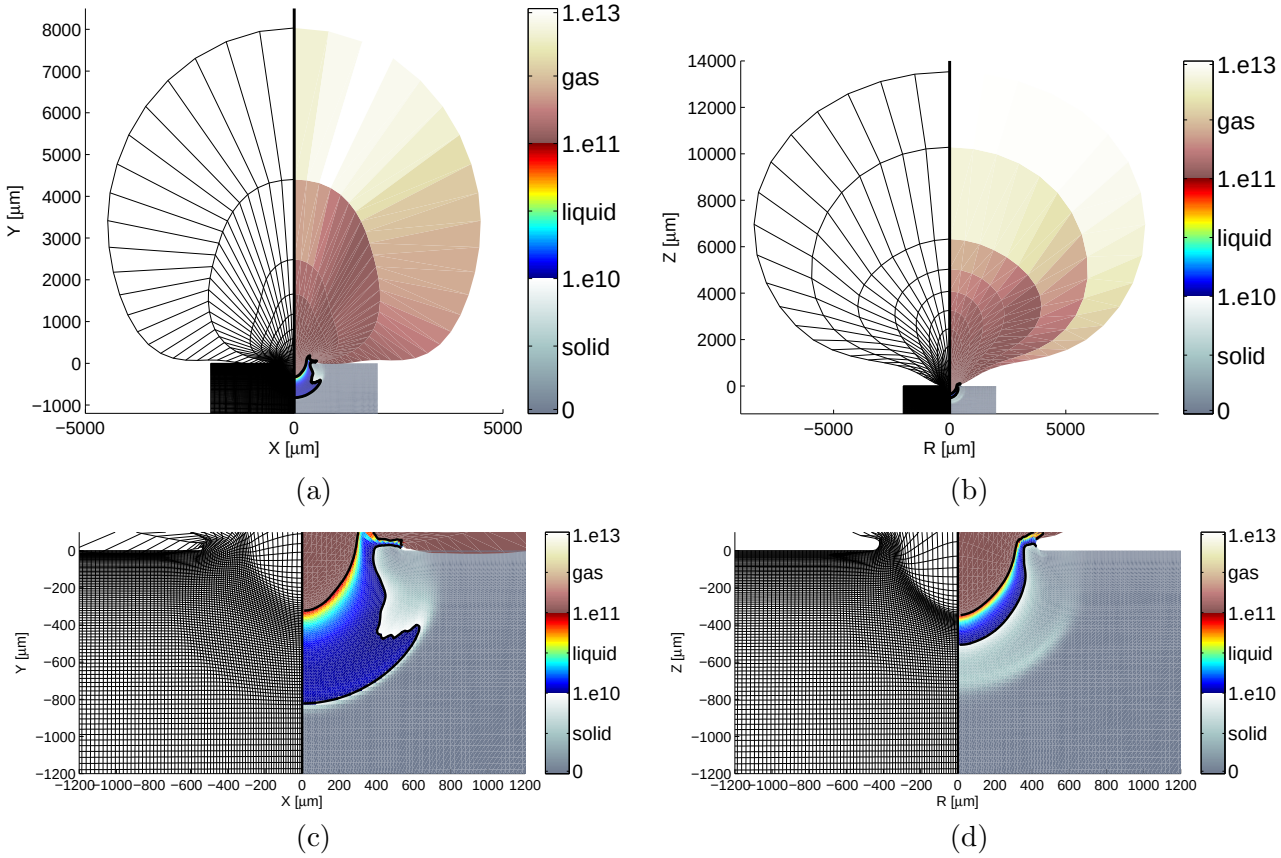


Figure 7.6: Computational mesh and specific internal energy increase colormap (in erg/g) for laser interaction with massive Aluminum target (problem 2, basic laser frequency) in time 70 ns after the laser pulse. Views to the whole computational domain (a), (b) and details of the crater region (c), (d) are presented, obtained by the Cartesian (a), (c) and cylindrical (b), (d) simulations by complete ALE method. Solid, liquid, and gas phases are separated by the specific energy isolines of melting $\epsilon^m = 9.72\,10^9$ erg/g and evaporation $\epsilon^v = 1.35\,10^{11}$ erg/g of Aluminum, different material phases are shown in different colormaps.

is probably caused by the fact, that the quotidian equation of state described in section 6.1 does not explicitly include the transition heats of phase changes. Therefore, it is better to draw the specific internal energy increase, and compute the melting and evaporating energies by including these transition heats. Such specific internal energy increase is shown in Figure 7.6 (a), (b). The specific energy increase needed for the melting of the material ϵ^m is equal to the specific energy increase needed for raising its temperature from the room temperature to the boiling point $\left(T^m_{Al} - T^r\right) c_{Al}$ plus the specific latent transition heat of melting of Aluminum $L^m_{Al} = 3.97 \cdot 10^5$ J/kg, where the specific heat of Aluminum is $c_{Al} = 900$ J/kg/K. The final value is $\epsilon^m = 9.72 \cdot 10^5$ J/kg $= 9.72 \cdot 10^9$ erg/g. By adding the specific energy increase needed for raising the temperature to the boiling (vaporization) temperature $\left(T^v_{Al} - T^m_{Al}\right) c_{Al}$ and specific latent transition heat of vaporization $L^v_{Al} = 1.09 \cdot 10^7$ J/kg, we get the value of the specific energy increase needed for evaporation of Aluminum $\epsilon^v = 1.35 \cdot 10^7$ J/kg $= 1.35 \cdot 10^{11}$ erg/g. Isolines in these specific energies ϵ^m and ϵ^v separate Aluminum phases, and again, different colormaps are used for different material phases. As in the Figure 7.5 displaying temperature, there is two orders higher energy in the low density corona than the energy needed for Aluminum melting. Zoom to the crater region is shown in Figure 7.6 (c), (d). The energy increase behind the cylindrical shock wave is zero, the material was not heated yet. The estimated crater depth is 320 μm in the Cartesian and 350 μm in

the cylindrical case, its radius is 290 μm in the Cartesian and 300 μm in the cylindrical case. As one can see, in this simulation, the crater size in energy increase is much closer to the experimental values, than the previous crater region size, deduced from temperature. As the experimental geometry is cylindrical, cylindrical simulations indeed agree with experiments much better than for planar geometry. In the rest of this section, we only use cylindrical values of crater depth and radius.
For completeness, in Table 7.3 we present the crater sizes in all 9 problems from [14]. For each problem,

Problem Nr.	experiment		temperature		energy	
	H_c	R_c	H_c	R_c	H_c	R_c
1	400	330	380	320	290	290
2	340	280	440	340	350	300
3	480	400	380	550	150	450
4	160	630	160	750	50	1100
5	600	800	600	850	200	700
6	500	480	470	400	360	360
7	600	580	510	370	420	330
8	700	600	450	600	220	450
9	280	700	240	800	50	1200

Table 7.3: Crater depth H_c and radius R_c (in μm) for each problem according to Table 7.2. Experimental values and values from cylindrical ALE simulations (temperature and specific internal energy increase) are presented.

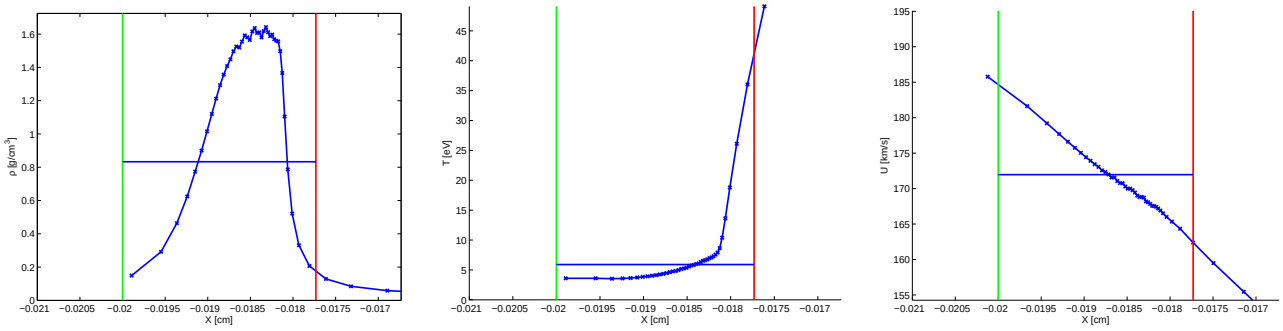
we present experimental crater depth H_c and radius R_c, and the computed crater sizes estimated from the distributions of temperature and specific internal energy increase in cylindrical coordinates. As one can see, the crater size values from numerical simulations are in most cases a little smaller than the experimental ones, which is probably caused by the fact, that also a piece of the liquid material is thrown off in the real experiment. Generally, the simulated values (especially from temperature, and for smaller laser intensities also from specific internal energy increase) correspond to the experimental values reasonably.

7.3 Ablative Flyer Acceleration by Laser Beam

The first simulated stage of the experiment is the interaction of the laser beam with disc flyer, see Figure 7.1. The laser pulse irradiates a small Aluminum disc flyer, the surface of the disc is evaporated and the rest of the disc is accelerated up to a very high velocity. The intense laser pulses have different energies $E^{\mathrm{L}} = 120\,\mathrm{J}$, 130 J, 240 J, or 390 J, the pulse duration is $t_{\mathrm{FWHM}} = 400\,\mathrm{ps}$. Laser wavelength is $\lambda_1 = 1.315\,\mu$m for the basic laser frequency and $\lambda_3 = 0.438\,\mu$m for the third harmonic frequency. The disc radius is $r_d = 150\,\mu$m, its thickness is either $d = 6\,\mu$m or $d = 11\,\mu$m, depending on the particular experiment. The simulation lasts till the moment, when the accelerated part of the disc reaches the massive target, or in other words, till it flies over the whole distance $L = 200\,\mu$m.
Two approaches are used for the simulations of the flyer irradiation, preliminary 1D Lagrangian simulations, and simulations by full ALE 2D method. Both approaches are demonstrated on the example of 11 μm thin Aluminum disc irradiated by 240 J laser pulse in third harmonic.

7.3.1 Purely Lagrangian Simulation in 1D

In [41, 53], several preliminary simulations have been done in 1D to estimate the flyer density, temperature, and velocity. For such 1D simulations, purely Lagrangian approach is sufficient. Maximum laser intensity $I_{\max}$ is estimated from the formula (7.7).



(a) (b) (c)

Figure 7.7: 1D Lagrangian simulation of 11 μm disc irradiation by 240 J laser beam on third harmonic frequency. Density (a), temperature (b), and velocity (c) profile in time 1.70 ns in the disc region is shown. Impacting disc between the red and green lines, blue line shows the average value.

For the 1D Lagrangian simulations, we used the code implemented by [41]. Here, we present the simulation of the experiment with 240 J laser beam operating on the third harmonic frequency $\lambda_1 = 0.438\,\mu$m with the radius of laser spot on target $r_f = 125\,\mu$m, irradiating $d = 11\,\mu$m thin Aluminum disc flyer. In Figure 7.7, we present the profiles of density, temperature, and velocity in time 1.70 ns, which is time needed by the disc to fly over the whole distance $L = 200\,\mu$m and reach the massive Aluminum target, shown by the green line marking its right boundary. The end of the flying disc is shown as the red line in the Figure 7.7, and is defined as the position, where the density profile decreases below 10 % of the maximum density. Each average value (average weighted by the cell mass) is shown as the blue line in the Figure 7.7. In the particular example, the impacting disc is 22.4 μm wide, and has the average density $\rho^{\mathrm{disc}} = 0.833\,\mathrm{g/cm^3}$, which corresponds to 66 % of the initial disc mass in the impacting disc region. The average impacting disc temperature is 5.9 eV, and average impacting velocity 172 km/s.
In the following Table 7.4, we present similar results for each experiment from [45]. Comparison of the

Problem	t^{imp} [ns]	d^{disc} [μm]	ρ^{disc} [g/cm^3]	$\frac{m^{\mathrm{disc}}}{m^{\mathrm{disc}}_{\mathrm{init}}}$	T^{imp} [eV]	v_{imp} [km/s]	v_{e} [km/s]
6 μm, 130 J, 1st	2.3	14	0.79	66 %	3.2	120	60
6 μm, 130 J, 3rd	1.5	10	0.88	54 %	4.5	201	150
11 μm, 120 J, 1st	3.9	32	0.66	71 %	2.1	62	40
11 μm, 120 J, 3rd	2.6	25	0.77	63 %	2.8	99	?
11 μm, 240 J, 1st	2.4	26	0.77	68 %	4.1	110	54
11 μm, 240 J, 3rd	1.7	23	0.83	63 %	5.9	172	?
11 μm, 390 J, 1st	1.8	14	1.18	57 %	6.6	164	?
11 μm, 390 J, 3rd	1.3	16	0.99	52 %	9.5	253	?

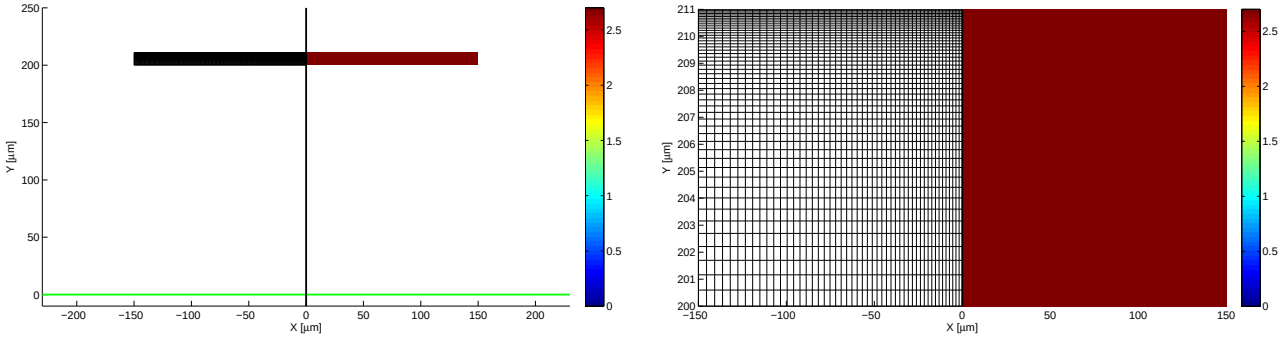
Table 7.4: Results of 1D Lagrangian simulation of flyer irradiation. For each problem, the following quantities are presented: time of impact t^{imp}, impacting disc width d^{disc}, impacting disc density ρ^{disc}, mass ratio in the impacting disc $m^{\mathrm{disc}}/m^{\mathrm{disc}}_{\mathrm{init}}$, average impacting disc temperature T^{imp}, average impacting disc velocity v_{imp}, and the experimental impacting velocity v_{e} measured in [45]. Symbol "?" means that the experimental impact velocity is not available.

simulated impact velocity v_{imp} and the experimental impact velocity v_{e} measured in [45] is included. The impact velocities from the simulations are much higher than the experimental ones. Laser energy is transfered only to longitudinal motion of the fluid in the direction of the laser beam, the fluid is not allowed to move in other direction. In reality, a part of the laser energy is transformed into the plasma motion in the direction perpendicular to the laser beam, which limits the resulting momentum of the

flyer and its impact speed. For 2D simulations of the presented problems, see the next section.
The values from Table 7.4 are used as the initial data for the second part of the simulation. Although, they come from rough 1D simulation, they provide the first estimate of the "homogeneous" initial values. The simulations of the disc impact problem using these initial data are presented in section 7.4.1.

7.3.2 2D Simulation by Complete ALE Method

In the previous section, we presented the 1D Lagrangian simulations of the irradiation of a thin Aluminum disc by an intense laser beam. In this section, we present the simulations of the same problems by our 2D ALE code. As in the section 7.2, the simulations are performed in mostly Lagrangian mode – no regular mesh smoothing/remapping step is performed, it is used only in the situation, that negative volume appears in some cell or subcell of the computational mesh. Heat conductivity is an important process in this part of the simulation, and has to be included as it seriously influences shock wave speed. Several 2D acceleration simulations have been presented in [65], [52].
As in the previous section, we present here figures of 11 μm disc irradiated by 240 J laser beam operated on the third harmonic frequency, with the radius of laser spot on target $r_f = 125\,\mu$m. In Figure 7.8, the



(a) (b)

Figure 7.8: Initial density (solid Aluminum density $\rho = 2.7\,\mathrm{g/cm^3}$) and initial mesh of the 11 μm wide disc with radius 150 μm. The disc (a) is placed in the height 200 μm above the massive Aluminum target (green line). Zoom (b) to the central region is presented.

initial density (constant solid mass density $2.7\,\mathrm{g/cm^3}$ in the whole disc) and the initial computational mesh projected to the left halfplane is shown. The initial mesh is quadrilateral, but not equidistant. The mesh cells are smaller in the upper part and close to the $r = 0$ axis, where the laser beam (incident from up) interacts with the matter. Each cell length geometrically depend on the cell index such as

$$\Delta x_i = \frac{r_d}{\Delta_x^{ni} - 1}\,\Delta_x^i\,(\Delta_x - 1)\,, \tag{7.16}$$

where the factor Δ_x is the parameter, set in our flyer simulations as $\Delta_x = 1.02$. Similarly the cell heights geometrically change in y direction, where the factor is more severe $\Delta_y = 0.95$.
After the laser beam irradiates the flyer, a shock wave is formed inside the disc, and hot, low density corona is created moving against the incident laser beam.
In Figure 7.9, the situation just before the impact is shown in both Cartesian and cylindrical coordinates. The figures (a) and (b) show the complete computational domain evolved from the initial disc in Figure 7.8. To be specific, the computational domain in cylindrical geometry increased more than 10^4times. The thin disc is rapidly heated and evaporated, and due to the momentum conservation, it starts the downwards motion. Most of the disc mass has the form of high density arch accelerated up to a very high velocity, moving in the direction of the laser beam. The arch shape is formed due to a higher laser beam intensity in the central region accelerating the disc more than the lower intensity

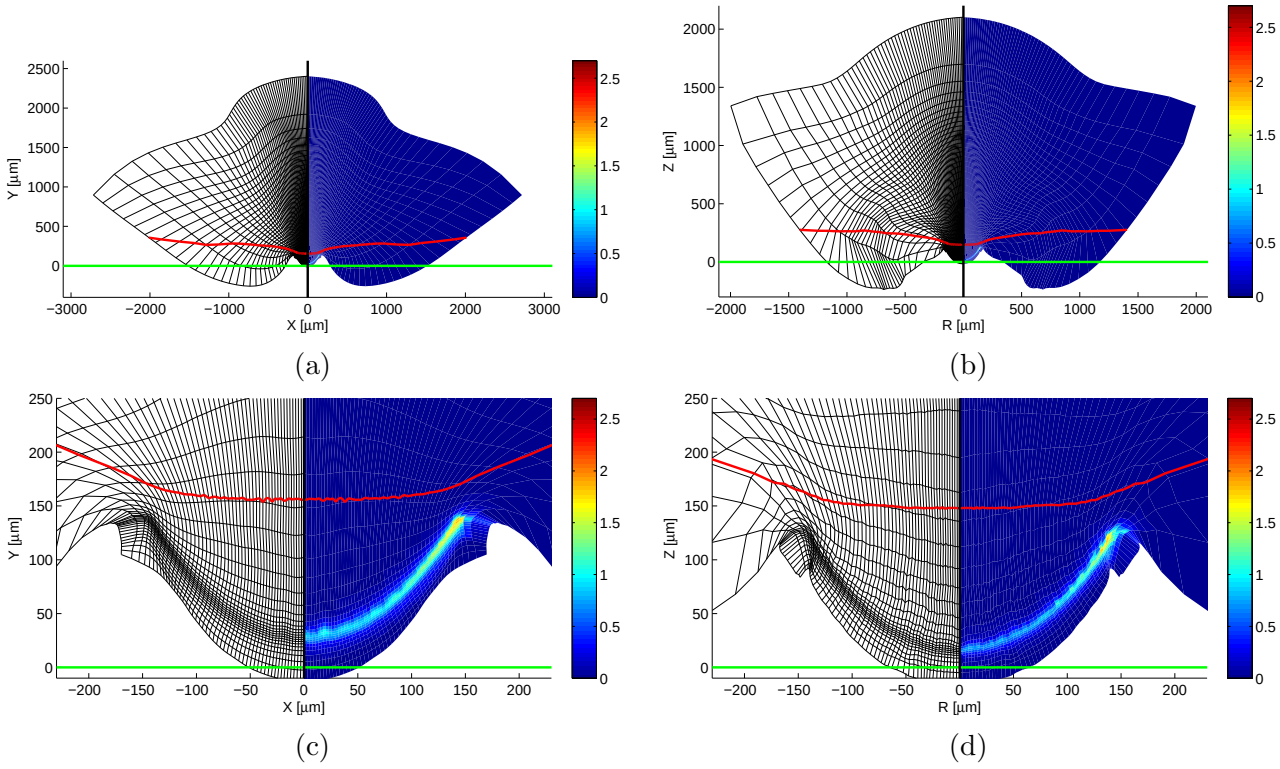


Figure 7.9: Density (in g/cm^3) and computational mesh of 2D ALE simulation of problem with 11 μm wide disc of radius 150 μm irradiated by 240 J laser beam operating on the third harmonic frequency in time of impact to the massive target in Cartesian (a), (c) and cylindrical (b), (d) coordinates. View to the whole computational domain (a), (b) and zoom to the critical region (c), (d) are shown. Green line represents the interface of the solid target, red line is the isoline of zero vertical velocity v.

close to the disc edge. Zoom to this high density region is shown in parts (c) and (d) of Figure 7.9. On the other hand, the disc has lower density in the central region than close to the disc edge, which can be interpreted as almost burning a hole in the disc. When we compare the Cartesian and cylindrical simulations, this effect is stronger in the cylindrical geometry. This is naturally caused by the higher laser beam central peak intensity in the cylindrical geometry.

When comparing the Cartesian and cylindrical simulations, the higher maximum of the laser beam intensity also causes faster acceleration of the disc and earlier disc impact (1.30 ns in cylindrical geometry instead of 1.60 ns in Cartesian coordinates). Faster disc movement and lower disc density in the central region, are general properties of the cylindrical flyer simulations, compared with the Cartesian case.

For comparison of 1D Lagrangian disc flyer simulation with the presented 2D ALE case, we present the average cylindrical values in the region of the high density. This region is specified by a rectangle including most of the disc mass. For this particular simulation, it is estimated as the (r, z) rectangle of size $\langle 0, 180\rangle\,\mu\text{m} \times \langle 0, 170\rangle\,\mu\text{m}$. The overview of the average impacting disc quantities is presented in Table 7.5. The disc rectangle includes 50 % – 80 % of the initial disc mass, depending on the time until the impact. Generally, for fast discs (and short flying times), the corona does not evolve so strongly, the disc remains more compact, and higher mass percentage is included in the disc region.

When compared with the presented 1D average values from Table 7.4, much higher average density in the 1D simulations can be observed. This is, of course, caused by much larger region affecting the impact in the 2D case, so the maximal density is more important parameter for the impact, than the average density. Generally, the maximal disc density is higher in 2D simulations (and the same with the disc temperature). What is most important for our simulations, the vertical impact velocities are lower than

Problem	t^{imp} [ns]	$\frac{V_{\mathrm{fin}}}{V_{\mathrm{init}}^{\mathrm{disc}}}$	$\frac{\rho^{\mathrm{disc}}}{\rho_{\mathrm{max}}^{\mathrm{disc}}}$ [g/cm^3]	$\frac{m^{\mathrm{disc}}}{m_{\mathrm{init}}^{\mathrm{disc}}}$	T^{imp} [eV]	v_{imp} [km/s]	v_{e} [km/s]
6 μm, 130 J, 1st	2.2	6.5e5	0.07/0.55	56 %	7.7	88	60
6 μm, 130 J, 3rd	1.1	9.8e3	0.07/3.19	71 %	43.9	153	150
11 μm, 120 J, 1st	3.4	1.8e6	0.11/0.80	68 %	3.4	46	40
11 μm, 120 J, 3rd	1.8	7.3e4	0.09/1.28	79 %	9.8	82	?
11 μm, 240 J, 1st	2.4	7.6e5	0.13/0.88	66 %	6.5	76	54
11 μm, 240 J, 3rd	1.3	1.9e4	0.13/1.89	73 %	23.9	134	?
11 μm, 390 J, 1st	2.0	4.7e5	0.13/0.90	65 %	10.5	104	?
11 μm, 390 J, 3rd	1.1	1.8e4	0.14/2.59	67 %	53.9	187	?

Table 7.5: Average values (mass-weighted) in the disc region in cylindrical coordinates. Following quantities are presented for each problem: time of impact t^{imp}, ratio of final computational domain to the initial computational domain (disc) $V_{\mathrm{fin}}/V_{\mathrm{init}}^{\mathrm{disc}}$, average and maximal densities ρ^{disc} and $\rho_{\mathrm{max}}^{\mathrm{disc}}$ of the impacting disc, ratio of the impacting disc mass to the total disc mass $m^{\mathrm{disc}}/m_{\mathrm{init}}^{\mathrm{disc}}$, average impacting disc temperature T^{imp}, and the average vertical velocity v_{imp} of the impacting disc, compared with the experimental impacting velocity v_{e} measured in [45]. Symbol "?" means that the experimental impact velocity is not available.

the 1D ones, and more realistically correspond to the measured experimental velocities also presented in the Table 7.5. The average velocities are computed as the mass-weighted average

$$v_{\mathrm{imp}} = \frac{\sum\limits_{n \in disc} m_n \, v_n}{\sum\limits_{n \in disc} m_n} \tag{7.17}$$

of the nodal vertical velocities in the disc region. In fact, it corresponds to the computation from the average disc momentum in direction z in cylindrical coordinates.
The more reasonable impact speed arises from the better 2D approximation, than in the 1D case. For the particular simulation of 11 μm disc irradiated by a 240 J laser beam in third harmonic, the kinetic energy of the impacting disc (now, by impacting disc, we understand everything moving in the direction of the laser beam) in 1D case is 21.9 J. When this value is compared with the 2D kinetic energy of the impacting disc 23.6 J, they are rather close to each other. In 1D, this total kinetic energy results from the disc movement towards the massive target in laser direction. In 2D, the disc velocity in the z axis direction (towards the target) gives only 16.5 J kinetic energy and the remaining 7.1 J results from the movement in r direction. Thus, lower kinetic energy of the disc in z direction corresponds to the lower impact speed in 2D. For completeness, let us note, that most of the kinetic energy of motion in z direction is situated in the region of massive disc (left from line $r = 180\,\mu$m), concretely 15.2 J. On the other hand, most of the impacting disc kinetic energy in r direction lies in the region of low density corona right from line $r = 180\,\mu$m, only 0.6 J is situated in the massive part of the target. The slower 2D impacting disc is caused by the lower kinetic energy in the direction of the laser beam, and better approximates real experiment behavior.

7.4 Disc Flyer Impact Simulations

The second stage of the simulations is the impact of the evaporated and accelerated part of the disc flyer on the massive Aluminum target. After the impact, a shock wave is formed moving from the area of the impact to the target interior. This shock wave causes heating, melting, and evaporation of the target material. The evaporated material streams out in a form of a low density corona, and a crater is formed. The size and shape of the crater formation is the main parameter of our investigation.

In [45], several experiments of small disc flyers to the massive target are described. We perform simulations for parameters of these experiments using our ALE approach, and compare the crates obtained in our simulations with the experimental ones.
At first, in section 7.4.1, we present the impact simulations using an easy approximation of the initial data from the 1D data described in section 7.3.1. In the next section 7.4.2, simulations starting from the complete 2D initial data resulting from the process described in section 7.3.2 are presented. Both Cartesian and cylindrical simulations of the same $11\,\mu$m disc irradiated by 240 J laser pulse in third harmonic are shown, and the differences between them are pointed out.

7.4.1 Impact Simulations Started from 1D Uniform Initial Data

Here, the initial data are taken from 1D simulations described in section 7.3.1. This simple approach was used for our preliminary simulations presented in [55], [54], [53], [56].
The density distribution on the initial mesh for the laser beam of third harmonic frequency of iodine laser of energy 240 J, and $d = 11\,\mu$m disc width is shown in Figure 7.10. In the disc region, the uniform density,

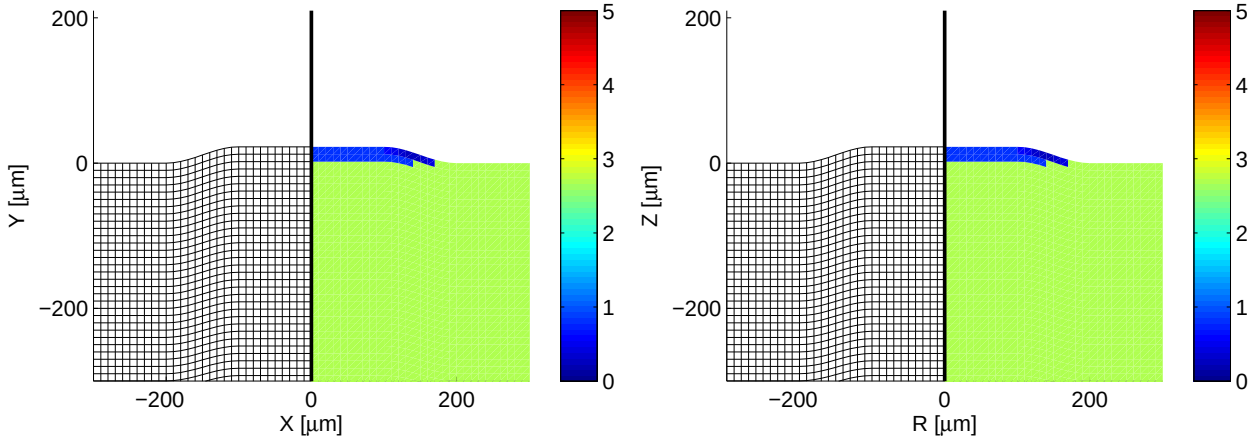


Figure 7.10: Initial data from the originally $d = 11\,\mu$m disc accelerated by $E^{\mathrm{L}} = 240\,$J laser beam on third harmonic frequency to the impact velocity $v_{\mathrm{imp}} = 172\,$km/s. In the impacting disc region, the constant density $\rho = 0.774\,\mathrm{g/cm^3}$, constant temperature $T = 5.9\,$eV, and the constant impact velocity $v_{\mathrm{imp}} = 172\,$km/s are obtained as the mass-weighted average values from the 1D simulation presented in Table 7.4.

velocity, and temperature distributions equal to the mass-weighted average values coming from the appropriate 1D simulation from Table 7.4 are used. The initial mesh is almost equidistant quadrilateral mesh in and out of the disc region, and smoothly connected in the region of the disc edge, as shown in the Figure 7.10.
After starting the simulation, the cells on the interface of the disc and the massive target are compressed to higher densities, and their temperature is increased. The disc sinks into the target, and in the region on the disc edge, the material is pushed out of the target, and creates a small rim. In Figure 7.11, comparison of the purely Lagrangian simulations and simulations by complete ALE method in time $t = 0.3\,$ns, in both Cartesian and cylindrical geometries, is shown. The purely Lagrangian simulation is not able to handle the computational mesh shear movement in the critical region around the disc edge, and the computational mesh starts to degenerate here. On the other hand, the simulation by complete ALE method has no troubles with distorted mesh, the mesh smoothing/quantity remapping process resolves all problems with severe mesh movement. Short time after the displayed snapshot, the purely Lagrangian simulations produce negative cell volumes and consequently negative cell densities, and cause the failure of the method as its assumptions are violated.

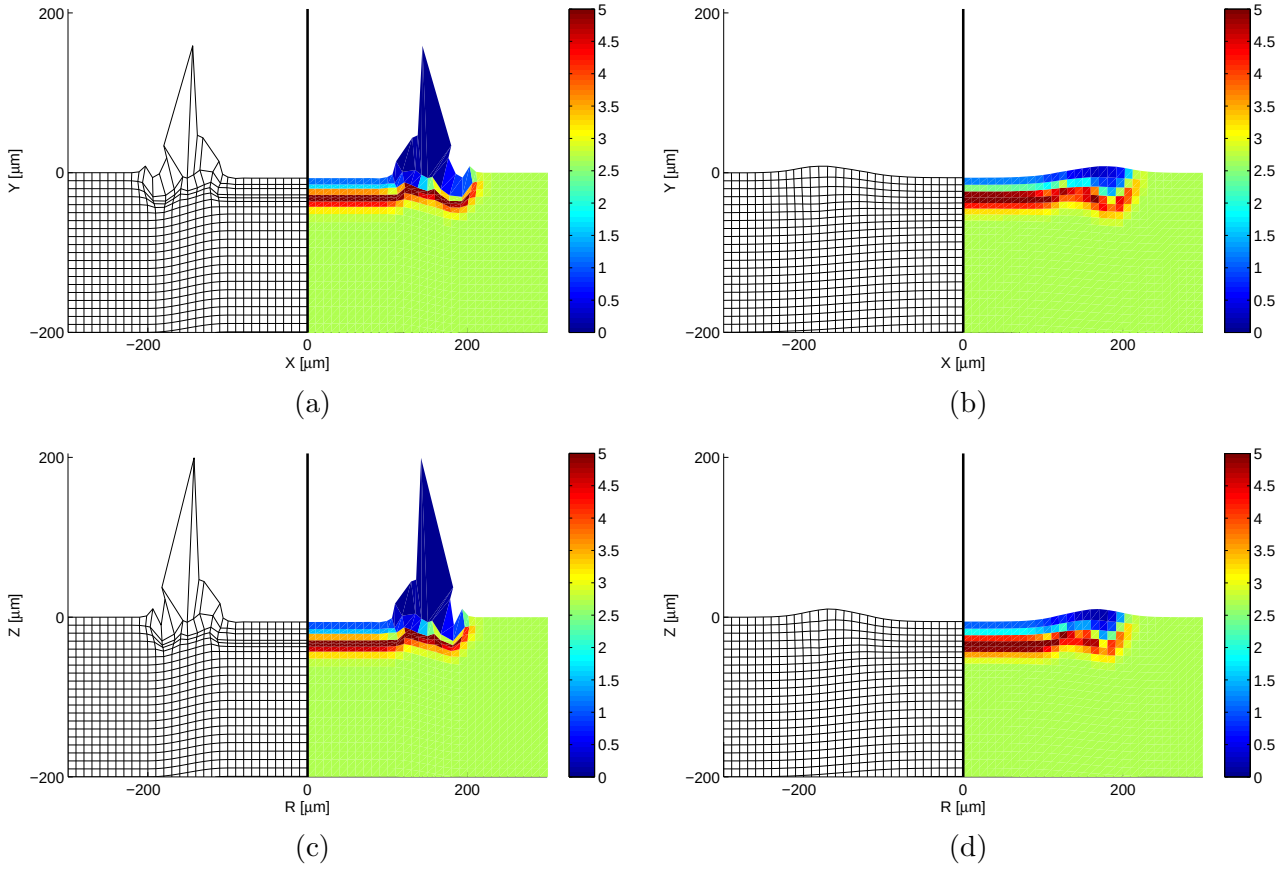


Figure 7.11: Comparison of purely Lagrangian simulations (a), (c) and simulations by complete ALE (b), (d) method (density colormap) of the uniform disc impact problem in both Cartesian (a), (b) and cylindrical (c), (d) geometries in time $t = 0.3\,\text{ns}$. Initial data are shown in Figure 7.10.

In Figure 7.12, the temperature colormaps in time 80 ns after the impact, in both coordinate systems are shown. In temperatures $T^m_{Al} = 0.085\,\text{eV}$ (melting of Aluminum) and $T^v_{Al} = 0.25\,\text{eV}$ (evaporation of Aluminum), isolines separating single phases of the material are contoured. The top pink colormap is used in the area of the evaporated material, the reflected gas is moving up in a form of huge corona. The central colored colormap shows the melted liquid part of the material, and the bottom gray shade colormap shows material remaining in solid state. The downwards propagating shock wave is positioned at the solid-liquid interface. The evolved crater shape can be seen. By the crater, we mean the interface between the gas and the liquid state, because the gas flows away, but the liquid Aluminum becomes solid again after temperature decrease. At later times crater does not propagate further down into the massive target, while the shock wave continues to move inside.
In Figure 7.13, similar picture is shown. This time, the increase of the specific internal energy is shown instead of the material temperature, which should correspond better to the real material behavior (compared with temperature), because the melting and evaporation isolines in $\epsilon^m = 9.72 \cdot 10^9\,\text{erg/g}$ and $\epsilon^v = 1.35 \cdot 10^{11}\,\text{erg/g}$ include the specific heats of melting and evaporation.
For the particular presented simulation, the experimental crater size is unknown. In Table 7.6, we present the comparison of the crater sizes obtained from the Cartesian and cylindrical simulations (in both temperature and specific energy increase), with the experimental values. The crates deduced from temperature are generally bigger, than the ones deduced from energy increase, which is caused by the absence of the specific heats in the EOS. As for the crater sizes deduced from energy, the cylindrical ones are closer to the experimental values than the Cartesian ones. In fact, the Cartesian simulation models

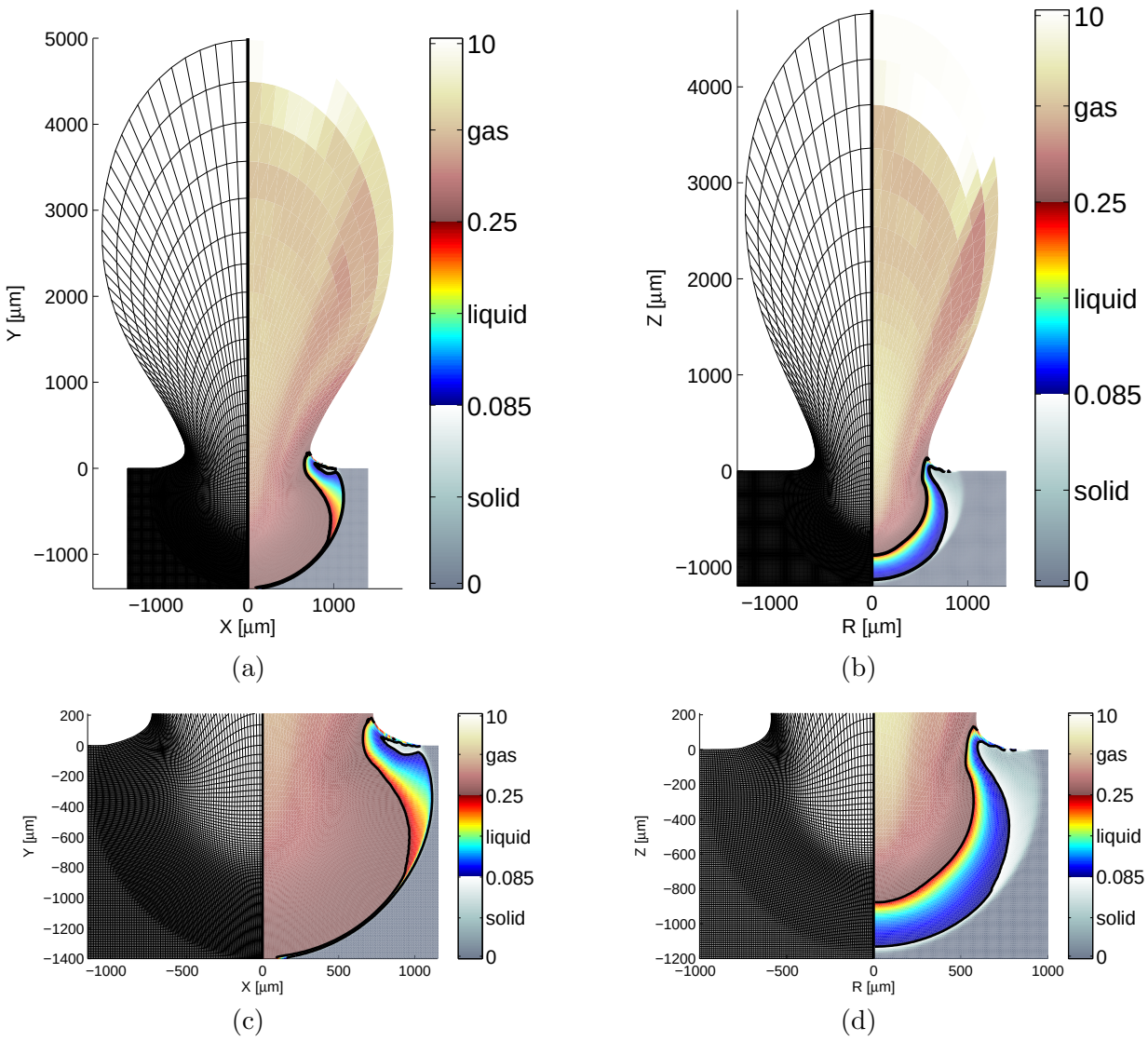


Figure 7.12: Computational mesh and temperature colormap (in eV) for uniform disc impact problem (11 μm wide disc of radius 150 μm irradiated by 240 J laser beam operating in the third harmonic) in time 80 ns after the impact. Views to the whole computational domain (a), (b) and details of the crater region (c), (d) are presented, obtained by the Cartesian (a), (c) and cylindrical (b), (d) simulations by complete ALE method. Solid, liquid, and gas phases are separated by the temperature isolines of melting $T^m_{Al} = 0.085\,\text{eV}$ and evaporation $T^v_{Al} = 0.25\,\text{eV}$ of Aluminum, different material phases are shown in different colormaps.

the creation of an infinitely long crater. The cylindrical simulation follows the cylindrical properties of the experiment, and the crater size is reasonably close to the experimental values.

7.4.2 Impact Simulations Started from 2D Interpolated Initial Data

In this section we show the results of the 11 μm disc accelerated by 240 J laser beam operating on the third harmonic frequency, impacting the massive Aluminum target. The initial data for this simulation were obtained from the 2D ALE simulation of ablative acceleration described in section 7.3.2. Several

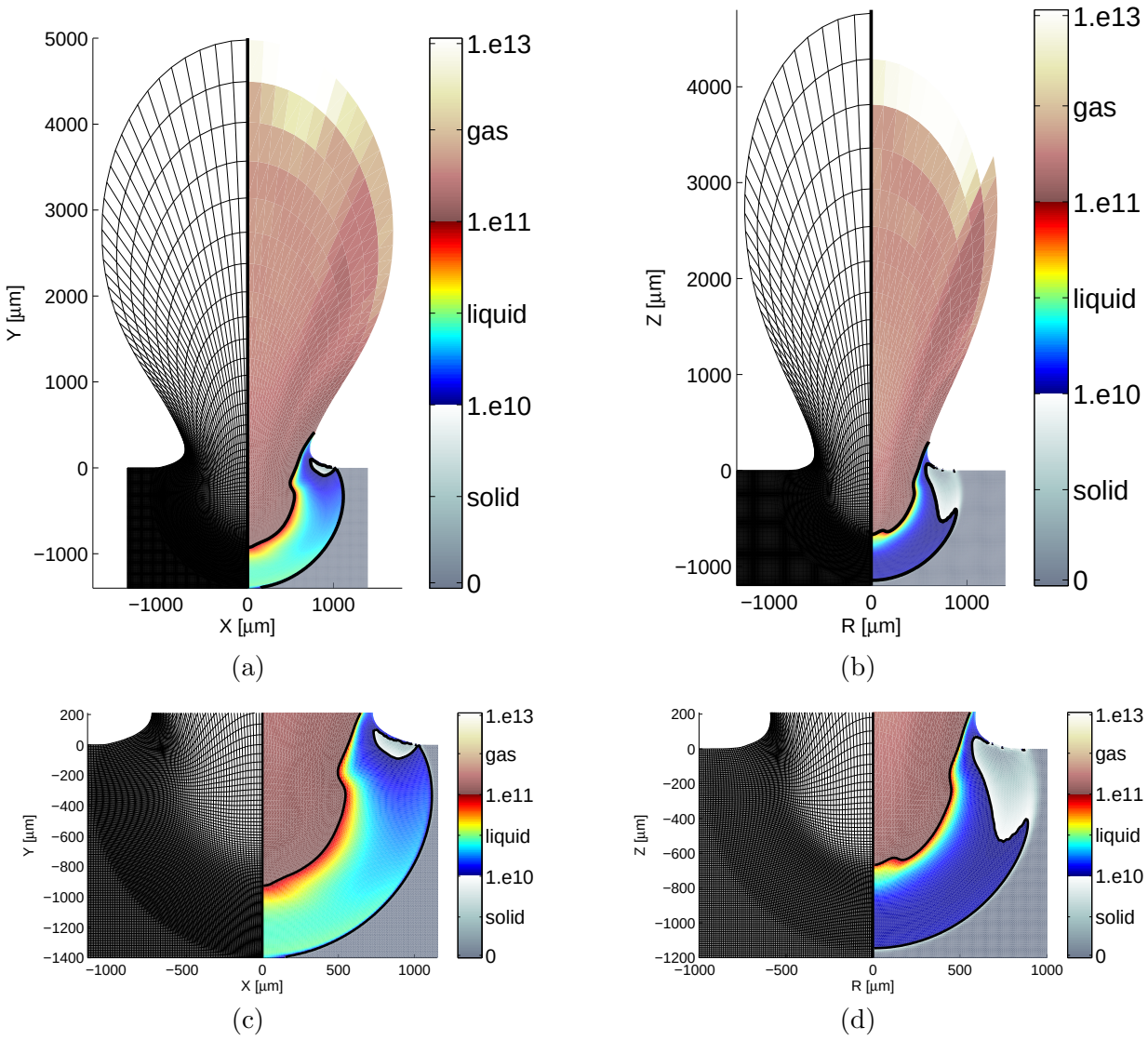
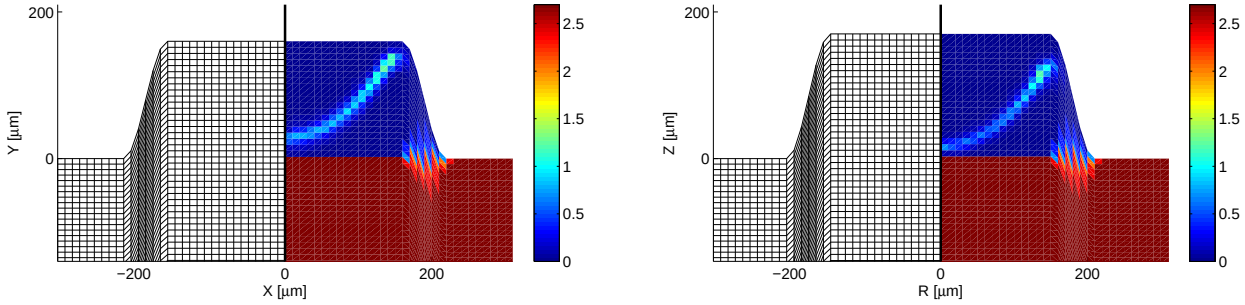


Figure 7.13: Computational mesh and specific internal energy increase colormap (in erg/g) for uniform disc impact problem (11 μm wide disc of radius 150 μm irradiated by 240 J laser beam operating in the third harmonic) in time 80 ns after the impact. Views to the whole computational domain (a), (b) and details of the crater region (c), (d) are presented, obtained by the Cartesian (a), (c) and cylindrical (b), (d) simulations by complete ALE method. Solid, liquid, and gas phases are separated by the specific energy isolines of melting $\epsilon^m = 9.72\,10^9$ erg/g and evaporation $\epsilon^v = 1.35\,10^{11}$ erg/g of Aluminum, different material phases are shown in different colormaps.

simulations of such impact problems are presented in [65], [52].
The initial mesh (shown in Figure 7.14) is similar to the mesh used for simple average impact simulations from section 7.3.2. The disc is much thicker now than in the simulations started from average initial data, to cover the part of the ablative accelerated flyer with the most of the disc mass. The mesh is almost equidistant in the part of the mesh out of the disc and below the disc, and is smoothly connected in the region of the disc edge. The initial mesh in both Cartesian and cylindrical geometry is shown in Figure 7.14. The initial size of the impacting disc is estimated by hand to include the whole disc arch (described in section 7.3.2) and a small neighborhood. For the particular example of 11 μm disc irradiated

Problem	experiment		temperature		energy	
	H_c	R_c	H_c	R_c	H_c	R_c
6 μm, 130 J, 1st, Cart.	?	?	850	470	480	410
6 μm, 130 J, 1st, cyl.	300	300	580	380	370	350
6 μm, 130 J, 3rd, Cart.	?	?	1000	850	800	510
6 μm, 130 J, 3rd, cyl.	550	500	800	530	590	420
11 μm, 120 J, 1st, Cart.	?	?	560	370	220	340
11 μm, 120 J, 1st, cyl.	280	300	480	350	240	320
11 μm, 120 J, 3rd, Cart.	?	?	750	420	370	380
11 μm, 120 J, 3rd, cyl.	?	?	570	380	360	350
11 μm, 240 J, 1st, Cart.	?	?	850	580	550	400
11 μm, 240 J, 1st, cyl.	320	320	660	400	480	370
11 μm, 240 J, 3rd, Cart.	?	?	1400	950	920	560
11 μm, 240 J, 3rd, cyl.	?	?	880	550	660	450
11 μm, 390 J, 1st, Cart.	?	?	> 1400	700	650	470
11 μm, 390 J, 1st, cyl.	380	400	740	460	520	400
11 μm, 390 J, 3rd, Cart.	?	?	> 1400	1000	1100	650
11 μm, 390 J, 3rd, cyl.	?	?	> 1400	900	1000	550

Table 7.6: Comparison of crater depths H_c and diameters R_c (in μm) according to temperature and specific internal energy increase in Cartesian and cylindrical simulations determined from the uniform 1D initial disc data. Symbol "?" denotes unknown experimental crater size.



(a) (b)

Figure 7.14: Initial density profile of the impact simulation of 11 μm wide disc of radius 150 μm irradiated by 240 J laser beam operating in the third harmonic. Both Cartesian (a) and cylindrical (b) initial data interpolated from disc acceleration simulation in Figure 7.9, are presented.

by 240 J laser pulse on the third harmonic frequency, the initial disc (r, z) size is $\langle 0, 180\rangle$ μm $\times$ $\langle 0, 170\rangle$ μm, with the right edge beveled in a 30 μm belt of the mesh.

Initial density profile is shown in Figure 7.14. In the disc region, the initial density, temperature, and velocities are conservatively interpolated from the disc flyer simulation, and in the massive target region, it is set to the solid Aluminum density $\rho = 2.7\,\mathrm{g/cm^3}$. Solid target is in the rest, and has room temperature.

The impact simulation is similar to the previous simulation presented section 7.4.1. The circular shock wave in time 80 ns after the impact is visible in material density profile shown in Figure 7.15. The shock wave is moving inside the massive target, and causes heating of the material, its melting and evaporation. Material temperature colormap in the same time is shown in Figure 7.16, and the increase of the specific internal energy in Figure 7.17. In both figures, the situations in the whole computational domain, and

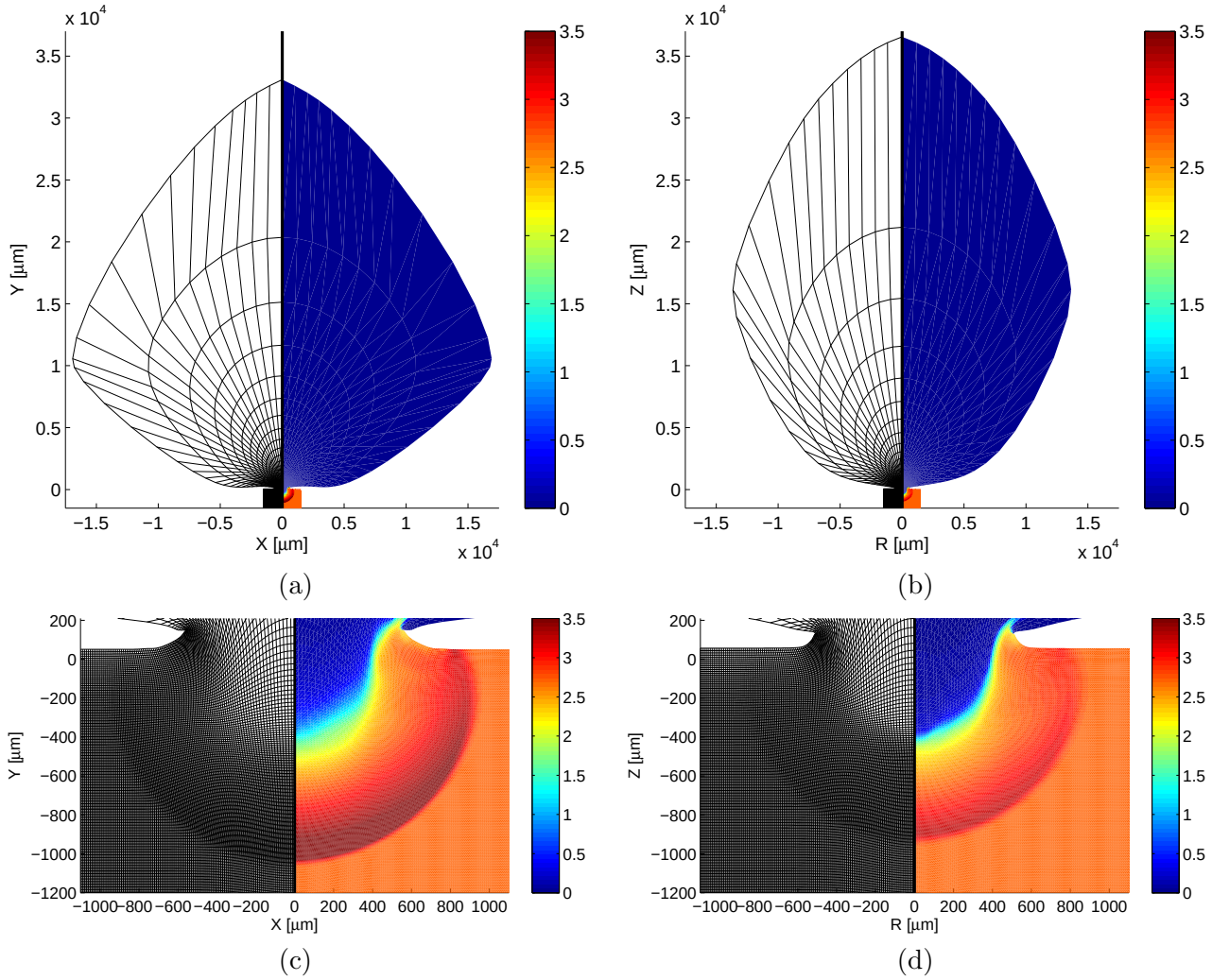


Figure 7.15: Computational mesh and density colormap (in g/cm^3) for the disc impact problem interpolated from the disc acceleration simulation by complete ALE method (11 μm wide disc of radius 150 μm irradiated by 240 J laser beam operating in the third harmonic) in time 70 ns after the impact. Views to the whole computational domain (a), (b) and details of the crater region (c), (d) are presented, obtained by the Cartesian (a), (c) and cylindrical (b), (d) simulations by complete ALE method.

zooms to the crater region, are presented. The Cartesian figures are shown for completeness only. As in the previous sections, the temperature and internal energy increase isolines of melting and evaporation of Aluminum are contoured, separating different phases of the material. For better insight, each material phase is shaded in different colormap – pink gas, colored liquid, and gray solid. Again, by the crater, we mean the interface between the gas and liquid materials – the evaporated gas escapes away in the form of hot corona, and the liquid material becomes solid again, due to the cooling down the material after some time.

In the following Table 7.7, the crater sizes of all the examples, both in temperature and specific internal energy increase, are summarized. Both Cartesian and cylindrical data are included, although the Cartesian simulations do not correspond to the experimental geometry. For comparison, experimental data are included.

The values of the crater depths and radials determined from the simulations starting from interpolated 2D impact data are closer to the experimental values than the values obtained by the simulations started

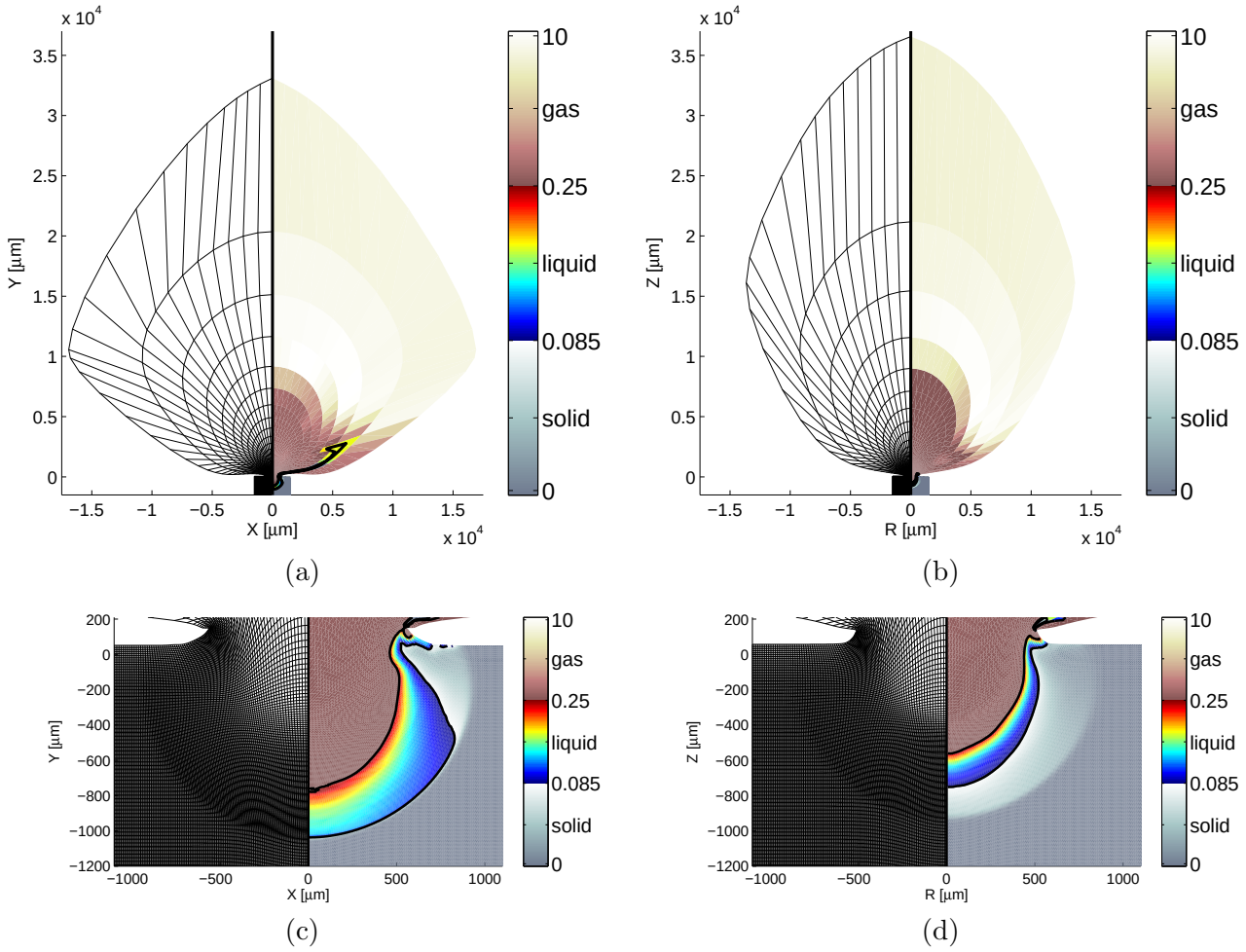


Figure 7.16: Computational mesh and temperature colormap (in eV) for the disc impact problem interpolated from the disc acceleration simulation by complete ALE method (11 μm wide disc of radius 150 μm irradiated by 240 J laser beam operating in the third harmonic) in time 80 ns after the impact. Views to the whole computational domain (a), (b) and details of the crater region (c), (d) are presented, obtained by the Cartesian (a), (c) and cylindrical (b), (d) simulations by complete ALE method. Solid, liquid, and gas phases are separated by the temperature isolines of melting $T^m_{Al} = 0.085\,\mathrm{eV}$ and evaporation $T^v_{Al} = 0.25\,\mathrm{eV}$ of Aluminum, different material phases are shown in different colormaps.

from 1D average initial data. We also have to note here that for low intensity laser beam (120 J and 130 J), the craters deduced from temperature are better approximation of the experimental craters than the ones deduced from energy increase. In the case of high energy laser beams (240 J and 390 J), the situation is opposite, craters deduced from energy increase represent better approximation of the experimental behavior.

The errors of crater volumes (according to the volumes of the experimental craters) of cylindrical simulations, estimated from temperature and energy increase, are compared in Table 7.8. The craters volumes have been computed as the volumes of ellipsoid with the axes H, R, and R. All crater volumes have been relatively compared with the experimental crater volumes, computed the same way. For low intensity lasers, the errors of volumes of craters deduced from temperature are between 9 % and 25 %. In the case of high density lasers, the errors of volumes of craters deduced from the specific energy increase are between 12 % and 15 %. Generally, the 2D cylindrical simulations provide better results than the Cartesian ones, and reasonably approximate the experimental craters.

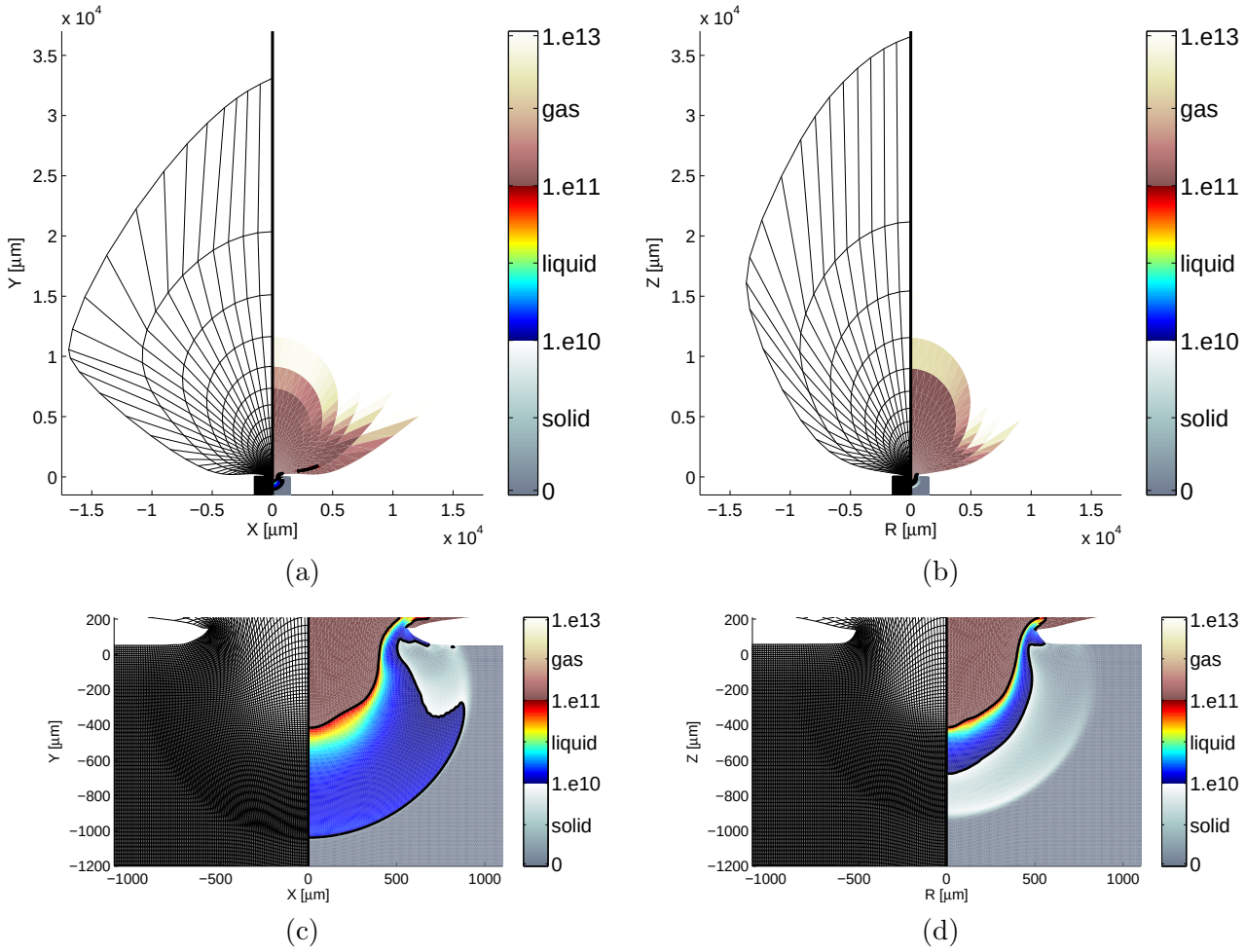


Figure 7.17: Computational mesh and specific internal energy increase colormap (in erg/g) for the disc impact problem interpolated from the disc acceleration simulation by complete ALE method (11 μm wide disc of radius 150 μm irradiated by 240 J laser beam operating in the third harmonic) in time 80 ns after the impact. Views to the whole computational domain (a), (b) and details of the crater region (c), (d) are presented, obtained by the Cartesian (a), (c) and cylindrical (b), (d) simulations by complete ALE method. Solid, liquid, and gas phases are separated by the specific energy isolines of melting $\epsilon^m = 9.72\,10^9$ erg/g and evaporation $\epsilon^v = 1.35\,10^{11}$ erg/g of Aluminum, different material phases are shown in different colormaps.

7.5 Energy Balance in the Simulations

In this section, we discuss the energy balance of the whole simulation. This is an important issue proving the relevance of the computed data – the total disc energy must correspond to the absorbed laser beam energy. The energy of the impacting disc should correspond to the energy of the part of the accelerated disc in the region including most of the disc mass. This energy should remain unchanged during the whole impact simulation. We focus here on the advanced simulations – 2D ALE disc flyer acceleration (described in section 7.3.2) and the 2D ALE impact simulation from the 2D ALE initial data (described in section 7.4.2), and omit the preliminary 1D Lagrangian acceleration simulations (described in section 7.3.1) and the impact simulations coming from constant uniform initial data (described in section 7.4.1). We also focus here on the cylindrical simulations only, because they are physically relevant when compared with the Cartesian ones, which in fact assume an infinitely long disc/crater. As

Problem	experiment		temperature		energy	
	H_c	R_c	H_c	R_c	H_c	R_c
6 μm, 130 J, 1st, Cart.	?	?	320	290	140	260
6 μm, 130 J, 1st, cyl.	300	300	350	290	210	260
6 μm, 130 J, 3rd, Cart.	?	?	560	410	330	380
6 μm, 130 J, 3rd, cyl.	550	500	470	470	320	420
11 μm, 120 J, 1st, Cart.	?	?	260	270	120	240
11 μm, 120 J, 1st, cyl.	280	300	270	280	200	260
11 μm, 120 J, 3rd, Cart.	?	?	490	390	290	350
11 μm, 120 J, 3rd, cyl.	?	?	410	370	330	350
11 μm, 240 J, 1st, Cart.	?	?	500	340	240	310
11 μm, 240 J, 1st, cyl.	320	320	460	330	300	310
11 μm, 240 J, 3rd, Cart.	?	?	760	530	410	410
11 μm, 240 J, 3rd, cyl.	?	?	560	450	400	410
11 μm, 390 J, 1st, Cart.	?	?	670	420	370	370
11 μm, 390 J, 1st, cyl.	380	400	530	370	450	340
11 μm, 390 J, 3rd, Cart.	?	?	> 1400	> 1400	> 1400	1100
11 μm, 390 J, 3rd, cyl.	?	?	680	540	560	450

Table 7.7: Comparison of crater depth H_c and radius R_c (in μm) according to temperature and specific internal energy increase in Cartesian and cylindrical simulations determined from interpolated disc initial data. Symbol "?" denotes unknown experimental crater size.

Problem	1D average		2D interpolation	
	temp	ener	temp	ener
6 μm, 130 J, 1st, cyl.	210 %	68 %	9 %	48 %
6 μm, 130 J, 3rd, cyl.	63 %	24 %	25 %	59 %
11 μm, 120 J, 1st, cyl.	85 %	3 %	16 %	46 %
11 μm, 240 J, 1st, cyl.	222 %	101 %	53 %	12 %
11 μm, 390 J, 1st, cyl.	158 %	37 %	19 %	14 %

Table 7.8: Comparison of crater volume deviations of simulated crater sizes (relative to experimental crater volume) obtained by cylindrical simulations determined from 1D average and 2D interpolated initial data, estimated from temperature and energy increase.

shown in the previous sections, the cylindrical simulations provide results closer to the physical results, than the Cartesian ones.
The total energy in the whole computational domain is defined as

$$E_T = E_I + E_K = \sum_{\forall c} m_c \left(\epsilon_c - \epsilon_0\right) + \sum_{\forall c} \sum_{l=1}^{4} \frac{1}{2} m_c^l \left(u_{n(c,l)}^2 + v_{n(c,l)}^2\right) , \tag{7.18}$$

where ϵ_0 is the specific internal energy in the initial cold massive target. In fact, the first part does not represent the total internal energy, but the internal energy increase. The ϵ quantity is a potential – it can be shifted by arbitrary constant value (in the case of QEOS, it is shifted to the level about -10^{14}). At the beginning of the disc flyer acceleration, the uniform disc has constant initial temperature (and thus constant specific internal energy) and zero velocity, and so the total initial energy of the simulation is 0. After the simulation starts, the laser pulse starts irradiating the flyer disc, heats it, and forces it to move, and thus both energy components are increasing, and the laser pulse energy is transfered to the

Problem	E_A [J]	E_T [J]	E_I [J]	E_K [J]	$E_K \downarrow$ [J]
6 μm, 130 J, 1st cyl.	58.6	61 (104 %)	4.5 (7.6 %)	56 (96 %)	5.5 (9.4 %)
6 μm, 130 J, 3rd cyl.	87.9	90 (103 %)	14.8 (16.8 %)	75 (86 %)	15.1 (17.2 %)
11 μm, 120 J, 1st cyl.	54.1	56 (104 %)	2.9 (5.3 %)	54 (99 %)	2.7 (5.0 %)
11 μm, 120 J, 3rd cyl.	81.1	85 (104 %)	8.0 (9.9 %)	77 (94 %)	8.2 (10.2 %)
11 μm, 240 J, 1st cyl.	108.2	114 (105 %)	6.6 (6.1 %)	107 (99 %)	11.8 (10.9 %)
11 μm, 240 J, 3rd cyl.	162.3	170 (105 %)	18.5 (11.4 %)	151 (93 %)	23.6 (14.5 %)
11 μm, 390 J, 1st cyl.	175.8	181 (103 %)	12.2 (6.9 %)	169 (96 %)	21.4 (12.2 %)
11 μm, 390 J, 3rd cyl.	263.7	277 (105 %)	34.6 (13.1 %)	242 (92 %)	46.2 (17.5 %)

Table 7.9: Energy balance of each problem in time of the disc impact in the whole computational domain. Following energies are summarized: E_A – absorbed laser beam energy, E_T – total domain energy, E_I – internal energy in computational domain, E_K – kinetic energy in computational domain, and $E_K \downarrow$ – kinetic in the part of the domain, where the material moves downwards (towards the massive target). Percentages of the energies according to the absorbed laser energy E_A are presented.

kinetic and internal energy of the disc material. At the time of the impact (when the accelerated disc reaches the massive target), the energy of the laser pulse in the disc region (about $C_d = 0.9$ for 125 μm radius of laser spot on target and 150 μm disc radius) is already absorbed into the kinetic and internal energy of the material. To be precise, only its C_a ratio is absorbed ($C_a = 0.5$ for first and $C_a = 0.75$ for third harmonic frequency) due to reflection of part of the laser beam.

In Table 7.9, the comparison of the absorbed laser beam energy E_A with the total energy of the whole domain $E_T = E_I + E_K$ is summarized. In all tables, we present the energies also as the percentage of the exact absorbed energy E_A. The total domain energy E_T reasonably approximates the exact absorbed energy $E_A = C_a\, C_d\, E^{\mathrm{L}}$, the maximal difference is about 5 %. Thus, the laser absorption process is modeled reasonably. Moreover, most of the disc energy has the form of the kinetic energy, and only about 10 % of the energy is the internal energy. The hydroefficiency (portion of downwards-moving kinetic energy) is $5 - 12$ % for first and $10 - 18$ % for third harmonic frequency. This is caused by the fact, that the huge, fast upwards moving corona includes most of the kinetic energy, because the kinetic energy depends on the second power of the fluid velocity. Thus, for our impact simulations, this energy deposited in the corona kinetic energy is lost for impact, we are only interested in the kinetic energy of the part of the disc moving downwards $E_K \downarrow$, which is the kinetic energy in the region bellow the red line in Figure 7.9.

However, we are interested in the energy of the region including most of the disc mass. As described in the previous sections, this region is estimated by hand to cover the complete arch of higher density, plus some neighborhood. In the next Table 7.10, we summarize the total, internal, and the kinetic energy in the disc region in the original flyer acceleration computational mesh. The energy of this region important for the impact is only a small part of the total domain energy. This is caused by huge corona, which is moving fast and very hot, so both the internal and kinetic energies of corona are bigger, than the energies of the more dense disc region. For third harmonic, the flying disc is faster, reaches the massive target earlier, and includes more of the total domain energy, than for the first harmonic. This effect is visible in both energy components – in the high density disc region, there is about $3 - 5$ % of kinetic and $6 - 7$ % of internal energies for the first harmonic. For the third harmonic, there is about $7 - 15$ % of kinetic and $15 - 25$ % of internal energies, when compared with the total domain kinetic and internal energies. This is caused by the faster movement of the disc in the third harmonic, less time is needed for the acceleration, and the high energy (fast and hot) corona does not have enough time to evolve as much, as in the first harmonic case.

We have to note here, that the previous Table 7.10 includes energies computed from the values in the original flyer acceleration disc. In the next Table 7.11, energies of the initial data of the impact problem,

Problem	E_A [J]	E_T [J]	E_I [J]	E_K [J]
6 μm, 130 J, 1st cyl.	58.6	2.9 (4.9 %)	0.3 (0.5 %)	2.6 (4.4 %)
6 μm, 130 J, 3st cyl.	87.9	15.6 (17.7 %)	3.6 (4.1 %)	12.0 (13.6 %)
11 μm, 120 J, 1st cyl.	54.1	1.8 (3.3 %)	0.2 (0.3 %)	1.6 (2.9 %)
11 μm, 120 J, 3st cyl.	81.1	7.9 (9.7 %)	1.1 (1.4 %)	6.8 (8.3 %)
11 μm, 240 J, 1st cyl.	108.2	4.7 (4.4 %)	0.4 (0.4 %)	4.3 (4.0 %)
11 μm, 240 J, 3st cyl.	162.3	18.9 (11.7 %)	3.1 (1.9 %)	15.8 (9.7 %)
11 μm, 390 J, 1st cyl.	175.8	9.0 (5.1 %)	0.9 (0.5 %)	8.2 (4.6 %)
11 μm, 390 J, 3st cyl.	263.7	34.6 (13.1 %)	6.8 (2.6 %)	27.8 (10.5 %)

Table 7.10: Energy balance of each problem in time of the disc impact in the approximate domain (selected part with most of the mass) of the impacting disc. Following energies are summarized: E_A – absorbed laser beam energy, E_T – total disc energy, E_I – internal disc energy, and E_K – kinetic disc energy. Percentages of the energies according to the absorbed laser energy E_A are presented.

Problem	E_A [J]	E_T [J]	E_I [J]	E_K [J]
6 μm, 130 J, 1st cyl.	58.6	3.2 (5.4 %)	0.3 (0.5 %)	2.9 (4.9 %)
6 μm, 130 J, 3st cyl.	87.9	22.5 (25.6 %)	9.8 (11.1 %)	12.8 (14.5 %)
11 μm, 120 J, 1st cyl.	54.1	1.9 (3.5 %)	0.2 (0.4 %)	1.7 (3.1 %)
11 μm, 120 J, 3st cyl.	81.1	9.4 (11.5 %)	2.3 (2.8 %)	7.1 (8.7 %)
11 μm, 240 J, 1st cyl.	108.2	5.0 (4.6 %)	0.5 (0.5 %)	4.4 (4.1 %)
11 μm, 240 J, 3st cyl.	162.3	23.1 (14.3 %)	6.3 (3.9 %)	16.9 (10.4 %)
11 μm, 390 J, 1st cyl.	175.8	8.9 (5.1 %)	1.1 (0.6 %)	7.8 (4.4 %)
11 μm, 390 J, 3st cyl.	263.7	47.0 (17.8 %)	17.3 (6.6 %)	29.7 (11.3 %)

Table 7.11: Initial impact energy balance of each problem in the impacting disc. Following energies are summarized: E_A – absorbed laser beam energy, E_T – total disc energy, E_I – internal disc energy, and E_K – kinetic disc energy. Percentages of the energies according to the absorbed laser energy E_A are presented.

obtained by the interpolation of the flyer acceleration data to the initial impact mesh, are summarized. When the energies in both Tables 7.10, 7.11 are compared, a reasonably good agreement in the kinetic energy is achieved, the deviation is lower than 0.5 % when compared with the total absorbed energy. For the internal energy, the deviation is lower than 2 % with only one exception – the 6 μm thick disc, and 130 J laser beam in the third harmonic, where the internal energy error is almost 10 %. This is caused by different impacting disc regions used – in the first Table 7.10, we used rectangles, and in the second Table 7.11, the right edge of the impacting disc was beveled to smooth the computational domain. The deviation is caused by the interpolation process in the beveled part of the impacting disc, which is in fact, in the disc flyer acceleration simulation stage out of the computational domain (see Figure 7.9). As the high temperature region along the boundary is close to the beveled part, the data are interpolated to the beveled part of the disc with higher temperature, than appropriate. In the other simulations, the temperature close to the boundary is not so high, so this source of deviation is not so strong. The kinetic energy is more important for the crater evolution than the internal one, so the deviation is still acceptable.

In the last Table 7.12, we summarize the energy balance in the final moment of the crater evolution simulations. The total domain energy reasonably corresponds to the total initial energy introduced in Table 7.11. We also present the relative total energy deviation ΔE_T according to the total initial impact energy from Table 7.11. The numerical error of the energy conservation is less than 10 %, which is caused by mesh smoothing/quantity remapping process on the domain boundary. Generally, the volumes of the

Problem	E_A [J]	E_T [J]	E_I [J]	E_K [J]	$E_K \downarrow$ [J]	ΔE_T
6 μm, 130 J, 1st cyl.	58.6	2.9 (4.9 %)	1.3 (2.2 %)	1.6 (2.7 %)	0.1 (0.2 %)	9.6 %
6 μm, 130 J, 3st cyl.	87.9	24.8 (28.2 %)	6.2 (7.0 %)	18.6 (21.2 %)	0.6 (0.7 %)	10.0 %
11 μm, 120 J, 1st cyl.	54.1	1.8 (3.3 %)	0.9 (1.7 %)	0.8 (1.7 %)	0.1 (0.2 %)	5.1 %
11 μm, 120 J, 3st cyl.	81.1	9.0 (11.1 %)	3.3 (4.1 %)	5.7 (7.1 %)	0.4 (0.5 %)	3.6 %
11 μm, 240 J, 1st cyl.	108.2	4.8 (4.4 %)	2.1 (2.0 %)	2.7 (2.5 %)	0.3 (0.3 %)	3.2 %
11 μm, 240 J, 3st cyl.	162.3	22.9 (14.1 %)	7.5 (4.6 %)	15.4 (9.5 %)	1.0 (0.6 %)	1.0 %
11 μm, 390 J, 1st cyl.	175.8	8.6 (4.9 %)	3.4 (2.0 %)	5.2 (2.9 %)	0.5 (0.3 %)	3.2 %
11 μm, 390 J, 3st cyl.	263.7	49.6 (18.8 %)	14.2 (5.4 %)	35.4 (13.4 %)	2.1 (0.8 %)	5.5 %

Table 7.12: Final impact energy balance of each problem in the whole computational domain. Following energies are summarized: E_A – absorbed laser beam energy, E_T – total domain energy, E_I – internal domain energy, E_K – kinetic domain energy, $E_K \downarrow$ – kinetic energy in the part of the domain, where the material moves downwards (inside the massive target), and ΔE_T – relative deviation of total energy according to the total initial impact energies from Table 7.11. Percentages of the energies according to the absorbed laser energy E_A are presented.

Lagrangian and smoothed meshes are not exactly the same, so the energy conservation can be a little bit violated. By composing the effects of all the remappings, the mentioned error is created.
In this section, we have discussed the energy balance of the whole disc flyer acceleration/disc high-speed impact simulations. The behavior corresponds reasonably to the expected process, and shows the ability of the code to perform relevant laser plasma simulations.

Chapter 8

Conclusion

In this thesis, we have described the complete arbitrary Lagrangian-Eulerian (ALE) method for fluid dynamics and laser plasma simulations on 2D logically orthogonal computational meshes, in Cartesian and cylindrical geometries. We have demonstrated its properties for selected problems of laser-plasma interactions.
Our ALE method uses the staggered Lagrangian step [18], several mesh smoothing techniques, and the conservative remapping process [72]. We have focused in details on the remapping algorithm and introduced several its improvements [58]. The complete ALE algorithm was generalized into the cylindrical geometry, allowing to perform simulations of naturally cylindrical laser-plasma processes.
The ALE method has been implemented, and several techniques for treating the laser plasma behavior – QEOS equation of state [75], thermal conductivity, simple laser absorption model, or dynamically changing boundary conditions – have been added to it. These techniques allow the presented method to be used for advanced laser-plasma simulations. We have demonstrated its properties on a set of laser-flyer-target simulations based on real experiments, and compared the simulations with the experimental results. Their correspondence and the energy balance analysis show the ability of the code to provide realistic simulations for problems, where neither the Eulerian nor Lagrangian approaches are suitable.
The main contribution and new ideas of the thesis lie in the following topics

- Conservative interpolation methods in Cartesian and cylindrical geometry.
- Complete remapping (for all state quantities) in cylindrical geometry.
- Implementation of 2D ALE method into code for logically-orthogonal meshes in Cartesian and cylindrical geometry.
- Simulations of disc flyer acceleration by an intense laser beam, and its impact to the massive target.

The conservative interpolation method is based on piecewise-linear reconstruction, approximate swept region integration, and repair stage enforcing local-bound preservation. We have also introduced the generalization of the remapping algorithm to general 3D meshes, and 2D meshes with changing connectivity. To recompute all state quantities in both geometries, we have generalized the algorithm from [70] to the cylindrical geometry. The presented complete ALE method (in both geometries) was implemented into a computer code in Fortran. Its memory and computational time demands are reasonable on current machines. Finally, we performed several simulations of laser-target and laser-flyer-target experiments, and present the obtained results. The simulations follow the real experiments performed on PALS laser system, the simulated and experimental craters are comparable.
This thesis does not entirely expound the topic, there are many points, which could be modified or improved to produce more realistic results. Although, our code provides reasonable and realistic simulations, close to the experimental behavior.

The methods described in this thesis give powerful tools to those who need to perform laser plasma simulations, for which the classically used Eulerian or Lagrangian methods have troubles. It can be used for such problems, as high velocity impact simulations, and it has been shown to produce realistic results. The field of laser plasma physics is a dynamically evolving topic influencing many aspects of the human being, and the demands of implementing new methods into our code can be expected in the future.

List of Selected Important Publications

For completeness, we add the list of selected important journal and conference proceedings articles of the thesis author. Most of them are directly connected to the thesis topic, and include parts of the presented results. Only few of them include previous work of the author, they have just a marginal relation to the topic. Besides the presented list of publications, the author published several technical reports, short reports in university and group collections, and abstracts in the conference books of abstracts. The author also presented the topic on many local meetings and international conferences, and keeps in touch with the scientific community.

Journal articles:

1. R. Garimella, M. Kuchařík, and M. Shashkov: An Efficient Linearity and Bound Preserving Conservative Interpolation (Remapping) on Polyhedral Meshes, *Computers and Fluids*, 2006. In press.

2. M. Kuchařík, R. Liska, S. Steinberg, and B. Wendroff: Optimally-Stable Second-Order Accurate Difference Schemes for Nonlinear Conservation Laws in 3D, *Applied Numerical Mathematics*, Vol. 56, Nr. 5, pp. 589–607, 2006.

3. M. Kuchařík, R. Liska, J. Limpouch, and P. Váchal: ALE Simulations of High-Velocity Impact Problem, *Czechoslovak Journal of Physics*, Vol. 54, Suppl. C, pp. 391–396, 2004.

4. M. Kuchařík, M. Shashkov, and B. Wendroff: An efficient linearity-and-bound-preserving remapping method, *Journal of Computational Physics*, Vol. 188, Nr. 2, pp. 462–471, 2003.

5. L. Drška, M. Kuchařík, J. Limpouch, R. Liska, M. Šiňor, and A. Iskakov: Hydrodynamical modeling of targets compression to high densities, *Czechoslovak Journal of Physics*, Vol. 52, Suppl. D, pp. 362–367, 2002.

Articles in Conference Proceedings:

1. R. Liska, and M. Kuchařík: Arbitrary Lagrangian Eulerian method for compressible plasma simulations, *Proceedings of EQUADIFF 11, International conference on differential equations*, Comenius University, 2005. 10 pages. Submitted.

2. M. Kuchařík, J. Limpouch, and R. Liska: Laser Plasma Simulations by Arbitrary Lagrangian Eulerian Method, *Proceedings of IFSA 2005, Fourth International Conference on Inertial Fusion Sciences and Applications*, 2005. 3 pages. Submitted.

3. M. Kuchařík: Conservative Interpolations in ALE Codes, *Proceedings of Workshop of Applied Mathematics 2005*, Czech Technical University in Prague, 2005. 9 pages. Accepted.

4. M. Kuchařík, R. Liska, and M. Shashkov: Conservative Remapping and ALE Methods for Plasma Physics, *Proceedings of HYP 2004, Hyperbolic Problems: Theory, Numerics and Applications*, volume 2, editors F. Asakura, S. Kawashima, A Matsumura, S. Nishibata, and K. Nishihara. Yokohama Publishers, 2006. 8 pages. ISBN 4-946552-22-7.

5. M. Kuchařík, J. Limpouch, R. Liska, and P. Havlík: ALE Simulations of Laser Interactions with Flyer Targets, *Proceedings of XXVIII ECLIM, 28th European Conference on Laser Interaction with Matter*, pp. 470–474, 2004.

6. M. Kuchařík, and R. Liska: Arbitrary Lagrangian-Eulerian (ALE) Code for Plasma Simulations, *Proceedings of Czech-Japanese Seminar in Applied Mathematics*, Czech Technical University in Prague, editors M. Beneš, J. Mikiška, and T. Oberhuber, pp. 96–105, 2005. ISBN 80-01-03181-0.

7. R. Garimella, M. Kuchařík, and M. Shashkov: Efficient Algorithm for Local-Bound-Preserving Remapping in ALE Methods, *Proceedings of ENUMATH 2003, Numerical Mathematics and Advanced Applications*, editors M. Feistauer, V. Dolejší, P. Knobloch, and K. Najzar. Springer-Verlag Berlin Heidelberg New York, pp. 358–367, 2004. ISBN: 3-540-21460-7.

8. M. Kuchařík, M. Shashkov, and B. Wendroff: Efficient Local Bound-Preserving Conservative Interpolation, *Proceedings of Seventh U.S. National Congress on Computational Mechanics*, Omnipress, USA, pp. 166, 2003. ISBN: 0-9743254-0-6.

Acknowledgment

Different aspects of this research were partially supported by the Czech Technical University grants No. CTU0310614, CTU0410914, CTU0411014, CTU0415314, the Czech Ministry of Education research grants and projects No. FRVS 2087G1/2004, FRVS 1987G1/2005, LN00A100, LC528, MSM 6840770022, MSM 6840770010, and the Czech Grant Agency grant project No. 202/03/H162.

At the first place, I would like to thank my supervisor, Richard Liska, for patient leading and many relevant comments and remarks concerning all viewpoints of the presented topic. Special thanks also belong to Mikhail Shashkov for many ideas in the field of conservative interpolations, and also for allowing me to work on this topic for several times in the Los Alamos National Laboratory. Many other people affected different aspects of the work also: Jiří Limpouch gave me many good comments about the thermal conductivity processes, and provided us many experimental data of the laser-disc-target interactions; Raphael Loubere helped me very much with the remapping of all conservative variables, and with many helpful hints and advices about the cylindrical Lagrangian solver; Burton Wendroff with the remapping algorithm and the repair process development; Rao Garimella with the 3D remapping and unstructured mesh representation tools; Pavel Váchal with many clarifications in the field of mesh rezoning methods, and the implementation of several of them; Petr Havlík with the 1D simulations of the initial stages of the disc impact problem.

At the last but not least place, I would like to thank my wife and my daughter for their support and understanding.

Bibliography

[1] A. A. Amsden and C. W. Hirt. YAQUI. Technical Report LA-5100, Los Alamos Scientific Laboratory, 1973.

[2] A. A. Amsden, H. M. Ruppel, and C. W. Hirt. SALE: A simplified ALE computer program for fluid flow at all speeds. Technical Report LA-8095, Los Alamos National Laboratory, 1980.

[3] R. W. Anderson, N. S. Elliott, and R. B. Pember. An arbitrary Lagrangian-Eulerian method with adaptive mesh refinement for the solution of the Euler equations. *Journal of Computational Physics*, 199(2):598–617, 2004.

[4] A. N. Aristova, A. B. Iskakov, I. G. Lebo, and V. F. Tishkin. 2D-Lagrangian code LATRANT for simulation radiation gas dynamic problems. In O. N. Krokhin, S. Y. Guskov, and Y. A. Merkulev, editors, *ECLIM 2002: 27th European Conference on Laser Interaction with Matter*, volume 5228 of *Proceedings of the SPIE*, pages 131–142, 2003.

[5] ASC FLASH Center, University of Chicago. *FLASH User's Guide, Version 2.5*, 2005.

[6] A. Barlow. ALE methods for solving shock hydrodynamics problems. *Discovery – The Science & Technology Journal of AWE*, 4:10–17, 2002.

[7] T. J. Barth. Numerical methods for gasdynamic systems on unstructured meshes. In C. Rohde D. Kroner, M. Ohlberger, editor, *An introduction to Recent Developments in Theory and Numerics for Conservation Laws, Proceedings of the International School on Theory and Numerics for Conservation Laws*, Berlin, 1997. Lecture Notes in Computational Science and Engineering, Springer. ISBN 3-540-65081-4.

[8] T. J. Barth and D. C. Jaspersen. The design and application of upwind schemes on unstructured meshes. In *AIAA-89-0366*, 1989. 27th Aerospace Sciences Meeting, January 9-12, Reno, Nevada.

[9] D. J. Benson. An efficient, accurate, simple ALE method for nonlinear finite element programs. *Computer Methods in Applied Mechanics and Engineering*, 72(3):305–350, 1989.

[10] D. J. Benson. A new two-dimensional flux-limited shock viscosity for impact calculations. *Computer Methods in Applied Mechanics and Engineering*, 93(1):39–95, 1991.

[11] D. J. Benson. Computational methods in Lagrangian and Eulerian hydrocodes. *Computer Methods in Applied Mechanics and Engineering*, 99(2-3):235–394, 1992.

[12] J. C. Boettger. SESAME equation of state for epoxy. Technical Report LA-12755-MS, Los Alamos National Laboratory, 1994.

[13] S. Borodziuk, A. Kasperczuk, T. Pisarczyk, M. Kalal, J. Ullschmied, J. Limpouch, K. Rohlena, N. N. Demchenko, S. Yu. Gus'kov, V. B. Rozanov, V. N. Kondrashov, and P. Pisarczyk. Application of the three frame interferometric and crater replica methods for investigation of a laser accelerated macroparticles – massive target interaction in the PALS experiment. *Optica Applicata*, 34(3):385–403, 2004.

[14] S. Borodziuk, A. Kasperczuk, T. Pisarczyk, K. Rohlena, J. Ullschmied, M. Kalal, J. Limpouch, and P. Pisarczyk. Application of laser simulation method for the analysis of crater formation experiment on PALS laser. *Czechoslovak Journal of Physics*, 53(9):799–810, 2003.

[15] K. Budge. ALE shock calculations using a stabilized serendipity rezoning scheme. In S. C. Schmidt, R. D. Dick, J. W. Forbes, and D. G. Tasker, editors, *Shock compression of condensed matter*, Proceedings of the APS Topical Conference on Shock Compression of Condensed Matter. Elsevier, 1991.

[16] J. C. Campbell and M. J. Shashkov. A compatible Lagrangian hydrodynamics algorithm for unstructured grids. Technical Report LA-UR-00-3231, Los Alamos National Laboratory, 2000.

[17] J. C. Campbell and M. J. Shashkov. A tensor artificial viscosity using a mimetic finite difference algorithm. *Journal of Computational Physics*, 172(2):739–765, 2001.

[18] E. J. Caramana, D. E. Burton, M. J. Shashkov, and P. P. Whalen. The construction of compatible hydrodynamics algorithms utilizing conservation of total energy. *Journal of Computational Physics*, 146(1):227–262, 1998.

[19] E. J. Caramana and M. J. Shashkov. Elimination of artificial grid distortion and hourglass-type motions by means of Lagrangian subzonal masses and pressures. *Journal of Computational Physics*, 142(2):521–561, 1998.

[20] E. J. Caramana, M. J. Shashkov, and P. P. Whalen. Formulations of artificial viscosity for muti-dimensional shock wave computations. *Journal of Computational Physics*, 144(2):70–97, 1998.

[21] K.-H. Chen, D. Fricker, J. Lee, and J. Moder. *ALLSPD-3D User Guide, Version 2.0a*. NASA Lewis Research Center, 1998.

[22] J. P. Christiansen, D. E. T. F. Ashby, and K. V. Roberts. MEDUSA a one-dimensional laser fusion code. *Computer Physics Communications*, 7(5):271–287, 1974.

[23] R. Courant and K. O. Friedrichs. *Supersonic Flow and Shock Waves*. Springer Verlag, 1999. Reprint of 1st ed. Interscience Publishers, New York 1948. ISBN 0-387-90232-5.

[24] R. M. Darlington, T. L. McAbee, and G. Rodrigue. A study of ALE simulations of Rayleigh-Taylor instability. *Computer Physics Communications*, 135(1):58–73, 2001.

[25] R. M. Darlington, T. L. McAbee, and G. Rodrigue. Large eddy simulation and ALE mesh motion in Rayleigh-Taylor instability simulation. *Computer Physics Communications*, 144(3):261–276, 2002.

[26] A. Djaoui and S.J. Rose. Calculation of the time-dependent excitation and ionization in a laser-produced plasma. *Journal of Physics B: Atomic, Molecular and Optical Physics*, 25(11):2745–2762, 1992.

[27] J. Donea, S. Guiliani, and J.P. Halleux. An arbitrary Lagrangian-Eulerian finite element method for transient fluid-structure interactions. *Computer Methods in Applied Mechanics and Engineering*, 33(1-3):689–723, 1982.

[28] J. Donea, A. Huerta, J.-Ph. Ponthot, and A. Rodríguez-Ferran. Arbitrary Lagrangian-Eulerian methods. In E. Stein, R. de Borst, and T. Hughes, editors, *The Encyclopedia of Computational Mechanics*, chapter 14, pages 413–437. Wiley, 2004.

[29] M. R. Dorr, F. X. Garaizar, and J. A. F. Hittinger. Adaptive laser plasma simulation (ALPS). Technical Report UCRL-TB-137290, Lawrence Livermore National Laboratory, 2001.

[30] L. Drška, A. Iskakov, M. Kuchařík, J. Limpouch, R. Liska, and M. Šiňor. Hydrodynamics modeling of targets for high density plasma generation. Technical report, Czech Technical University, 2001.

[31] L. Drška, M. Kuchařík, J. Limpouch, R. Liska, M. Šiňor, and A. Iskakov. Hydrodynamical modeling of targets compression to high densities. *Czechoslovak Journal of Physics*, 52(Suppl. D):362–367, 2002.

[32] V. Dyadechko, R.Garimella, and M. Shashkov. Reference Jacobian rezoning strategy for arbitrary Lagrangian-Eulerian methods on polyhedral grids. In *Proceedings of 13th International Meshing Roundtable*, pages 459–470, 2004.

[33] A. P. Favorskii. Variational-discrete models of hydrodynamics equations. *Differential Equations*, 16:1308–1321, 1980.

[34] J. H. Ferziger and M. Peric. *Computational Methods for Fluid Dynamics*. Springer-Verlag Berlin Heidelberg New York, 2002. ISBN 3-540-65373-2.

[35] R. Garimella, M. Kuchařík, and M. Shashkov. Efficient algorithm for local-bound-preserving remapping in ALE methods. In M. Feistauer, V. Dolejsi, P. Knobloch, and K. Najzar, editors, *Numerical Mathematics and Advanced Applications*, pages 358–367. Springer-Verlag Berlin Heidelberg New York, 2004. ISBN: 3-540-21460-7.

[36] R. Garimella, M. Kuchařík, and M. Shashkov. An efficient linearity and bound preserving conservative interpolation (remapping) on polyhedral meshes. *Computers and Fluids*, 2006. In press.

[37] G. Gisler, R. Weaver, M. Gittings, and C. Mader. Two- and three-dimensional asteroid ocean impact simulations. *International Journal of Impact Engineering*, 29(1-10):283–291, 2003.

[38] J. Grandy. Conservative remapping and region overlays by intersecting arbitrary polyhedra. *Journal of Computational Physics*, 148(2):433–466, 1999.

[39] S. Yu. Guskov, S. Borodziuk, M. Kalal, A. Kasperczuk, V. N. Kondrashov, J. Limpouch, P. Pisarczyk, T. Pisarczyk, K. Rohlena, J. Skala, and J. Ullschmied. Investigation of shock wave loading and crater creation by means of single and double targets in the PALS-laser experiment. *Journal of Russian Laser Research*, 26(3):228–244, 2005.

[40] J. A. Harte, W. E. Alley, D. S. Bailey, J. L. Eddleman, and G. B. Zimmerman. LASNEX – A 2D physics code for modeling ICF. *ICF Quarterly Report*, 6(4):150–164, 1997. Lawrence Livermore National Laboratory, UCRL-LR-105821-96-4.

[41] P. Havlík. Diferenční schémata pro hydrodynamiku na nerovnoměrných a Lagrangeovských sítích (Difference schemes for hydrodynamics on irregular and Lagrangian meshes). Czech Technical University, 2004. Research project.

[42] C. W. Hirt, A. A. Amsden, and J. L. Cook. An arbitrary Lagrangian-Eulerian computing method for all flow speeds. *Journal of Computational Physics*, 14(3):227–253, 1974.

[43] A. B. Iskakov, V. F. Tishkin, I. G. Lebo, J. Limpouch, K. Masek, and K. Rohlena. Two-dimensional model of thermal smoothing of laser imprint in a double-pulse plasma. *Physical Review E*, 61(1):842–847, 2000.

[44] K. Jungwirth, A. Cejnarova, L. Juha, B. Kralikova, J. Krasa, P. Krupickova E. Krousky, L. Laska, K. Masek, T. Mocek, M. Pfeifer, A. Prag, O. Renner, K. Rohlena, B. Rus, J. Skala, P. Straka, and J. Ullschmied. The prague asterix laser system. *Physics of Plasmas*, 8(5):2495–2501, 2001.

[45] M. Kalal, S. Borodziuk, N. N. Demchenko, S. Yu. Guskov, K. Jungwirth, A. Kasperczuk, V. N. Kondrashov, B. Kralikova, E. Krousky, J. Limpouch, K. Masek, P. Pisarczyk, T. Pisarczyk, M. Pfeifer, K. Rohlena, V. B. Rozanov, J. Skala, and J. Ullschmied. High power laser interaction with single and double layer targets. In *Proceedings of XXVIII ECLIM*, pages 249–260, 2004.

[46] D. Keller, T. J. B. Collins, J. A. Delettrez, P. W. McKenty, P. B. Radha, B. Whitney, and G. A. Moses. DRACO – A new multidimensional hydrocode. *Bulletin of the American Physical Society*, 44(7):37, 1999.

[47] P. Kjellgren and J. Hyvarinen. An arbitrary Lagrangian-Eulerian finite element method. *Computational Mechanics*, 21(1):81–90, 1998.

[48] P. M. Knupp, L. G. Margolin, and M. J. Shashkov. Reference Jacobian optimization-based rezone strategies for arbitrary Lagrangian Eulerian methods. *Journal of Computational Physics*, 176(1):93–128, 2002.

[49] M. Kuchařík. Diferenční schemata pro zákony zachování ve 3D (Difference Schemes for Conservation Laws in 3D). Master's thesis, Czech Technical University, 2001/2002.

[50] M. Kuchařík. 2D conservative interpolation for ALE methods. Technical Report LA-UR-02-6395, Los Alamos National Laboratory, Los Alamos, USA, 2002.

[51] M. Kuchařík. Conservative interpolations in ALE codes. In *Proceedings of Seminář z aplikované matematiky u příležitosti 100. výročí narození profesora Františka Vyčichlo*. Czech Technical University in Prague, 2005. Accepted.

[52] M. Kuchařík, J. Limpouch, and R. Liska. Laser plasma simulations by arbitrary Lagrangian Eulerian method. In *Proceedings of IFSA 2005*. CEA, Biarritz, France, 2005. Fourth International Conference on Inertial Fusion Sciences and Applications (IFSA 2005), Biarritz, France, September 4-9. Submitted.

[53] M. Kuchařík, J. Limpouch, R. Liska, and P. Havlík. ALE simulations of laser interactions with flyer targets. In *Proceedings of 28th ECLIM*, pages 470–474, 2004. September 6-10, Rome, Italy.

[54] M. Kuchařík and R. Liska. Arbitrary Lagrangian-Eulerian (ALE) code for plasma simulations. In M. Benes, J. Mikyska, and T. Oberhuber, editors, *Proceedings of Czech-Japanese Seminar in Applied Mathematics*, pages 96–105. Czech Technical University in Prague, 2005. ISBN 80-01-03181-0.

[55] M. Kuchařík, R. Liska, J. Limpouch, and P. Váchal. ALE simulations of high-velocity impact problem. *Czechoslovak Journal of Physics*, 54(Suppl. C):391–396, 2004.

[56] M. Kuchařík, R. Liska, and M. Shashkov. Conservative remapping and ALE methods for plasma physics. In F. Asakura, S. Kawashima, A Matsumura, S. Nishibata, and K. Nishihara, editors, *Hyperbolic Problems: Theory, Numerics and Applications*, volume 2. Osaka University, Yokohama Publishers, 2004. ISBN 4-946552-22-7.

[57] M. Kuchařík and M. Shashkov. An efficient linearity-and-bound-preserving conservative interpolation (remapping) method for meshes with changing connectivity. 2005. In preparation.

[58] M. Kuchařík, M. Shashkov, and B. Wendroff. An efficient linearity-and-bound-preserving remapping method. *Journal of Computational Physics*, 188(2):462–471, 2003.

[59] V. F. Kuropatenko. *Difference Methods for Solutions of Problems of Mathematical Physics*, volume 1, page 116. American Mathematical Society, Providence, 1967.

[60] R. Landshoff. A numerical method for treating fluid flow in the presence of shocks. Technical Report LA-1930, Los Alamos National Laboratory, 1955.

[61] C. B. Laney. *Computational Gasdynamics*. Cambridge University Press, Cambridge, UK, 1998. ISBN 0-521-57069-7.

[62] R. J. LeVeque. *Finite Volume Methods for Hyperbolic Problems*. Cambridge University Press, 2002. ISBN 0-521-81087-6.

[63] R. J. LeVeque. *CLAWPACK Version 4.2 User's Guide*. University of Washington, 2003.

[64] J. Limpouch. Personal communication, 2005.

[65] R. Liska and M. Kuchařík. Arbitrary Lagrangian Eulerian method for compressible plasma simulations. In *Proceedings of EQUADIFF 11: International conference on differential equations*. Comenius University Bratislava, Slovakia, 2005. Submitted.

[66] R. Liska and B. Wendroff. Comparison of several difference schemes on 1D and 2D test problems for the Euler equations. *SIAM Journal on Scientific Computing*, 25(3):995–1017, 2003.

[67] R. Loubere. Personal communication, 2005.

[68] R. Loubere and M. Shashkov. A subcell remapping method on staggered polygonal grids for arbitrary-Lagrangian-Eulerian methods. Technical Report LA-UR-04-6692, Los Alamos National Laboratory, 2004.

[69] R. Loubere and M. Shashkov. An arbitrary Lagrangian-Eulerian code for polygonal mesh: ALE INC(ubator). Technical Report LA-UR-05-3853, Los Alamos National Laboratory, Los Alamos, USA, 2005.

[70] R. Loubere and M. Shashkov. A subcell remapping method on staggered polygonal grids for arbitrary-Lagrangian-Eulerian methods. *Journal of Computational Physics*, 211(2):385–404, 2006.

[71] L. G. Margolin and M. Shashkov. Second-order sign-preserving remapping on general grids. Technical Report LA-UR-02-525, Los Alamos National Laboratory, 2002.

[72] L. G. Margolin and M. Shashkov. Second-order sign-preserving conservative interpolation (remapping) on general grids. *Journal of Computational Physics*, 184(1):266–298, 2003.

[73] M. M. Marinak, R. E. Tipton, B. A. Remington, S. W. Haan, and S. V. Weber. Three-dimensional simulations of ablative hydrodynamics instabilities in indirectly driven targets. *ICF Quarterly Report*, 5(3):168–178, 1995.

[74] B. Mirtich. Fast and accurate computation of polyhedral mass properties. *Journal of Graphics Tools*, 1(2):31–50, 1996.

[75] R. M. More, K. Warren, D. Young, and G. Zimmerman. A new quotidian equation of state (QEOS) for hot dense matter. *Physics of Fluids*, 31(10):3059–3078, 1988.

[76] C. Nieter and J. R. Cary. VORPAL: A versatile plasma simulation code. *Journal of Computational Physics*, 196(2):448–473, 2004.

[77] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 1999. ISBN 0-387-98793-2.

[78] J. O'Rourke. *Computational Geometry in C*. Cambridge University Press, Cambridge, UK, 1996. ISBN 0-521-64010-5.

[79] J. S. Peery and D. E. Carroll. Multi-material ALE methods in unstructured grids. *Computer Methods in Applied Mechanics and Engineering*, 187(3-4):591–619, 2000.

[80] T. Pisarczyk, S. Borodziuk, N. N. Demchenko, S. Yu. Guskov, M. Kalal, A. Kasperczuk, V. N. Kondrashov, B. Kralikova, E. Krousky, J. Limpouch, K. Masek, M. Pfeifer, P. Pisarczyk, K. Rohlena, V. B. Rozanov, J. Skala, and J. Ullschmied. Experimental and theoretical investigations of the crater formation process by means of double-target technique. In *Proceedings of 31st Conference on Plasma Physics*, volume 28G, pages 5066–5069. ECA, 2004.

[81] L. F. Richardson. *Weather Prediction by Numerical Process*. Cambridge University Press, 1922.

[82] W. Rozmus and A. A. Offenberger. Thermal conductivity for dense, laser compressed plasmas. *Physical Review A*, 31(2):1177–1179, 1985.

[83] W. D. Schulz. Two-dimensional Lagrangian hydrodynamic difference equations. *Methods of Computational Physics*, 3:1–45, 1964.

[84] M. Shashkov. *Conservative Finite-Difference Methods on General Grids*. CRC Press, Boca Raton, Florida, 1996. ISBN 0-8493-7375-1.

[85] M. Shashkov. Personal communication, 2005.

[86] M. Shashkov and S. Steinberg. Solving diffusion equations with rough coefficients in rough grids. *Journal of Computational Physics*, 129(2):383–405, 1996.

[87] M. Shashkov and B. Wendroff. The repair paradigm and application to conservation laws. *Journal of Computational Physics*, 198(1):265–277, 2004.

[88] A. I. Shestakov, M. K. Prasad, J. L. Milovich, N. A. Gentile, J. F. Painter, and G. Furnish. The radiation-hydrodynamic ICF3D code. *Computer Methods in Applied Mechanics and Engineering*, 187(1-2):181–200, 2000.

[89] L. Spitzer and R. Harm. Transport phenomena in a completely ionized gas. *Physical Review*, 89(5):977–981, 1953.

[90] B. Swartz. Good neighborhoods for multidimensional van Leer limiting. *Journal of Computational Physics*, 154(1):237–241, 1999.

[91] J. F. Thompson, F. C. Thames, and C. W. Mastin. Automatic numerical generation of body-fitted curvilinear coordinate system for field containing any number of arbitrary two-dimensional bodies. *Journal of Computational Physics*, 15(3):299–427, 1974.

[92] R. Tipton. *CALE Users Manual, version 910701*. Lawrence Livermore National Laboratory, 1991.

[93] E. F. Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics*. Springer Verlag, Berlin, Heidelberg, 1997. ISBN 3-540-61676-4.

[94] P. Váchal, R. Garimella, and M. Shashkov. Untangling of 2D meshes in ALE simulations. *Journal of Computational Physics*, 196(2):627–644, 2004.

[95] J. VonNuemann and R. D. Richtmyer. A method for the numerical calculation of hydrodynamic shocks. *Journal of Applied Physics*, 21(3):232–237, 1950.

[96] M. L. Wilkins. Use of artificial viscosity in multidimensional fluid dynamic calculations. *Journal of Computational Physics*, 36(3):281–303, 1980.

[97] A. M. Winslow. Equipotential zoning of two-dimensional meshes. Technical Report UCRL-7312, Lawrence Livermore National Laboratory, 1963.