TU KAISERSLAUTERN
FACHBEREICH MATHEMATIK

Efficient Algorithms for Flow Simulation related to Nuclear Reactor Safety.

TATIANA GORNAK

Datum der Disputation: 17.12.2013

Vom Fachbereich Mathematik der Technischen Universität Kaiserslautern zur Verleihung
des akademischen Grades Doktor der Naturwissenschaften
(Doctor rerum naturalium, Dr. rer. nat.) genehmigte Dissertation

1. Gutachter: Prof. Dr. Oleg Iliev
2. Gutachter: Prof. Dr. Peter Minev

D 386

Tatiana Gornak
Abteilung Strömungen und komplexe Strukturen
Fraunhofer-Institut fuer Techno- und Wirtschaftsmathematik (ITWM)
Fraunhofer Platz 1, 67663 Kaiserslautern, Germany.
gornak@itwm.fraunhofer.de
http://www.itwm.fhg.de

Abstract

Safety analysis is of ultimate importance for operating Nuclear Power Plants (NPP). The overall modeling and simulation of physical and chemical processes occuring in the course of an accident is an interdisciplinary problem and has origins in fluid dynamics, numerical analysis, reactor technology and computer programming. The aim of the study is therefore to create the foundations of a multi-dimensional non-isothermal fluid model for a NPP containment and software tool based on it. The numerical simulations allow to analyze and predict the behavior of NPP systems under different working and accident conditions, and to develop proper action plans for minimizing the risks of accidents, and/or minimizing the consequences of possible accidents. A very large number of scenarios have to be simulated, and at the same time acceptable accuracy for the critical parameters, such as radioactive pollution, temperature, etc., have to be achieved. The existing software tools are either too slow, or not accurate enough. This thesis deals with developing customized algorithm and software tools for simulation of isothermal and non-isothermal flows in a containment pool of NPP. Requirements to such a software are formulated, and proper algorithms are presented. The goal of the work is to achieve a balance between accuracy and speed of calculation, and to develop customized algorithm for this special case. Different discretization and solution approaches are studied and those which correspond best to the formulated goal are selected, adjusted, and when possible, analysed. Fast directional splitting algorithm for Navier-Stokes equations in complicated geometries, in presence of solid and porous obstales, is in the core of the algorithm. Developing suitable pre-processor and customized domain decomposition algorithms are essential part of the overall algorithm amd software. Results from numerical simulations in test geometries and in real geometries are presented and discussed.

Acknowledgments

I would like to express my very great appreciation to Prof. Dr. Oleg Iliev for his patient guidance and valuable suggestions in regards to my research. I would like to thank Dr. Aivars Zemitis for his very useful practical suggestions.

I had a privilege to work with Prof. Dr. Peter Minev during my visit to the University of Alberta. The work presented in this thesis would not have been possible without collaboration with him. I would also like to acknowledge funding from the DAAD (”Visit Research Grant” for work in University of Alberta during April/May 2013) which make this visit possible.

Furthermore I would also like to acknowledge the Graduirtenkolleg and Fraunhofer ITWM for their financial support.

Finally, I would like to thank my family and friends for their support and my dog Gina for the incredible joy with which she has greeted me each time I’ve returned home.

Contents

1 **Introduction and Motivation** **7**

  1.1 Motivation, Goals and Structure of the Thesis 7

  1.2 Support of complex geometries and immersed objects 12

    1.2.1 Immersed boundary methods 13

    1.2.2 Fictitious domain method. 31

    1.2.3 Simulation results 33

    1.2.4 Conclusion 34

2 **Models for geometry generation and simulation process** **36**

  2.1 Models for reactor safety 36

    2.1.1 Reduced model 37

    2.1.2 Lumped parameter model 39

    2.1.3 Full model 40

    2.1.4 Summary 46

  2.2 Geometry and grid generation 47

    2.2.1 Grid generation 50

    2.2.2 Sinks and sources. Room flooding 50

    2.2.3 Summary 54

3 **Numerical algorithm for isothermal flows** **55**

  3.1 Discretization of penalized Navier-Stokes equations 55

    3.1.1 Fractional time step discretization 55

    3.1.2 Spatial discretization 57

    3.1.3 Numerical experiments 63

  3.2 ADI method for the pressure correction equation 64

3.2.1 Numerical algorithm 68
3.2.2 Computational cost of the direction splitting operator vs. Laplace operator in the pressure correction equation 72
3.2.3 Numerical experiments 73
3.2.4 Summary 79
3.3 ADI for a domain with partially matching grid 80
3.3.1 Solution of a parabolic equation on a domain with partially matching grid 81
3.3.2 Application to the Navier-Stokes equations. 88

4 Numerical algorithm for non-isothermal flows 92
4.1 Fractional time step discretization of the Oberbeck-Boussinesq equations . . 92
4.2 Numerical experiments 94
4.2.1 Preliminaries 94
4.2.2 Side heated 3D cavity. 95
4.2.3 ThAI facility 96
4.3 Conclusion 100

5 Summary 101

Nomenclature 103

Acronyms 104

Bibliography 105

Chapter 1

Introduction and Motivation

1.1 Motivation, Goals and Structure of the Thesis

The main purpose of this thesis is development of customized fast algorithm for simulation of non-isothermal fluid flow in complex geometries with various obstacles inside.

This type of processes arises in some parts of containment buildings of nuclear reactors. A containment building is a passive safety system for some types of nuclear reactors (e.g. pressurized water reactors (PWR), boiling water reactors (BWR)). The main function of a containment building is to contain the escape of radiation in case of emergency. Containment is designed on the performance of its functions, taking into account all possible mechanical, thermal and chemical effects. Design of a containment building depends on the type of reactor and the specific external threats (e.g. seismic events). Classic containment building geometry is called can and consists of cylinder with a sphere on top of it. In the figure 1.1a one can see a twin PWR reactor containment at the Cook Nuclear Plant in Michigan, where each part has a can-type geometry. A containment building in a German PWR plant shows a nearly completely spherical containment design, which is very common for German PWRs (see figure 1.1c). A single unit of the Pickering Nuclear Generating Station (see figure 1.1d) has a slightly different from a typical PWR containment shape. Modern plants have tended towards a design that is not completely cylindrical or spherical, like containment at the Clinton Nuclear Generating Station (see figure 1.1b) [16].

In the field of nuclear power plants safety, the research has to ensure the safety of human and environment in a different situations on various types of accidents. So the design and operation of nuclear power plants aims to minimize the likelihood of accidents, and to

(a) Donald Cook Nuclear Power Plant. Image is a courtesy of Wikipedia.

(b) Containment at the Clinton Nuclear Power Station. Image is a courtesy of Daniel Schwen.

(c) German PWR. Image is a courtesy of Wikipedia.

(d) Pickering Nuclear Generating Station. Image is a courtesy of Ilker Ender.

Figure 1.1: Containment buildings.

avoid major consequences when they occur. There have been three major reactor accidents in the history of civil nuclear power - Three Mile Island, Chernobyl and Fukushima. One was contained without harm to anyone, the next involved an intense fire without provision for containment, and the third severely tested the containment, allowing some release of radioactivity [81].

So there is no question that processes in the containment building of nuclear power plants have to be efficiently simulated in connection with the analysis of different accidents scenarios.

Currently, essentially three types of software tools are used for this purpose: tools based on lumped parameter models, commercial CFD software tools, specially developed for reactor engineering tasks CFD codes.

The software tools based on lumped parameter models, e.g. COCOSYS are fast and have very well developed and validated description of the chemical processes. But on the other hand in case of heat transfer they reduce a thermal system to a number of discrete lumps and assumes that the temperature difference inside each lump is negligible. This approximation is useful to simplify the otherwise complex differential heat equations, but very

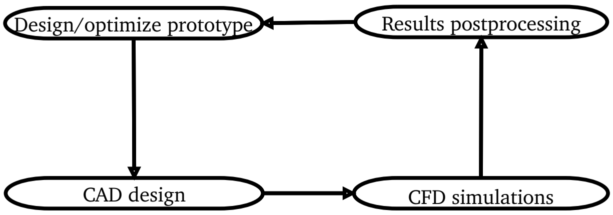


Figure 1.2: Engineering design process.

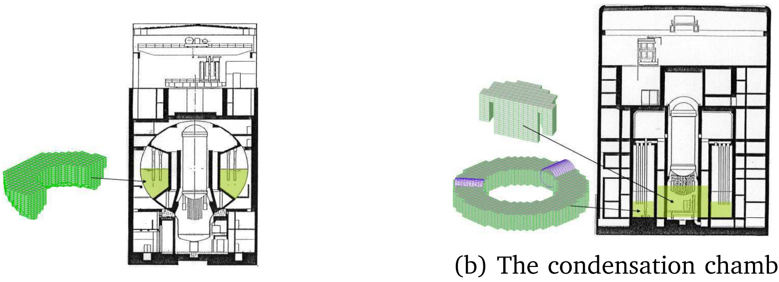
roughly accounting for the thermo stratification prediction. In a section 2.1.2 containment Code System COCOSYS which is based on lumped parameter models is discussed in details. Some other containment simulation codes such as CONTAINSTABLE [52], or integral codes such as MELCOR [79] or ASTEC [8] are concentrated on other processes in a containment pool, so they do not resolve desired CFD processes on the desired level.

Another class of software tools is general commercial CFD software (such as ANSYS Fluent [5], COMSOL [15]). These tool are able to resolve in details the thermo and hydrodynamics in the liquid zones, but the overhead of such general purpose software is the large computational costs and the complexity of their usage. For example such general purpose codes use special methods to simulate arbitrary deformations of the free surface, so in addition to the solution of the full fluid motion equation, computations of the kinematics of the free surface is also required. Since these codes usually work with the k–ϵ turbulence model, a sufficiently fine discretization is necessary for a reliable simulation.

It is often questionable whether a potential information which is gained from the CFD simulation justifies the considerable additional effort. Recent studies in the THAI project [1] have shown that the CFD models based on the k–ϵ turbulence model can't currently give more accurate predictions of thermal stratification.

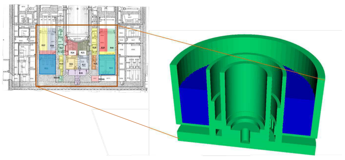
As it can be seen from the discussion above, the flow simulations for the need of the NPP safety analysis have very specific requirements and existing tool are unable to fulfill them, what demands development of customized algorithms and software. Let us shortly discuss the challenges and the priorities for such simulations.

- *Complex-shaped geometry of the containment pool.* The containment building inside is separated onto different zones, so different types of processes can arise there and should be taken into account: aerosol/iodine behavior, pool scrubbing, core-concrete interaction, unsteady heat transfer, fluid flow. As we've stated above we want to address non-isothermal fluid flow. That type of the processes can be found in following parts of a reactors building:



(a) The condensation chamber of BWR.

(b) The condensation chamber and the control rod drive chamber of BWR.



(c) One part of computational domain and cross section of the real domain.

Figure 1.3: Space areas in the containment buildings.

- the sump in the containment of a PWR,
- the condensation chamber of a BWR,
- the control rod drive chamber of a BWR.

Space areas in the containment building of PWR and BWR, where fluid flow may be present normally (condensation chamber at BWR, see figure 1.3a) or appear in the course of accident (the sump in the containment of a PWR, see figure 1.3b), have a complex-shaped three-dimensional geometry.

In the figure 1.2 main steps in engineering design process is shown. One of the important step is availability of the CAD data, but this requirement is not fulfilled in our case. As was explained above the containment pools usually have a complicated geometry (see figure 1.3c). Although the geometry is complicated, it can be seen that it can be very well approximated with a compound resulting from binary operations with simple objects. This fact is exploited for developing fast , such as and easy to use pre-processor [99]. We discuss geometry generation procedure in details in section 2.2.

• *Free surface flows.* Free surface flows occurs in the containment pool. However, the intrinsic free surface effects are important only during the first instances of some kinds of flooding. In most of the cases, due to the finite size of the rooms in the containment pool, the water is deep, and flat surface is a reasonable assumption. At the same time, careful water management is needed to reflect mass and momentum balance in case of localized sources/sinks. This latter issue is shortly discussed below.

- *Thermal stratification in the pool.* Typically, during an accident, often either cold water will be poured into a pool with hot water, or hot water will be poured into a pool with cold water. This leads to mixing and to temperature stratification in the pool. The latter is of high importance for the accuracy of the predictions, because the chemical reactions in the air and on the air/water interface are very much influenced by the surface temperature of the water. In case of thermal stratification the surface temperature may differ very much from the average temperature in the pool, what makes lumped parameters codes not accurate enough for such simulations. Calculation of the temperature stratification is important requirement for the simulation software.

The aim of the study is therefore to create the foundations of a multi-dimensional fluid pool model and software tool based on it. Based on the application of the model following requirements can be formulated: (i) temperature distribution and thermal stratification within the water; (ii) mass and energy conservation; (iii) support of submerged structures.

Further technical objectives are ability to obtain simulation results fast, and to make the software easy for use. It is important to note that the described problem is interdisciplinary, and has origins in the fields of fluid dynamics, numerical analysis, reactor technology and computer programming.

It is well known that computational modeling process consists from several important stages: problem definition, mathematical model (continuous model), numerical model (discrete model), software development and computer simulation. Until now we have defined our problem along with goals and requirements.

The thesis is organized as follows.

Next section of this Chapter deals with possible discretization techniques for solving flow problems in complicated domains on Cartesian grid. Main attention is paid to different variants of immersed boundary method, as well as on fictitious region method. The aim is to analyse and classify those methods with respect to criteria which are specific for containment pool simulations, and thus to make educated choice which discretization method to use. The results from this survey are published in [90].

Chapter 2 is dedicated to description and discussion on full and reduced mathematical models describing the flow of incompressible fluid (Section 2.1). Similar to above, the goal is to evaluate which model fits better to our specific requirements. Furthermore, in this Chapter the selected geometry and grid generation approaches, as well as their software implementation in a pre-processor tool, are discussed (Section 2.2). Both, the geometry generation and the grid generation are selected according to the specific requirements formulated above. For details on the pre-processor we refer to [99].

Chapter 3 is one of the main algorithmical Chapters in the thesis. It is dedicated to adjusting and analysing a fast directional splitting algorithm for solving unsteady isothermal incompressible Navier-Stokes-Brinkman equations in complicated domains. This is extension to the algorithm proposed in [42] for solving Navier-Stokes equations in parallelepiped. Section 3.1 presents the discretization of the penalized Navier-Stokes equations. Section 3.2 introduces ADI algorithm for the pressure correction equation, and discusses its analysis. As a part of this Section, numerical results illustrating the performance of the method are presented. The material from these two sections is published in [34]. Section 3.3 is dedicated to another challenging issue, namely, extending the fast directional splitting algorithm to the case of non-matching (more precisely, partially matching) grids.

Chapter 4 discusses fast directional splitting algorithm for solving unsteady non-isothermal incompressible Navier-Stokes equations in Obereck-Boussinesq approximation (Section 4.1). Simulation results for test and for real geometries are presented in Section 4.2.

Finally, some conclusions are drawn in the last Chapter.

1.2 State of the art: support of complex geometries and immersed objects

Solving flow problems in complex-shaped geometries is required in many scientific and practical problems. In such situation boundary fitted grids theoretically allow for achieving higher accuracy, but at the same time can face severe difficulties in developing robust grid generators. The alternative is to discretize the governing equations on a Cartesian, taking special care for the discretization near the boundaries. The choice of the discretization method, especially in practical problems, is governed by a number of criteria, which often do differ for the specific applications. Those criteria include different combinations of specific accuracy and robustness requirements for the solvers, for the grid generators, for the CPU time, for the qualification of the developers, for the qualification of the users, etc. As was stated above our task is developing a fast and moderately accurate algorithm for simulation of the thermal stratification in containment pools of nuclear power plants (NPP). Here we present a result on our review of the literature in the area of flow simulation on Cartesian grid. It should be noted that there exist excellent reviews on IB methods, e.g., [62], but they did not provide systematization according to our criteria (see below), and therefore we had to perform an own review. Mostly immersed boundary (IB) methods and fictitious domain (FD) methods, are used for solving flow problems on Cartesian grids in complex shaped geometry. While there is no big variety of FD methods, a number of IB methods were developed during the years. At the end we perform some numerical simulations on a classical test problem, aiming to understand if immersed boundary and fictitious region method give similar accuracy for the quantities which are of interest for us (the pressure drop is one of them).

1.2.1 Immersed boundary methods

Peskin introduced in 1972 a concept of immersed boundary (IB) method [69] where he used this method to compute flow patterns around heart valves. The main feature of this method was that in contrast to methods where body-fitted grid was used in order to represent immersed body accurately simulation was performed on Cartesian grid (Eulerian coordinate system) while the immersed boundary was represented by a set of elastic fibers, which moved with local flow velocity (Lagrangian coordinate system).

Depending on the way how boundary conditions are imposed IB methods can be divided into following categories:

- continuous forcing methods,
- discrete forcing methods,
- ghost cell immersed boundary methods,
- cut-cell methods.

Continuous forcing approach.

Series of papers [69, 70, 71, 72, 74] written by the group of C. Peskin introduces a continuous forcing approach for elastic boundaries. They consider simulation of a viscous incompressible fluid in a region containing immersed boundaries which interact with the fluid. The fluid motion is governed by Navier-Stokes equations:

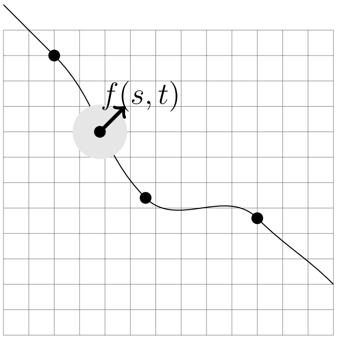


Figure 1.4: Shaded region is the area of force spread from fibers point.

$$\mathbf{u}_t + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \nabla^2 \mathbf{u} + \mathbf{F} \tag{1.1}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{1.2}$$

The boundary configuration is described by the curve $\mathbf{x}(s,t)$, where each value of a parameter s has a corresponding physical point of the boundary for all times t. The equation of motion of the boundary follows from the fact that boundary moves with the local flow velocity:

$$\mathbf{x}_t(s,t) = \mathbf{u}(\mathbf{x}(s,t))$$

The effect of immersed boundary is incorporated by transfering the fiber stress to the fluid. The forces which the fiber boundary exerts on the fluid are described by function $\mathbf{f}(s,t)$. The force field **F** can be derived from constitutive law (e.g. Hooke's law):

$$\mathbf{F} = \int_\Gamma \mathbf{f}(s,t)\delta(\mathbf{x} - \mathbf{x(s)})ds, \tag{1.3}$$

$$\mathbf{f} = \mathbf{M}(\mathbf{x}),$$

where δ is Dirac delta function, **M** is a non linear operator describing the elastic properties of the boundary. For the construction of difference equation from equation 1.3, special care should be taken for discrete representation of the δ function. This representation should appear from the fact that fiber stress in Lagrangian point is spread over surrounding grid nodes (see figure 1.4). Therefore the sharp delta function is replaced by smoother distribution function d, which is suitable for use on a discrete mesh:

$$\mathbf{F} = \int_{\Gamma} \mathbf{f}(s,t) d(\mathbf{x} - \mathbf{x(s)}) ds \tag{1.4}$$

Various variants of distribution functions were developed over the years, for details see [82, 55, 9].

This method has been applied to variety of problems in two and three space dimensions, including blood flow in the heart [76, 75], the design of prosthetic cardiac valves [73], platelet aggregation during blood clotting [28], aquatic animal locomotion [25] and many others. In [77] is mentioned that from time to time volume conservation in this type of methods is not exact.

Peskin's method is well suited for elastic bodies but has a limitation in case of rigid bodies. The origin of this limitation is in the application of a constitutive laws, namely Hooke's law (see equation (1.3)), which is not well posed in the rigid limit. One way to deal with this problem is to assume that the body is elastic, but extremely stiff, see e.g. the discussion in [62].

Another approach which allows to handle rigid boundaries, called *virtual boundary method*, was introduced in [32, 33]. The main idea of the virtual boundary method is to treat the body surface as a virtually existing boundary embedded in the fluid. The force filed from the boundary if formulated in a way that no-slip condition is handled on the surface.

As before, body surface is denoted by $\mathbf{x}(s,t)$. The force on the element of surface $\mathbf{f}(s,t)$ is determined by the requirement that the fluid velocity $\mathbf{u}(\mathbf{x},t)$ should satisfy the no-slip condition on the boundary:

$$0 = \mathbf{u}(\mathbf{x}(s,t),t) = \int_{\Omega} \mathbf{u}(\mathbf{x},t)\delta(\mathbf{x} - \mathbf{x}(s,t)) d\mathbf{x}$$

The body force is not known a priori, thus it must be calculated in a feedback way, so that the velocity on the boundary is used to determine the desired force [55]. In the virtual boundary formulation, the force is governed by the following feedback loop:

$$\mathbf{f}(s,t) = \alpha \int_{0}^{t} \mathbf{u}(\mathbf{x}(s,t),\tau) d\tau + \beta \mathbf{u}(\mathbf{x}(s,t),t),$$

where $\mathbf{u}$ is the fluid velocity at surface points, α, β are negative constants, chosen in such a way that **u** stays close to zero. This method was used for simulation of the two-dimension flow around a circular cylinder, three-dimensional plane- and ribbed-turbulent channel flow.

The main disadvantage of the virtual boundary method is that it contains two free constants that need to be tuned according to the problem being solved. In particular, for unsteady flows this forcing introduces a time step limitation that reduces the efficiency of the algorithm. Another disadvantage of the described method is that, in order to avoid equation stiffening and unphysical flow oscillations, the boundary forcing terms are spread across the boundary which therefore is smeared over the grid, thus decreasing the solution accuracy [29].

In [82] a modification of the forcing method, such that spurious oscillations caused by feedback forcing are eliminated, was suggested. Modified forcing function proposed there has a following form:

$$\mathbf{f}(s,t) = \alpha \int_0^t (\mathbf{u}(\mathbf{x}(s,t),t)dt - \mathbf{v}(\mathbf{x}(s,t),t)) + \beta(\mathbf{U}(\mathbf{x}(s,t),t) - \mathbf{v}(\mathbf{x}(s,t),t)), \tag{1.5}$$

where the velocity of the boundary is controlled by specifying velocity field **v** at the boundary points. In case of moving boundary ($\mathbf{v} \neq 0$) the position of the boundary point at each time step is computed by the integration of $\mathbf{v} = d\mathbf{x}(s,t)/dt$.

Thereby continuous forcing approach is a good choice when flow with elastic boundaries has to be simulated, but when flow problems involve rigid bodies then continuous forcing approach can lead to a stiff numerical system. Also, this type of methods unable to give for a sharp representation of the boundary, because of the smoothing of the forcing function.

Discrete forcing approach.

The idea of discrete forcing approach has first appeared in the work [65]. To describe that idea let us consider Navier-Stokes equations (1.1, 1.2). In this method the forcing term has to be defined in a way that the velocity **u**, tends to a desired value **v** on some immersed boundary Γ . If we know the velocity **u**, then the forcing term is simply:

$$\mathbf{F} = \begin{cases} \mathbf{u} \cdot \nabla \mathbf{u} - \nabla^2 \mathbf{u} + \nabla p + \frac{1}{\tau}(\mathbf{v} - \mathbf{u}^n), & \text{on } \Gamma \\ 0, & \text{elsewhere} \end{cases} \tag{1.6}$$

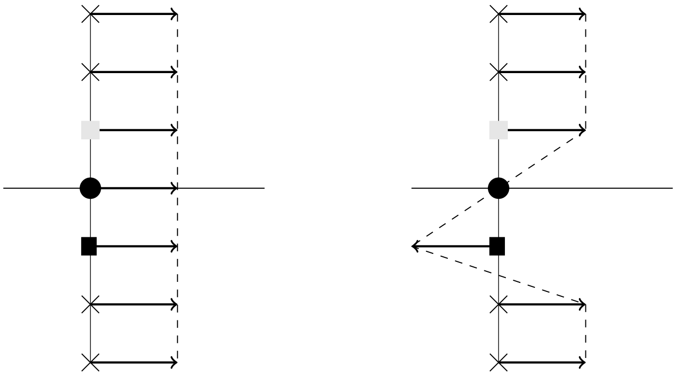


Figure 1.5: Sketch showing the effect of forcing. (a) initial velocity field, (b) velocity field imposed by forcing.

Most of the discrete forcing methods follow this idea on the continuous level, but differ on a discrete one. In general, the equation for the forcing (1.6), would be correct if the position of the unknowns on the grid coincide with that of the immersed boundary. In general this is not true because it would require the boundary to lie on coordinate lines or surfaces which is not the case for complex curvilinear geometries. Therefore, an interpolation procedure is needed [24]. For completeness, let us provide few examples of such interpolations.

In [65], forcing is applied to the grid points immediately inside the immersed boundary. As example, let us consider no-slip walls, so the velocity at the point immediately interior to the surface is forced to be reverse of the velocity at the point immediately exterior to the surface with interpolation. This approach is also called *mirroring*. In the figure 1.5 the effect of forcing is shown.

As shown in [24], methods based on mirroring satisfy the velocity boundary conditions with the accuracy of the interpolation method. In [58] it was reported that due to the reverse velocity field in the boundary cells, problems with mass conservation can arise. Furthermore, an adaptation of the method from [65] was given in [24].The main difference between methods introduced in [65] and [24] is in the interpolation procedure. In the proposed method the forcing is introduced at the first grid point outside the body using the velocity (u_i in figure 1.6) obtained by linearly interpolating the velocity at the second grid point (u_{i+1}) and the desired velocity ($\mathbf{v}$). The interpolation direction (the direction towards the second grid point) is either the stream-wise (x) or the transverse (y) direction. The wrong choice of the interpolation direction can generate problems in complex configurations.

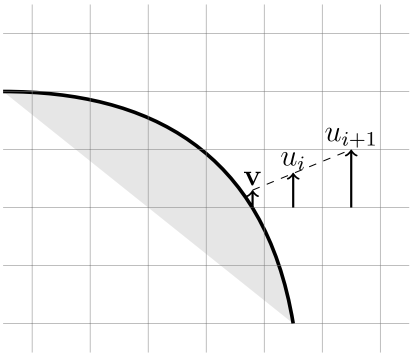


Figure 1.6: Velocity interpolation proposed in [24].

In the approach presented in [24], mass conservation at the immersed boundary is satisfied by the velocity fields both in fluid and solid regions. In this case, the (nonphysical) velocity field in the solid becomes important because it affects the pressure and velocity distribution through the velocity divergence across the immersed boundary. For the method proposed in [24] this issue can become very serious, since treatment of the velocities at the first grid points into the solid region is notionally undefined [46].

In [50] an approach for enforcing mass conservation in cells crossed by immersed boundary was proposed. The momentum forcing and mass source/sink in continuity equation were used to enforce the no-slip boundary conditions on the immersed boundary and to satisfy the continuity for the cells containing the immersed boundary. Let us shortly discuss the proposed mass source/sink approach. In [50], along with the momentum forcing, authors introduced modified version of the continuity equation:

$$\nabla \cdot \mathbf{u} - q = 0, \tag{1.7}$$

where q is mass source/sink is defined at cell center on the immersed boundary or inside the body. On a discrete level the source term has the form:

$$q = \frac{1}{\Delta V} \sum_i \omega \mathbf{u} \cdot \mathbf{n} \Delta S_i, \tag{1.8}$$

where ΔV is the cell volume, ΔS_i is the area of each cell face, and $\mathbf{n}$ is the unit normal vector outward at each cell face. Furthermore, ω equals to 1 in the points where momentum forcing is applied and 0 everywhere else. Thus, on a discrete level sinks and sources are incorporated in a way that the total sum of the volume fluxes due to the mass sources over the computational domain is zero and the global mass conservation is satisfied.

It is shown in [50] that mass source/sink term improves the quality of the solution and corrects the non-physical behavior. However, it was observed in [44] that the approximation used in [50], namely that the grid points fall on the immersed boundary when calculating the mass source/sink term, may lead to a degradation of the quality of the solution. In order to solve this problem, in [44] authors derive more accurate representation of the source/sink term (1.8), which now is calculated for a *virtual cell* of arbitrarily shape, where virtual cell is a cell with discarded solid part.

Different flow problems, such as decaying vortices and flows over a cylinder and a sphere, were simulated with the immersed boundary method proposed in [50] and [44].

One disadvantage of the proposed variants of the immersed boundary method is that these methods were successfully applied for flows with low and moderate Reynolds numbers, but faced difficulties with the simulations of high Reynolds flows.

In [13] finite volume immersed boundary method for complex incompressible flows and high Reynolds numbers was proposed. In this method the immersed boundary surfaces are defined as a cloud of points (which may be structured or unstructured). Immersed boundary objects are rendered as a level set in the computational domain, and concepts from computational geometry are used to classify points as being outside, near or inside the immersed boundary. The velocity field near an immersed surface is determined from separate interpolations of the tangential and normal components. Flow problems such as flow over circular cylinder, an in-line oscillating cylinder, a NACA0012 airflow. a sphere and stationary mannequin were successfully simulated with the proposed approach.

One advantage of the methods presented in this subsection is that forcing is determined from predicted velocity field, therefore there is no need in user-specified parameters (like α and β in the forcing function approach). Another advantage is that since the velocity boundary condition is enforced with implicit forcing, there is no limitation on the time step.

Ghost cell methods.

Formally, ghost cell method belong to a *discrete forcing*, but we will put it in a distinct section. The idea of the ghost cell method will be first described on the basis of ideas proposed in [91]. Then some improvement of this approach will be discussed. The goal of the this immersed boundary method is to achieve a higher-order representation of the boundary using a ghost zone inside the body.

In [91] the same idea as in [65, 24] is used: the force depends on the location and the fluid velocity, and thus it is a function of time (for details see equation (1.6)). As in the direct forcing methods, ghost cell methods define their own interpolation procedure, which is based on the concept of *ghost cells*.

Ghost cell is defined as a cell in the solid that has at least one neighbor in the fluid. In [91] the local flow variable ϕ is expressed in terms of polynomial, and it is used to evaluate ghost point values. The accuracy depends on the degree of the polynomial. Although polynomials of higher degree are expected to be more accurate, they often lead to boundedness problems and numerical instability. Linear and quadratic approaches, which preserve the second-order accuracy of the overall numerical scheme, are presented in [91].

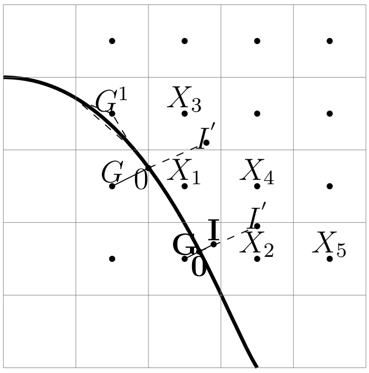


Figure 1.7: Various cell and node types: **G** – ghost cell, **0** – point at the boundary, $(X_1; X_2)$ – points in fluid for linear reconstruction, $(X_1; X_2; X_3; X_4; X_5)$ – points in fluid for bilinear reconstruction, **I** – image point from [22, 86], G^1 – additional ghost cell if the boundary is close to the fluid points, I' – image point like in [67], $\delta = |OI'|$ distance from point on immersed boundary (O) to image point (I').

As example let us consider linear reconstruction scheme for Dirichlet boundary conditions. In this scheme the ghost cell value is a weighted combination of the values at the nodes (X_1, X_2 and O, see figure 1.7):

$$\phi = a_0 + a_1 x + a_2 y \tag{1.9}$$

$$a = \begin{bmatrix} 1 & x_O & y_O \\ 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \end{bmatrix}^{-1} \begin{bmatrix} \phi_O \\ \phi_1 \\ \phi_2 \end{bmatrix} \tag{1.10}$$

For three-dimensional domains the interpolation scheme in equations (1.9) and (1.10) should be modified, because more neighbor nodes are involved. Thus, for linear reconstruction, the variable in the cell center is interpolated using four points (three nearest neighbor nodes and one boundary point are involved). The proposed scheme is equally applicable to both, steady and moving, boundaries. In the case of moving bodies, the points at which the boundary condition is enforced must be recomputed at every time step, but this does not affect the reconstruction scheme.

The major drawback with this extrapolation is that large negative weighting coefficients are encountered when the boundary point is close to one of the fluid nodes used in the extrapolation. Although algebraically correct, this can lead to numerical instability, i.e. the absolute value at the ghost point may be greater than the nearby fluid point values and the solution may not converge. Two approaches are used to remedy this difficulty.

The first approach is to use the image of the ghost node inside the flow domain to ensure positive weighting coefficients [22]. The point I is the image of the ghost node G through the boundary as shown in figure 1.7. The flow variable is evaluated at the image point using the interpolation scheme. The value at the ghost node is then $\phi_G = 2\phi_O - \phi_I$.

The other approach is to alter the piecewise linear boundary. When the boundary is close to a fluid node (normal distance of fluid point G^1 to the boundary $OG^1 < 0.1\Delta x$, Δx is the cell size) and far from the ghost point as in figure 1.7, we simply move the boundary point to the fluid node closest to the boundary [30]. Since the boundary is approximated as piecewise linear, the accuracy is hardly affected when the boundary segment is divided into two pieces. In [30] it is demonstrated that this approach could be used to obtain second order accuracy on irregular domains.

The above immersed boundary treatment focuses on the collocated grid arrangement. However it was stated in [91] that the ghost-cell approach can be extended to staggered grid arrangement in which all three velocity components and the pressure are computed on different grids. For each velocity component and for the pressure, one should find different weighted coefficients at the boundary, i.e., different linear system for each variable should be solved. This approach was validated on flow around a cylinder and on three-dimensional turbulent flow over wavy boundary.

Let us shortly comment on the mass conservation in [91]: ghost pressure is extrapolated from inside by the mirroring reflection procedure using Neumann condition. However, this practice leads to mass fluxes across the solid boundaries and mass error in ghost cells [86].

In [86], a ghost cell approach for staggered grids, whose primary feature is preserving the local mass conservation in each cell, was proposed. The idea is to satisfy the continuity equation directly for the ghost cells and to determine the pressure in the usual way through the Poisson equation. However, the mass errors should not be evaluated using the ghost velocities because they are not solutions of the momentum equations. Instead, the boundary velocities must be directly substituted, and the ghost velocities (outside the boundary) must be used only for the momentum equations. This practice preserves global continuity and avoids mass source/sinks in ghost cells. Also the idea to express the value in a ghost cell via values in image cell (like image point on figure 1.7), was used in [86].

The discussed method was applied to study shear- and buoyancy-driven flows in number of complex two dimensional cavities. In [86] it is also mentioned that authors had successfully implemented the method in three-dimensions.

In order to deal with boundary points which lie close to one of the fluid nodes, different concept of image point was introduced in [67]. An image point of the ghost cell is defined as the point which is a distance δ away from the body surface. The point $I^{'}$ in the figure 1.7 is an image point of new type. Based on the new image point, also new and more stable recontruction procedure was given. As example let us consider a extrapolation scheme for the velocity value at the ghost cell:

$$\mathbf{u}_G = \mathbf{v} - \frac{\mathbf{u}_{I'} - \mathbf{v}}{\delta} \|\mathbf{r}_G - \mathbf{r}_O\|, \tag{1.11}$$

where u_G is the velocity at the ghost point, **v** is the velocity at the immersed boundary, $\mathbf{u}_{I'}$ is the velocity at the image point, $\mathbf{r}_G$ and $\mathbf{r}_v$ are the position vectors of the ghost center and the projection point, respectively (various cell types are defined in the figure 1.7). Due to the fact that this model uses velocity value at the IB, it is able to handle moving bodies. It is claimed in [67] that the proposed method is second order accurate in space. Various steady and unsteady flows over a two-dimensional circular cylinder and a three-dimensional sphere were computed in order to validate the proposed method.

In [6] is noticed that approaches of obtaining ghost cell value, e.g. image point approach, are natural choices for smooth boundaries, but it is not that obvious for more irregular shaped geometries. Thus authors introduce an immersed boundary method for the solution of incompressible Navier-Stokes equations in the presence of highly irregular boundaries. This method is based on co-called a *local ghost cell* approach. This method extends the solution smoothly across the boundary in the same direction as the discretization it will be used for. The ghost cell value is determined locally for each irregular grid cell, making it possible to treat both sharp corners and thin plates accurately. The method was tested and validated for a number of problems including uniform flow past a circular cylinder, impulsively started flow past a circular cylinder and a flat plate, and planar oscillatory flow past a circular cylinder and objects with sharp corners, such as a facing square and a chamfered plate.

To summarize, among the advantages of the methods presented in this section are: (i) an ability to handle rigid bodies, (ii) absence of user defined parameters which may impact the stability of the methods, (iii) no need to compute flow variables inside a rigid body. Finally, since the boundary conditions are imposed directly in the numerical scheme, these methods are able to provide a sharp representation of the immersed boundary. A disadvantage of these methods is that they strongly depend on the discretization method. in contrast to the continuous forcing approach. It is worth noticing that methods in this category possess wide range of characteristics: three-dimension simulations, staggered grid, mass conservation.

Cut-cell methods.

The idea of cut-cell method is to use a uniform Cartesian grid over most of the domain with the Cartesian cells cut into a smaller irregular cells for any cell which is intersected by the boundary.

In the standard method of obtaining such cells, the immersed boundary is first represented by a series of piecewise linear segments, then cells that were cut by the immersed boundary are identified. After such a procedure, two types of cut-cells are created: cells whose nodes are in the fluid region (called regular cut-cells), and cells whose nodes are in the solid region (called small cut-cells). For an example of such cells see figure 1.12.

Given that a small cell can be arbitrarily small, discretization of the equations for these cells along the lines of that described for the regular cells can be highly problematic. While for convection-diffusion type of equations these small cells can cause CFL or viscous stability problems, for elliptic equations, such as the pressure Poisson equation, small cells produce ill-conditioned matrices that slow down the convergence of the iterative solution methods. Furthermore, discretization of these cells as separate finite-volumes changes the total number of unknowns that have to be solved for at any given time-step [85].

Another problem of the cut-cell method appears in the case of staggered grid: it is possible to have a velocity cell that does not have two pressure cells associated with it (an example of such a cell can be found in figure 1.10).

Each method cut cell method is characterized by three features:

1. way of obtaining sharp representation of immersed boundary,

2. finite volume discretization,

3. way of handling small cell and cells of non-standard shape.

For the methods described below. we will focus on the way they handle the third property.

Initially cut-cell method was introduced in [18] as Cartesian grid method for inviscid flow computations. Then in [94, 93] this method was adopted for solving two-dimensional incompressible fluid flow problems in the presence of both irregularly shaped solid boundaries and moving/free-phase boundaries. A related solver, developed in [98], uses a similar formulation but includes an improved interpolation scheme at the boundaries and a fractional time step method for time advancement.

In the method proposed in [98], small cut-cells are absorbed by neighboring cells. The result of this procedure is in the formation of trapezoidal control-volumes (see figure 1.8). Different treatments are applied to the uniform Cartesian cells and to the trapezoidal cells. For uniform Cartesian cells, the fluxes and pressure gradients on the face-centers can be computed with second-order accuracy by a simple linear approximation between neighboring cell-centers. This however is not the case for a trapezoidal boundary cell since the center of some of the faces of such a cell (marked by a shaded arrow in figure 1.8) may not lie in a location which puts it in the middle of neighboring cell-centers where a linear approximation would give an accurate estimate of the gradients.

In order to deal with flux evaluation for trapezoidal cells, it was proposed in [98] to express the flow variable ϕ in terms of two-dimensional polynomial interpolating function in

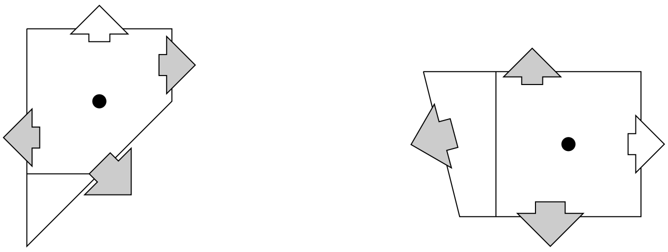
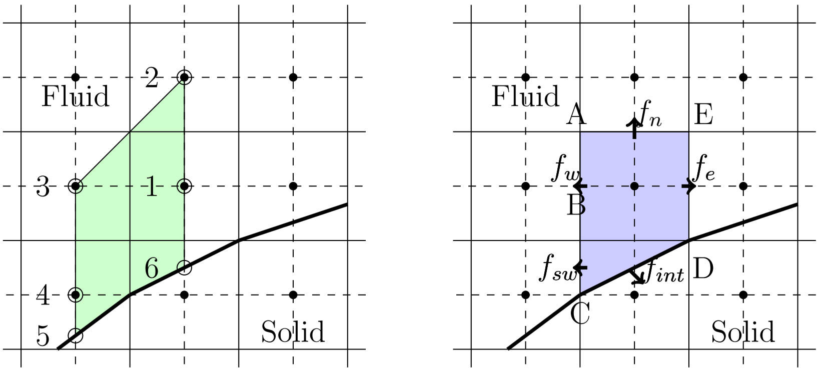


Figure 1.8: Typical reshaped trapezoidal boundary cells. Shaded arrows indicate fluxes that need special treatment.



(a) Various fluxes required for trapezoidal boundary cell.

(b) Trapezoidal region, and stencil used in computing f_{sw}.

Figure 1.9: Schematic of interpolation for cell face values and derivative at boundary cells.

an appropriate region, and then to evaluate fluxes (such as f_{sw} or f_e) based on interpolating function. For instance, in order to approximate flux f_{sw} (see figure 1.9a), ϕ is expressed in terms of a function that is linear in x and quadratic in y (phi is defined in the trapezoidal region shown in the figure 1.9b)):

$$\phi = c_1 x y^2 + c_2 y^2 + c_3 x y + c_4 y + c_5 x + c_6, \tag{1.12}$$

where c_1 to c_6 are six unknown coefficients.

It can be seen in the figure 1.9b that the sides of the trapezoid in which the interpolation is performed pass through four nodal points and two boundary points. Thus, the six unknown coefficients in (1.12) can be expressed in terms of the values of ϕ at these six locations (these locations are shown in the figure 1.9b)

$$\begin{bmatrix} \phi_1 \\ \phi_2 \\ \dots \\ \phi_6 \end{bmatrix} = \begin{bmatrix} x_1 y_1^2 & y_1^2 & x_1 y_1 & y_1 & x_1 & 1 \\ x_2 y_2^2 & y_2^2 & x_2 y_2 & y_2 & x_2 & 1 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ x_6 y_6^2 & y_6^2 & x_6 y_6 & y_6 & x_6 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \dots \\ c_6 \end{bmatrix} \tag{1.13}$$

Coefficients can now be expressed by inverting (1.13). When values of coefficients c_i are obtained, they can be used in order to evaluate the value of ϕ at the center of the face BC (see figure 1.9b) using interpolation similar to (1.12).

The overall solution procedure described in [98] is as follows:

- Determine the intersection of the immersed boundary with the Cartesian mesh.
- Using this information, reshape the boundary cells.
- For each reshaped boundary cell, compute and store the coefficients.
- Use these coefficients to develop discrete expressions and operators for the various terms in the discretized Navier-Stokes equations.
- Advance the discretized equations in time.

In [98], the authors claim that presented interpolation scheme coupled with the finite-volume formulation guarantees that the accuracy and the conservation property of the underlying algorithm re retained. In their test for the accuracy of the overall scheme they use simulations of Wannier flow [51]. This method, with some modification, has been used

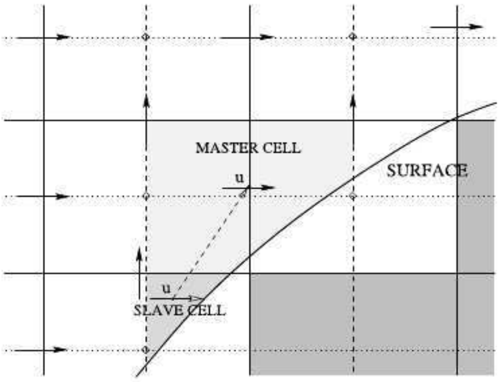


Figure 1.10: A master and slave cell are shown for the u component of velocity. The cell velocity has only one pressure node associated with it. It is moved to the same position as the master cell node. Image is a courtesy of [51].

to simulate various flows with stationary and moving boundaries, including flow-induced vibrations (see [59]), flapping foils (see [63]), objects in free fall through a fluid (see [64]), and diaphragm-driven synthetic jets (see [61]), moving boundaries (see [94]) [62].

The benefit of this method is that merging procedure eliminates the small-cell problem, but the merging process is highly complex, especially in three-dimensions, and can also lead to additional dependencies in the computational stencil (one need to calculate fluxes with additional diagonally adjacent neighbors) that can adversely impact convergence properties [85].

In [51], a second-order accurate Cartesian cut-cell method for the Navier-Stokes equations on a three-dimensional, non-uniform, staggered grid was presented. The article describes in details a finite volume discretization near immersed boundary cells. The main novelty of the proposed method is a cell-linking approach designed to avoid the problems related to the cell-merging procedure and to small-cells problem in case of staggered grid (see figure 1.10).

The idea of this approach is that instead of actually merging two cells in order to form a single cell, the two cells are linked as a *master/slave* pair in which the two nodes are made coincident while each cell remains a distinct entity. Because the slave node and slave cell continue to exist as entities separate from the master node and master cell, the fluxes and wall shear stresses, as well as the volumetric and surface information, are calculated in exactly the same way for the master and for slave cells, as they are calculated for the standard boundary cells.

The criteria used to determine when cell-linking is performed, are designed aiming to resolve problems associated with small cells. The first criteria is that any velocity cell which has only one associated pressure cell becomes a slave cell and is linked to a master cell. As a consequence, pressure gradient and velocity correction calculated for the master node can also be used for the slave node. The second criteria comes from fact that pressure cells with very small volumes can appear. This is overcome by requiring that the area of the larger face, in each pair of cell faces, is no less than 1% of the original cell face area. Pressure cells which do not meet this criterion are merged by removing the node from the calculation and treating the associated velocity cells as slave cells. Linking the master and slave velocity nodes is achieved in the following manner. The two nodes are not made exactly coincident but rather, the slave node is placed at a small distance from the master node. The diffusion flux between the two nodes then automatically becomes extremely high and forces the two velocities to take the same value. The described procedure is equally applicable to two- and to three-dimensional formulations.

The advantages of this method are obvious: (i) due to finite volume formulation, the method is mass conservative, (ii) it supports three-dimensions, and (iii) it supports staggered grid. A disadvantage is that the way in which master and slave velocity nodes are linked, can lead to problems for high Reynolds number simulations [43]. This method had been used to simulate flow in square driven cavity containing a circular cylinder, laminar flow through a channel placed skewed to the grid, flow past a circular cylinder (in two- and three-dimension).

In [12], cut-cell method, called the LS-STAG method, for staggered Cartesian grids and where the irregular boundary is sharply represented by its level-set function, was proposed.

As a result of applying level-set function, the curved boundary is approximated by linear line segments. As observed in figure 1.11, there are three basic types of cut-cells: trapezoidal cells such as $\Omega_{i,j}$, triangular cells (e.g. $\Omega_{i-1,j+1}$) and pentagonal cells ($\Omega_{i-1,j}$). The discretization of the momentum equations will be performed in the staggered control volumes $\Omega^u_{i,j}$ and $\Omega^v_{i,j}$, whose shape has to be adapted to each type of cut-cells. For example, in figure 1.11, the faces of $\Omega^u_{i,j}$ read:

$$\Gamma^u_{i,j} = \Gamma^{u,w}_{i,j} \cup \Gamma^{u,e}_{i,j} \cup (\Gamma^{s,e}_{i,j} \cup \Gamma^{s,w}_{i,j}) \cup (\Gamma^{ib,e}_{i,j} \cup \Gamma^{ib,w}_{i,j}) \tag{1.14}$$

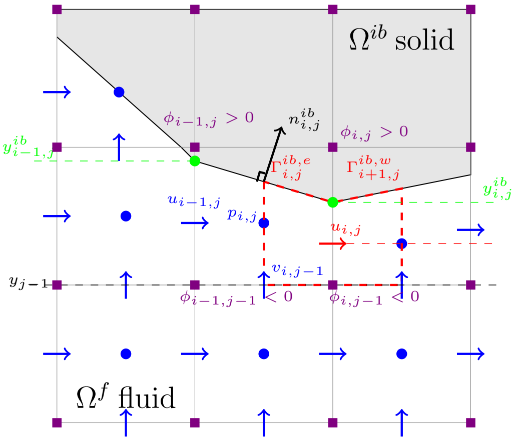


Figure 1.11: Staggered arrangement of the variables near the trapezoidal cut-cell $\Omega_{i,j}$ on the LS-STAG mesh, ϕ is a level set function which defines approximation of a curved boundary by linear line segments.

In [12] 12 types of such cut-cells in two-dimension and about 256 types of such cut-cells in three-dimensions, out of which 108 are admissible, are prescribed. The main feature of this method is that flow variables are actually computed in the cut-cells, and not interpolated. Furthermore, the LS-STAG method has the ability to discretize the fluxes in Cartesian and cut-cells in a consistent and unified fashion: there is no need for deriving an adhoc treatment for the cut-cells, which would be totally disconnected from the basic MAC discretization used in the Cartesian cells. Authors of [12] state that the method has conservation of the total mass, momentum and kinetic energy for flow with stationary boundary, but does not guarantee the discrete conservation of momentum or kinetic energy in the case of moving boundaries.

LG-STAG method was tested on flows at low to moderate Reynolds number: Taylor-Couette flow, flows past a circular cylinder, also simulation for flow with moving boundaries was performed: transversely oscillating cylinder flow in a free-stream.

The problem of pressure oscillations, observed when simulating moving boundary flow problems with sharp-interface immersed boundary methods (e.g. ghost cell method), was addressed in [85]. According to [85], the primary cause of pressure oscillations is the violation of the geometric conservation law near the immersed boundary. As a solution they adopt a cut-cell method to strictly enforce geometric conservation.

The incompressible Navier-Stokes equations are solved by a fractional time step method based algorithm. The key feature of this method is that the momentum equation is solved by a second-order central finite-difference discretization using the ghost-cell method from [60] (see section 1.2.1), and one does not need to spend additional efforts in computing momentum fluxes and stresses for the cut-cells. Furthermore, the stability restrictions caused by small cut-cells can be avoided. The cut-cell approach is applied to Poisson pressure equation and velocity correction equations in order to achieve better volume and mass conservation.

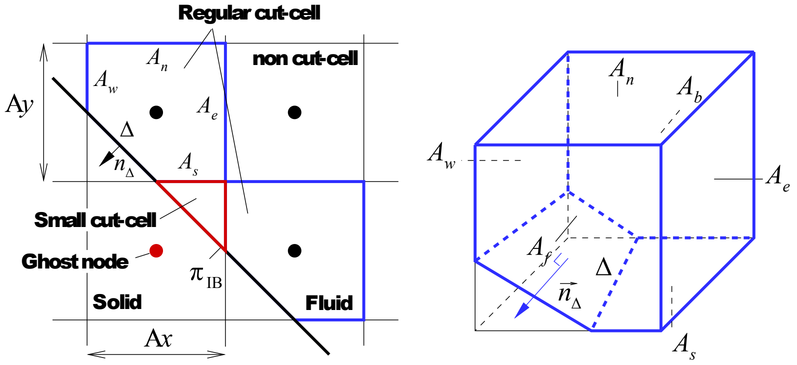


Figure 1.12: Cut-cell notations in 2D (left) and 3D (right, only one regular cut-cell is shown). Image is a courtesy of [85].

In this method the authors proposed different treatment for regular and for small cut-cells. For regular cut-cells, conventional discretization of the finite-volume equations are used (face-fractions and boundary surface for regular cut cell can be found on figure 1.12). A different approach is used for the small cells. The mass conservation associated with the small cells is accounted for via a *virtual cell-merging technique*. The idea it that source term of the Poisson equation for the small-cells is transferred to the adjacent cells (which may include regular cut-cells, as well as non-cut, regular Cartesian cells). The amount transferred to each target cell is chosen based on the direction of surface normal vector, as well as on the face area shared with the target-cell. This is consistent with the general notion that mass-transport associated with boundary motion would primarily be aligned with the direction normal to the boundary and the amount of mass flux would be proportional to the area. This method had been used to simulate flow around a circular cylinder and oscillating sphere.

An advantage of a cut-cell methods is that the cut-cell method is based on finite volume, so strict conservation of mass and momentum is guaranteed even in the vicinity of the immersed boundary. One of the drawbacks of this approach is that implementing the boundary conditions in irregular cells requires a large number of special treatments, which could result in complex coding logic. When using cut-cells, these cell should not become too small. Otherwise this could not only lead to stability problems, but also lead to slow convergence of the Poisson solver.

1.2.2 Fictitious domain method.

In this section we will recall the basics of the fictitious domain (FD) method. This method pursues the same goal as IB-methods: handle complex-shaped geometries without introducing a body-fitted grid. In contrast to most of the IB-methods, FDM's impletation is relatively simple.

The fictitious domain approach was introduced by Petrowsky [78] who utilized it to prove existence of a solution to the Dirichlet problem for the Poisson equation in a domain of a complex shape. The first works on fictitious domain method for Navier Stokes problems were published by [87], see also references in [95].

The basic idea behind the fictitious domain methods is that instead of solving the original equation(s) in complicated domain, properly perturbed equation(s) is solved in a simple domain. An abstract formulation of the fictitious domain method reads as follows (in this explanation we'll follow the ideas from [95]). Assume that in some area $D \subset \mathbb{R}^n$ we are seeking for solution $u(x)$ of the equation:

$$Lu = -\phi(x),\ x = (x_1, x_2, \ldots, x_n) \in D \tag{1.15}$$

with boundary conditions:

$$lu = g(x),\ x \in \partial D \tag{1.16}$$

Instead of solving the above problem in iregular domain D, we will be seeking for a solution of a perturbed problem in "n"-dimensional parallelepiped domain Ω. The perturbed problem reads as follows:

$$L_\epsilon u_\epsilon = -\phi^\epsilon(x), x = (x_1, x_2, \ldots, x_n) \in \Omega \tag{1.17}$$

$$l_\epsilon u_\epsilon = g^\epsilon(x), x \in \partial\Omega \tag{1.18}$$

The value of the small parameter ϵ introduces jump in the coefficient of the operator L_ϵ on the interface between the original domain D and the fictitious domain $D_0 = \Omega \setminus D$. Therefore, special care has to be taken to use proper discretization approach for the perturbed problem.

It can be shown that the solution of the perturbed problem tends in D to the solution of the original problem when ϵ tends to zero:

$$u_\epsilon(x) \xrightarrow[\epsilon \to 0]{} u(x), x \in D \tag{1.19}$$

more details can be found, e.g., in [95] or in other books/papers on fictitious domain method.

In order to solve (1.17)-(1.18) and get u_ϵ, usually a numerical method is applied. Therefore besides estimate for how close u_ϵ is to u (basis for fictitious domain method on a continuous level), we need a corresponding estimate for the numerical solution $u_{\epsilon h}$ (basis for fictitious domain method on a discrete level). So the full basis for fictitious domain method is based on following scheme: $u_{\epsilon h} \to u_\epsilon \to u$.

The current literature offers several possibilities to enforce Dirichlet or Neumann boundary conditions in a fictitious domain formulation. One of them is to include appropriate L^2 or H^1 penalty terms (see, e.g., [3, 4]). Another one is to enforce them as a side constraint via Lagrange multipliers (see for example [31]).

The fictitious domain method was introduced above in a general setting, below we will provide specific formulations of FDM for flow problems. The penalty can be introduced in two different ways.

Navier-Stokes equations with L^2 penalty term read as:

$$\begin{aligned} &\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot \nabla)\mathbf{u} - \nu\Delta\mathbf{u} + c_\epsilon \mathbf{u} + \nabla p = \mathbf{f}, \\ &\nabla\cdot \mathbf{u} = 0 \text{ in } \Omega \times [0, T], \end{aligned} \tag{1.20}$$

where

$$c_\epsilon = \begin{cases} \frac{1}{\epsilon^2}, & \text{in } \in \Omega \setminus D \\ 0, & \text{in } D \end{cases} \tag{1.21}$$

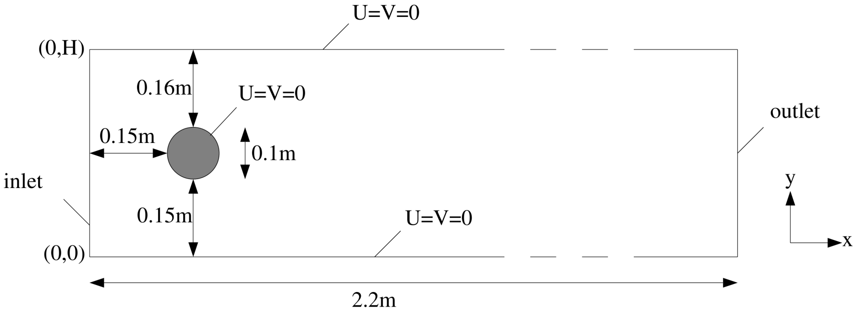


Figure 1.13: Test case geometry with boundary conditions. Image is a courtesy of [92].

Navier-Stokes equations with H^1 penalty term read as:

$$\begin{aligned} &\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot \nabla)\mathbf{u} - \nu^{\epsilon}\Delta \mathbf{u} + \nabla p = \mathbf{f}, \\ &\nabla\cdot \mathbf{u} = 0 \text{ in } \Omega \times [0,T], \end{aligned} \tag{1.22}$$

where

$$\nu^{\epsilon}(x) = \begin{cases} 1, & x \in D \\ \frac{1}{\epsilon^2}, & x \in \Omega \setminus D \end{cases} \tag{1.23}$$

1.2.3 Simulation results

After introducing immersed boundary and fictitious domain methods, in this Section we will present comparison between a distinct immersed boundary method and L^2-penalty fictitious domain method. For this comparison we've chosen classical benchmark problem: flow around circular cylinder. Thi sproblem allows us to evaluate two quantities which are of interest for us. First, how well the velocity field around a body with complex geometry can be computed using Cartesian grid, and how well the pressure drop can be computed.

In [92] a full description of several test cases values for reference parameters obtained in a range of simulations are given. We have chosen two dimensional steady flow around circular cylinder.

In the figure 1.13 simulation geometry and boundary conditions are shown. In order to set the inflow condition following expression has been used:

$$U(0,y) = 4U_m y(H-y)/H^2, V = 0$$

$Re = 20$	FD	IB	Benchmark
C_D	5.56	5.679	5.57-5.59
ΔP	0.1115	-	0.1172-0.1176
$Re = 100$	FD	IB	Benchmark
C_D	3.1	3.333	3.22-3.24
ΔP	2.43	-	2.46-2.5

Table 1.1: Values for the reference parameters for the 2D flow over a cylinder asymmetrically placed in a channel.

with $U_m = 0.3m/s$, yielding the Reynolds number $Re = 20$ and with $U_m = 1.5m/s$, yielding the Reynolds number $Re = 100$.

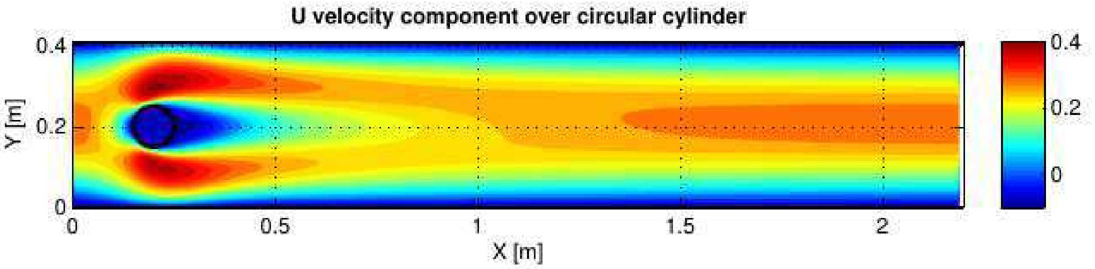
As a reference parameters we will use drag coefficient (C_D) and pressure drop $\Delta P = P(x_a, y_a, t) - P(x_e, y_e, t)$, between points $(x_a, y_a) = (0.15, 0.2)$ and $(x_e, y_e) = (0.25, 0.2)$. In [92] directions on how reference parameters should be computed can be found.

Table 1.1 presents values for the reference parameters for the simulation results computed with FD method, ghost cell immersed boundary method from [57] and range of parameters values obtained in [92]. In the figure 1.14 first and second components of the velocity computed with FD method and IB method from [96] are shown.

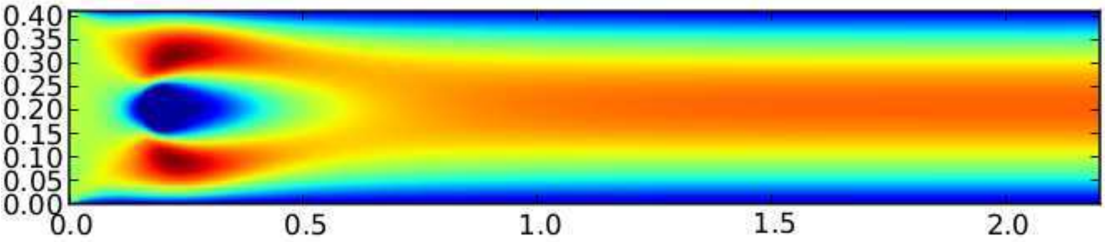
1.2.4 Conclusion

We have reviewed large amount of immersed boundary methods. These methods vary in the boundary conditions imposition approach, difficulty of implementation, field of application. In general all these methods provide a way to resolve complex boundaries on a Cartesian grid.

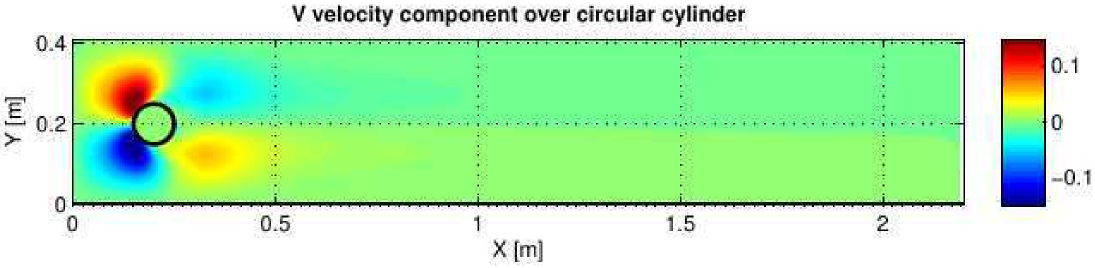
In section 3 we'll also reveal some additional arguments which are behind our choice.



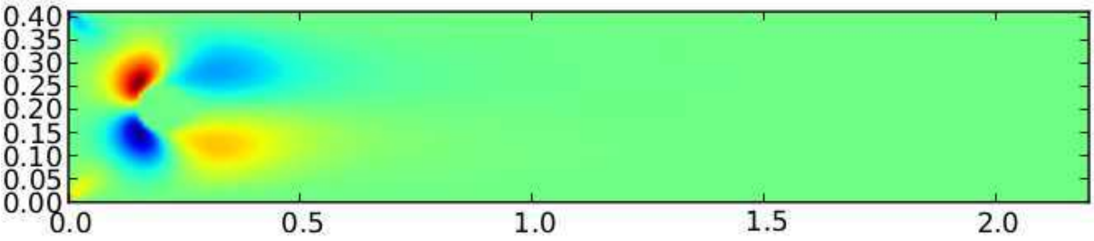
(a) First component of the velocity obtained with IB-method. Image is a courtesy of Reinout Vander Meulen [96].



(b) First component of the velocity obtained with FD-method; $u_{max} = 0.3966$



(c) Second component of the velocity obtained with IB-method. Image is a courtesy of Reinout Vander Meulen [96].



(d) Second component of the velocity obtained with FD-method; $v_{max} = 0.19$

Figure 1.14: Flow around cylinder computes with IB-method and FD-method.

Chapter 2

Models for geometry generation and simulation process

In this section we discuss two issues: (i) models which can be used for simulations related to reactor safety evaluation, and (ii) a proper approach for creating computational domain for containment pools and grid generation there.

2.1 Models for reactor safety

The physical processes in containment pools can be described by different mathematical models. Among them are shallow water equation, reduced 3D models based on hydrostatic approximation for the pressure, and full 3D Navier Stokes equations, and lumped parameter models. Important criteria for selecting a model in our case are: (i) how well the model accounts for the quantity of interest; (ii) can the model be solved in a fast and robust way. It is often questionable whether an information which is gained from the more accurate model justifies the considerable additional effort. We will try to answer this question, and for this we will discuss the reduced model, the lumped parameter model, and the full models. Shallow water equations would be suitable only for the first few minutes of flooding of an empty containment pool. Furthermore, they are expensive to solve, and it is difficult to find robust algorithms for SWE for the complicated containment pool geometries. Therefore shallow water equations are not considered here.

In the conclusion we will explain which model (any why) is more appropriate for our purposes.

2.1.1 Reduced model

Reduced model for the fluid motion

Reduced 3D Navier-Stokes equations [27] were studied to evaluate if they are suitable for simulation of thermal stratification in containment pools. In this case, under severe assumptions (shallow basin etc.) the momentum equation for the vertical reduces to equation for hydrostatic pressure. The continuity equation is used then for computing the vertical component of the velocity. No horizontal variation of pressure is possible with this model. For tests performed with the reduced model we refer to [100]. The reduced model is presented here for completeness.

Let Ω_f be a connected bounded domain in $\mathbb{R}^d$, where $d = 2$ denotes 2D domain and $d = 3$ denotes 3D domain. The fluid motion in the containment is given by time dependent Navier-Stokes equations on a finite time interval $[0, T]$:

$$\begin{cases} \dfrac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\,\mathbf{u} - \nu \Delta \mathbf{u} + \frac{1}{\rho}\nabla p = \mathbf{f}, \text{ in } \Omega_f \times [0, T] \\ \nabla \cdot \mathbf{u} = Q, \text{ in } \Omega_f \end{cases} \tag{2.1}$$

where **u** is a velocity, p is a pressure, ν is a kinematic viscosity, **f** denotes body forces, Q is a mass.

In case of 3D domain Ω Navier-Stokes equations can be rewritten for each velocity component in a following way:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right) + f_x, \tag{2.2}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z} = -\frac{1}{\rho}\frac{\partial p}{\partial y} + \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2}\right) + f_y, \tag{2.3}$$

$$\frac{\partial w}{\partial t} + u\frac{\partial w}{\partial x} + v\frac{\partial w}{\partial y} + w\frac{\partial w}{\partial z} = -\frac{1}{\rho}\frac{\partial p}{\partial z} + \nu\left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2}\right) + f_z, \tag{2.4}$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = Q. \tag{2.5}$$

In the reduced model the main assumption is that the pressure in the pool is hydrostatic. It means that the pressure at each point in the pool is defined by the thickness of the water layer above the point in the vertical direction. The flow in horizontal directions directly depends on the pressure gradient in the corresponding direction. In this case only two momentum equations are solved for the horizontal components of the velocity (equations (2.2) and (2.3) respectively). The third momentum equation (2.4) for the vertical component must not be solved. The vertical component of the velocity is estimated by the solving of the mass balance equation (2.5).

Let us summarize the sequence of the steps and equations which are used after applying proposed simplifications:

Hydrostatic pressure

A very important assumption is that pressure is defined by the hydrostatic distribution:

$$\frac{\partial p}{\partial z} = -\rho g. \tag{2.6}$$

Assume that the position of the water level at each time moment is described by the function $z = H(x, y, t)$. Under assumption that the pressure on the free surface is equivalent to the atmospheric one (the pressure jump is not taken into account) then pressure at any point can be estimated by integrating the equation (2.6):

$$p(x, y, z, t) = p_{atm} + \int\limits_{z}^{H(x,y,t)} \rho g dz. \tag{2.7}$$

Momentum equations in horizontal directions

Horizontal velocities are estimated from simplified momentum equations which can be written in a following way where the vertical convective terms are omitted:

$$\begin{aligned} \frac{\partial u}{\partial t} &= -\left(u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y}\right) - \frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right) + f_x, \\ \frac{\partial v}{\partial t} &= -\left(u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y}\right) - \frac{1}{\rho}\frac{\partial p}{\partial y} + \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2}\right) + f_y \end{aligned} \tag{2.8}$$

These two equations have to be discretized in space and time. The discretization is done in such a way that from each equation one component of the horizontal velocity can be estimated for the next time step. It means that these two momentum equations deliver the velocities in horizontal directions.

Vertical velocity component w from the mass balance

An important property of the described simplifications is that the vertical velocity is calculated from the mass balance equation.

Heat transfer

The temperature can be computed from full 3D convection diffusion equation, using the velocities computed with the reduced model. The reduced model is not very suitable for computing natural convection in closed domains, because in this case essential horizontal stratification of the temperature (and density) may occur.

2.1.2 Lumped parameter model

In order to simulate heat transfer in containment building, lumped parameter models are applied. The Containment Code System (COCOSYS) is a typical example of thermal hydraulic code which is based of lumped parameter model. COCOSYS is being developed and validated for the comprehensive simulation of severe accident propagation in containment of light water reactors. This system allows the simulation of relevant phenomena, containment systems and conditions during the course of design basis accidents and severe accidents. In COCOSYS, mechanistic models are used as far as possible for analyzing the physical and chemical processes in the containment. Essential interactions between the individual processes, like e.g. between thermal hydraulics, hydrogen combustion as well as fission product and aerosol behavior, are treated in an extensive way [54].

But the modeling temperature distribution in parts of containment filled with water is currently done in a primitive way. The compartments of the considered power plant (or other buildings) have to be subdivided into control volumes (zones). The thermodynamic state of a zone is defined by its temperature(s) and masses of the specified components. All components (liquid water, vapour and other non-condensable gases) are assumed to be mixed homogeneously, resulting in a homogeneous distributed temperature in the control volume (only one value) [53].

In the case of an accident the safety systems have to guaranty that the temperature in the plant remains in appropriate limits. Cooling with water is a very important procedure in this case. It is well known that during an interaction between cold and warm liquid in some cases thermal stratification can appear. For many physical processes which have to be taken into account the surface temperature of the liquid is essential. This temperature, in the case of thermal stratification, can be very different to the average volumetric temperature. Because COCOSYS is working with volumetric average temperatures in relatively large volumes, the effects of thermal stratification is difficult to simulate.

Lumped parameter model also unable to simulate convection flows within large water pools, because such an water volume typically is modeled as a single zone only, and only one set of values for the velocity vector and one value for the temperature are available there.

2.1.3 Full model

Models for flow simulation and temperature distribution from the previous sections have a lot of simplifications which affect physical accuracy. In this section we want to present from one hand side more complex model, but from the other hand side model which is more accurate from the physical point of view.

Fluid motion simulation

In this section we will describe mathematical model which can be used in order to simulate flow motion in a containment building of nuclear reactor. Here we should keep in mind that this building may have different wall inside it (permeable or impermeable).

Navier-Stokes-Brinkman equations

Let us consider more general domain Ω filled with fluid and with different solid and porous obstacles immersed in it: $\Omega = \Omega_f \cup \Omega_s \cup \Omega_p$, where Ω_p is a porous obstacles, Ω_s are solid obstacles in the domain Ω and Ω_f as before denotes to pure fluid regions of the domain. We assume that domain Ω_f is not necessarily connected, and the solid impermeable obstacles which separate it to subrooms belong to Ω_s (but not to Ω_p, which is reserved for porous, permeable obstacles). Now let us consider each part of the domain Ω separately in order to establish mathematical model there.

As before the fluid motion in the domain Ω_f is governed by incompressible Navier-Stokes equations (2.1).

In order to describe flow of fluid fluid through a porous medium Ω_p one can use Darcy's law [19].

$$\mathbf{v} = -\frac{k_p}{\nu}\nabla p \ \text{ in } \Omega_p, \tag{2.9}$$

where $\mathbf{v}$ denotes rate of flow through a surface elements of unit area, k_p is the permeability of porous mass.

Extension to the traditional form of Darcy's law called the Brinkman equation is used to account for transitional flow between boundaries [11]:

$$\nabla p = -\frac{\nu}{k_p}\mathbf{u} + \nu^{'}\Delta\mathbf{u} \ \text{ in } \Omega_p \tag{2.10}$$

where $\nu^{'}$ may take different from ν values. For example relation between $\nu^{'}$ and ν can be expressed via Einstein formula $\nu^{'} = \nu\,(1 + 2.5V)$ where V is the volume occupied by the dispersed solid [11]. But in case of our model we will assume that $\nu^{'} = \nu$.

To formulate problem in domain Ω we need review possible values of permeability. It should take different values in the fluid, solid and porous parts of the domain. In the porous medium Ω_p the permeability is assigned to k_p, but the fluid and solid domains it has to be treated differently:

$$K = \begin{cases} k_f \to +\infty & \textbf{in}\,\Omega_f \\ k_s \to +0 & \textbf{in}\,\Omega_s \\ k_p & \textbf{in}\,\Omega_p \end{cases} \tag{2.11}$$

Using this formula for permeability we can formulate the problem in the full domain Ω:

$$\begin{cases} \dfrac{\partial\mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\,\mathbf{u} \overbrace{- \nu\Delta\mathbf{u} + \dfrac{1}{\rho}\nabla p + \dfrac{\nu}{K}\mathbf{u}}^{\text{Brinkman equation}} = \mathbf{f} \ \text{ in } \Omega \times (0,T) \\ \nabla\cdot\mathbf{u} = Q \ \text{ in } \Omega \times (0,T), \end{cases} \tag{2.12}$$

This single system of equations describes the flow in the pure liquid, pure solid or the porous domains. This system of equations is called Navier-Stokes-Brinkman equations and also can be considered as penalized Navier-Stokes equations [48] or Navier-Stokes equations with fictitious domain method [7] [83]. It is clear that from equation (2.12), by means of the penalty method we can find equations for each subdomain, leaving only the leading terms and dropping negligible terms [48]:

- Ω_f: if $K = k_f$, the term $\nu K^{-1}\mathbf{u}$ tends to zero, so the equations in the fluid domain tend to Navier-Stokes equations.

- Ω_s: if $K = k_s$, the term $\nu K^{-1}\mathbf{u}$ forces the velocity to tend towards zero in the solid domain, therefore terms $\frac{\partial\mathbf{u}}{\partial t}$, $(\mathbf{u}\cdot\nabla)\,\mathbf{u}$, $\nu\Delta\mathbf{u}$ become negligible. So the equation in the solid domain tends to Darcy's law.

- Ω_p: if $K = k_p$, the term $\nu K^{-1}\mathbf{u}$ will be small and the equation in the porous domain will tend to Brinkman equation.

Sinks and sources.

Usually the second equation in the system (2.12), namely the incompressibility constraint $\nabla\cdot\mathbf{u} = 0$, is used in connection with incompressible fluids. Let us recall, that this is a particular case of the mass balance equation. Later on we will use perturbations to incompressibility constraint, therefore we will recall here how it is obtained.

Reynolds transport theorem states that the sum of the changes of some intensive property (ρ in this case) defined over a control volume must be equal to what is lost (or gained) through the boundaries of the volume plus what is created/consumed by sources and sinks inside the control volume:

$$\frac{d}{dt}\int\limits_{\Omega} \rho \, dV = -\int\limits_{\partial\Omega} \rho\mathbf{v}\cdot\mathbf{n}\, dA - \int\limits_{\Omega} Q\, dV \tag{2.13}$$

By applying divergence theorem to the equation (2.13) we get:

$$\frac{\partial\rho}{\partial t} + \nabla\cdot(\rho\mathbf{u}) + Q = 0 \tag{2.14}$$

And since we deal with incompressible fluid, for which the density is constant, the equation (2.14) reduces to incompressibility constraint.

Let us continue now with discussion on sinks and sources in our flow simulations. Due to the fact that the flat water-air interface is a reasonable approximation for our considerations, we do not consider a free surface problem. On the other hand, the water level in the containment pool changes dynamically due to leakages and/or the work of pumps. Mass balance equation helps to account for this dynamics, by introducing source and sink term. Depending on the accident, the containment pool can be in various states: (i) containment is full of water and water level doesn't change, (ii) containment is empty and water comes due to leakage, or the containment is full and the water is drained with pumps, (iii) or some combinations of scenarios above where water can come through several leakages and drain with several pumps, and different combination may appear in the different rooms of the pool. The pumps and the breakages in this cases acts like sinks and sources and are simulated with a mass balance equation 2.5, where a negative value of Q represents the sink and positive value of Q represents the source.

For example, if the leakage is distributed over all the surface of the water mirror, we introduce source terms for all near surface finite volumes. If the leakage is localized, we introduce source terms locally. If the leakage fluid comes with a low impulse, we introduce source terms only in the mass conservation equation. If, however, the leakage has a form of a jet, we introduce source terms also in the momentum equations, by prescribing relevant body force term **f**. Furthermore, the source/sinks can be located near the surface, but also can be located in the depth of the pool, to reflect that fact that there we may have pumps, exit from pipes, etc.

Note, that sinks and sources in the above equations are functions of space and time, and can appear and disappear in the course of the simulations, according to the simulated scenario (e.g., different leakages/sources acts for 5-10 min each in different locations and at different time).

Temperature in the water simulation: heat equation and Oberbeck-Boussinesq approximation

In this section we will introduce mathematical model which can be used in order to simulate a temperature distribution in a containment building of a nuclear reactor. Let us follow the logic from [56] and explain model which can be used in order to account the motion of an incompressible, heat-conducting fluid occupying a domain in $\mathbb{R}^3$.

If there is a mechanical equilibrium in a fluid then the temperature distribution depends only on a height z: $T = T(z)$. Absence of the equilibrium leads to an occurrence of internal fluid flows, which tries to mix the fluid so that a constant temperature is established. This process is called free or natural convection.

We will derive equations in assumption that we deal with incompressible fluid, basically this means that changes in density based on changes in pressure can be neglected. But on the other hand we can't neglected changes in density caused by non-uniform temperature distribution.

Let us introduce temperature in a following way $T = T_0 + T^{'}$, where T_0 is an average temperature and $T^{'}$ is a fluctuation of the temperature from the average value. We are assuming that $T^{'}$ is small in comparison to T_0. Fluid density can also be written in the same manner $\rho = \rho_0 + \rho^{'}$. Since the temperature fluctuation $T^{'}$ is small and the density fluctuation $\rho^{'}$ is also small, we can write the following relation:

$$\rho^{'} = \frac{\partial \rho_0}{\partial T} T^{'} = -\rho_0 \beta \tag{2.15}$$

where β is a thermal expansion coefficient. We can represent the pressure in a same way as the temperature and the density: $p = p_0 + p^{'}$. In this case p_0 is not a constant, but the value of the pressure in the case of mechanical equilibrium with T_0 and ρ_0 as values for temperature and density. The pressure changes with a height according to the hydro-static equation:

$$p_0 = -\rho_0 g z + \text{const} \tag{2.16}$$

Navier-Stokes equation in the presence of gravity reads as follows:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\, \mathbf{u} - \nu \Delta \mathbf{u} + \frac{1}{\rho} \nabla p = \mathbf{g} \tag{2.17}$$

Using expressions for pressure and density let us derive representation for term $\frac{1}{\rho}\nabla p$:

$$\frac{1}{\rho} \nabla p = \frac{\nabla p_0}{\rho_0} + \frac{\nabla p\prime}{\rho_0} - \frac{\nabla p_0}{\rho_0^2} \rho^{'} \tag{2.18}$$

Then substitute (2.15) and (2.16) to the equation (2.18) we get:

$$\frac{1}{\rho} \nabla p = \mathbf{g} + \frac{\nabla p\prime}{\rho_0} + \mathbf{g} T^{'} \beta \tag{2.19}$$

Finally if we substitute (2.19) to the Navier-Stokes equation and omit a zero sub-index in ρ_0, we get:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\, \mathbf{u} - \nu \Delta \mathbf{u} + \frac{1}{\rho} \nabla p = -\beta \mathbf{g} T^{'} \tag{2.20}$$

This modification to the Navier-Stokes equation is called Oberbeck-Boussinesq approximation and was introduced by A. Oberbeck in 1879 [66] and also by J. Boussinesq in 1903 [10].

In order to get the temperature distribution $T^{'}$ in the fluid we will use the heat equation:

$$\frac{\partial T^{'}}{\partial t} + (\mathbf{u} \cdot \nabla) T^{'} - \nabla \cdot (\lambda \nabla T^{'}) = 0, \tag{2.21}$$

where λ is a thermal conductivity.

In order to get the full system, which describes natural convection phenomena we need to add incompressibility constraint to the system of equations 2.20, 2.21. Thus, the full system in a domain $\Omega \subset \mathbb{R}^d$ $(d = 2, 3)$ reads as follows:

Navier-Stokes equations:

$$\begin{aligned} &\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\,\mathbf{u} - \nu \Delta \mathbf{u} + \frac{1}{\rho} \nabla p = -\beta \mathbf{g} T^{'}, \text{ in } \Omega \times [0, T] \\ &\nabla \cdot \mathbf{u} = 0 \text{ in } \Omega \times [0, T], \end{aligned} \tag{2.22}$$

Heat equation:

$$\frac{\partial T^{'}}{\partial t} + (\mathbf{u} \cdot \nabla) T^{'} - \nabla \cdot (\lambda \nabla T^{'}) = 0 \text{ in } \Omega \times [0, T] \tag{2.23}$$

The described system besides three unknowns (velocity vector, pressure and temperature), contains three parameters which determine its behavior: ν, λ, β. Usually in the literature the behavior of the natural convection is controlled by dimensionless Prandtl (Pr) and Rayleigh (Ra) numbers for Navier-Stokes equations, and by Peclet (Pe) number for the heat equation:

$$\begin{aligned} Pr &= \frac{\nu}{\lambda}, \\ Ra &= \frac{g \beta \theta L^3}{\lambda \nu}, \\ Pe &= \frac{UL}{\lambda}, \end{aligned} \tag{2.24}$$

with θ, L being characteristic temperature difference and characteristic length scale, correspondingly.

Prandtl number shows the ratio between viscosity and thermal diffusivity, while the ratio between the buoyancy and the viscous force acting on a fluid is given by Rayleigh number. The dimensionless form of the equations reads as follows:

Navier-Stokes equations in dimensionless form:

$$\begin{aligned} &\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} - Pr \Delta \mathbf{u} + \nabla p = Ra Pr \mathbf{g} T^{'} + \mathbf{f}, \\ &\nabla \cdot \mathbf{u} = Q \quad \text{in } \Omega \times [0, T], \end{aligned} \tag{2.25}$$

Heat equation in dimensionless form:

$$\frac{\partial T^{'}}{\partial t} + (\mathbf{u}\cdot \nabla)T^{'} - \nabla \cdot (\frac{1}{Pe}\nabla T^{'}) = 0, \text{ in } \Omega \times [0,T] \tag{2.26}$$

2.1.4 Summary

Let us summarize here the main advantages and disadvantages of presented models.

Reduced 3D model:

- *Advantages:* model is fast and simple.
- *Disadvantages:* model suffers from incomplete physics and has a restriction on the times step.

Full model:

- *Advantages:* adequate physics, weak restrictions on the time step.
- *Disadvantages:* as it will be shown in a section 3 this model requires to solve equation for pressure which usually slows down a computation process.

The Table 2.1 summarizes the features of the two approaches with a little bit more details.

Features	Reduced 3D model (TISA)	Full 3D Navier-Stokes Equations
CPU required	Fast, if small time step has to be used due to physical reasons. Slow in other cases.	Slower if small time step has to be used due to physical reasons. Fast if large time steps are allowed.
Robustness	Time step restrictions, sensitive	robust
Suitability for black box software	Needs a lot of tuning for the particular cases	The same approach works for all cases
Free surface	Can account for waves (it is not obvious that this is needed on a coarse grid in CoPool case)	flat surface approximation. reasonable for deep water

Limitations	Very suitable for ocean modeling, limited applicability for closed rooms with complex geometries.	no limitations
Horizontal internal walls	Difficulties due to the use of hydrostatic pressure.	No special attention is needed
Porous objects	Correction factors could be used, but the accuracy is questionable	Easy with Navier-Stokes-Brinkmann approach.

Table 2.1: Main characteristics of full and redice model.

The main advantage of the full model is that it can properly simulate thermal stratification in containment pools. In general, simulations based on the full model are slower, except in the cases when the reduced model has to use very small time step to account for small, and not important in our case, oscillations of the free boundary. Our decision is to use full model, and to improve its performance on the discrete level as much as possible.

2.2 Geometry and grid generation

As it was explained in the introduction, containment buildings in general, and the containment pools in particular, have a complex-shaped, but specific geometry. In many cases CAD data are not available, and a suitable way for creating the computational domain is needed. Taking advantage form the specific shape of the containment buildings, we create the computational domain as a result of Boolean operations on primitives, where the primitives are simple shaped objects, namely sphere, cylinder and parallelpiped. Such approach was used earlier in lumped parameter codes, but because the heat conduction in the walls is not calculated there, those developments were not suitable for our goals. Another reason why we could not use the existing geometry generating algorithms is the fact that for simulating the flooding from one room to another, we have to have proper classification of rooms. The classification will be shortly discussed below. Since we create computational domain in order to perform 3D simulations, a grid has to be generated there. It is well known in the CFD community that grid generation in complex domain may be as complicated as solving the problem itself. Because the tools which we develop have to be used by engineers with little CFD experience (but highly qualified in NPP safety analysis), we decided to avoid the generation of unstructured grid. Having also in mind that we target at moderate accuracy, but fast and robust algorithm, we discretize the flow equations on Cartesian grid, while the grid in the walls consists from overlapping pieces of orthogonal grids. For example, figure 2.1 shows spherical grid for the walls of the hemisphere, Cartesian grid for the fluid and for an obstacle, as well as the overlapping part. Both approaches, Chimera grids (overlapping grids) and Cartesian grid for complex geometry are widely used in the CFD, although not exactly in this combination.

To summarize, the idea behind geometry and grid generation is that they have to be simple, robust and reliable. Furthermore, the generated grids should provide preconditions for fast and reliable computations. To achieve these goals, the following demands have to be taken into account:

- easy construction of the continuous geometry representation: flow domain, walls and obstacles;
- easy generation of non-uniform Cartesian grid in the flow domain;
- automatic grid generation of orthogonal grids in local coordinates (Cartesian, spherical, cylindrical) for each object (walls, obstacles);
- easy specification of sinks and sources;
- reliable algorithm for classification of connectivity of rooms to be used in room flooding simulation.

Let us shortly present some details of the pre-processor.

Continuous geometry generation. At this step computational domain is created, based on geometry description provided by the user. The main idea is that the user has to describe the walls surrounding the fluid domain (the pool), as well as to describe the obstacles within the containment pool.

The basic geometrical objects which are used in the preprocessor are called *primitives*. Currently primitives of three type are presented: parallelepiped, sphere and cylinder. Each primitive is described by set of appropriate parameters such as length, radius, etc. Besides the parameters which describes the shape and the volume of the primitive, the user has also to specify its position and orientation in the space. In the figure 2.2 transformations of a cylinder are shown.

More complex objects, called here *compounds*, can be constructed on the basis of the primitives using Boolean operations: conjunction, disjunction and negation. This means

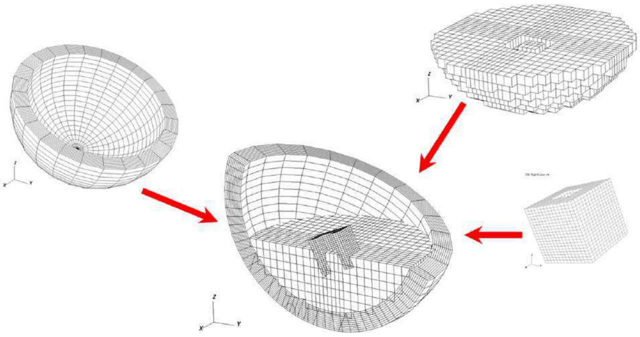


Figure 2.1: Geometry and grid.

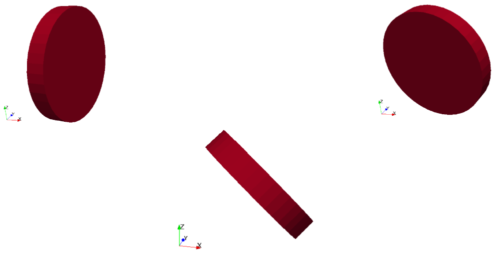


Figure 2.2: Transformation of the cylinder.

that compounds can be created by adding, cutting and overlapping different primitives.

Let us follow steps which need to be done in order to generate a hemisphere. Three primitives are used for this procedure: two spheres and one cuboid. Green sphere in the figure 2.3 is an outer wall of our desired hemisphere. Now we need to create an empty space inside it, thus we *cut* blue sphere of a smaller radius from the green one. As a result we get a hollow green sphere and then we cut red cuboid from the top of it. The whole process in summarized in the figure 2.3 where the quarter is cut from the green sphere for

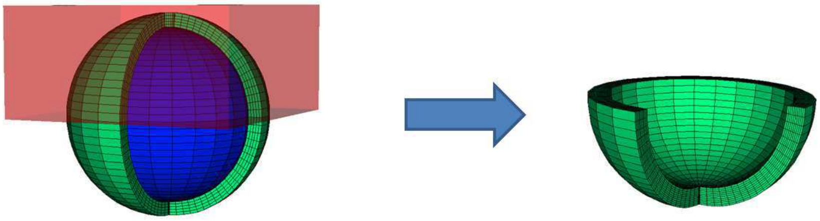


Figure 2.3: Generation of the hemisphere from primitives and Boolean operations.

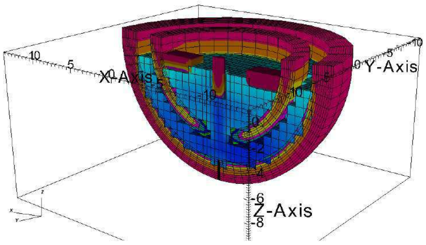


Figure 2.4: Typical pool geometry.

a better visibility.

Let us describe geometry generation using another, more complex example, which is closer to the typical pool geometry (see figure 2.4) and also uses some other Boolean operations. In this example let us assume that we already have created a hemisphere (see figure 2.5a). And as a result we want to have a hemisphere with flat bottom and an obstacle (see figure 2.5b) inside it. In order to create desired geometry we first overlap the hemisphere with a blue cuboid which will be a part of the flat bottom and then we add small cuboid as an obstacle. The whole process is shown on the figure 2.5

2.2.1 Grid generation

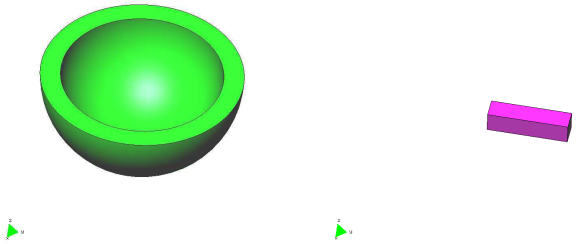
As mentioned above, we generate Cartesian grid in the fluid domain. Let us denote by Ω the part of the containment pool which is occupied by water. The domain Ω is partitioned into non-uniform Cartesian control volumes. Very often the domain Ω can not be represented exactly as a union of control volumes, typical situation is illustrated on figure 2.6. Usually two criteria are used to decide if a cell is to be assigned to the fluid part: (i) either if the center of the cell belongs to the fluid domain, or (ii) if the largest part of the cell volume is occupied by fluid. We use here the first criterion. On the right panel of figure 2.6 we illustrate a Cartesian grid in the fluid part, together with a uniform grid in spherical coordinates in a piece of the wall.

2.2.2 Sinks and sources. Room flooding

As it was explained in the section 2.1.3 sinks and sources play important role in the simulation of the accidents where the change of the water level has to be accounted for.

Let us go through few examples in order to understand what problems appear in the

Building blocks for pool geometry creation:



(a) Hemispere. (b) Obstacle.

Generation of continuous domain from basic building blocks:

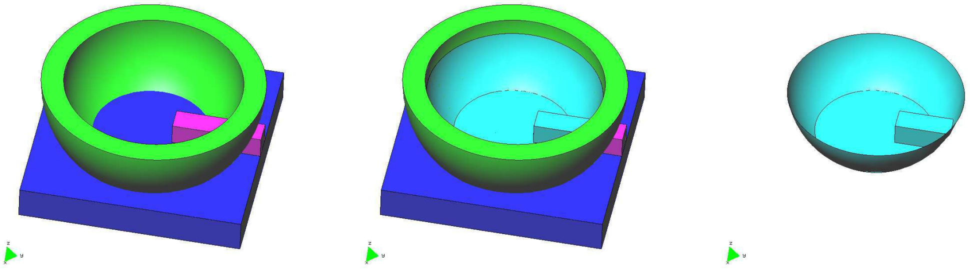


Figure 2.5: Continuous domain generation.

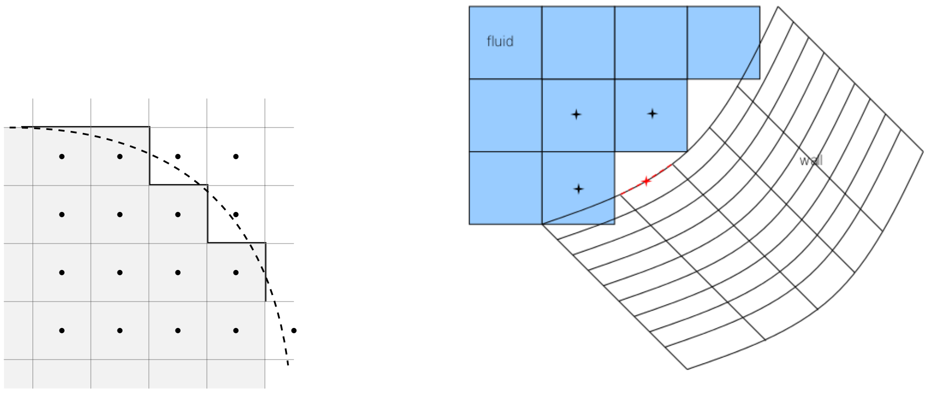
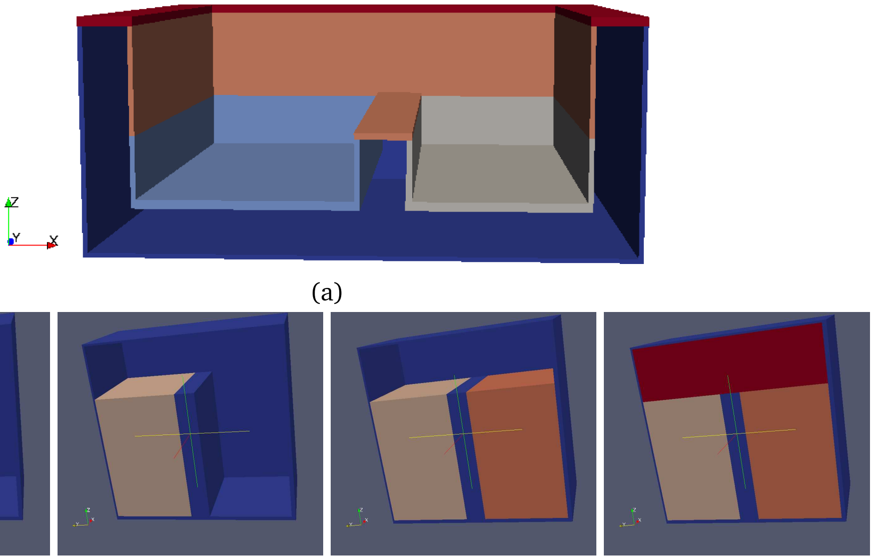


Figure 2.6: Domain approximation procedure.



(a)

(b) (c) (d) (e)

Figure 2.7: Flooding and room classification. Different colors represent different room parts.

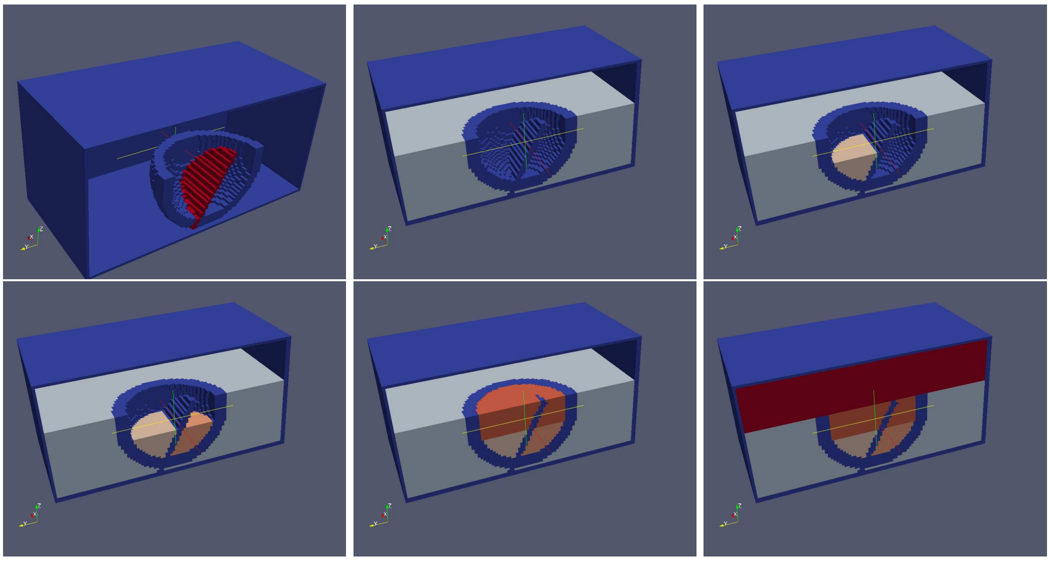


Figure 2.8: Flooding and room classification for the typical containment pool geometry. Pictures are generated based on vtk-files created as a result of running preprocessor.

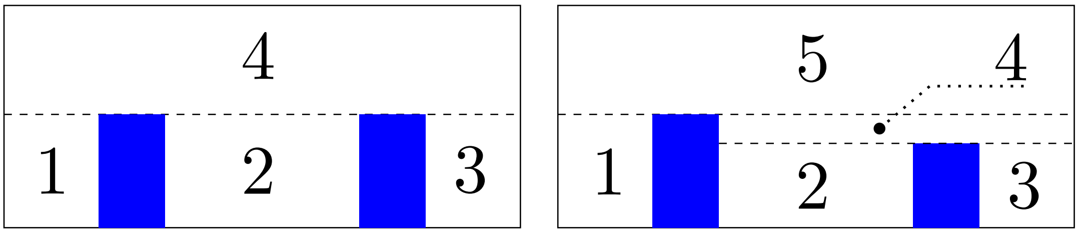


Figure 2.9: On the left the link layer (sub-room 4) connects 3 sub-rooms (1, 2 and 3). On the right two link layers can be seen. The link layer 5 connects sub-room 1 and sub-room 4. The link layer 4 connects sub-room 3 and sub-room 4. Blue blocks represent walls.

presence of sinks and sources. It should be noted that each control volume may act as a sink or as a source. In the simplest case the containment pool is full with water or water presences in a separated part of the containment (like light blue part in the figure 2.7a) and also a mass which comes through the source is equal to the mass which flow out through the sink. In this case no additional treatment is needed neither from the point of the preprocessor nor from the point of the simulation software. But let us consider another example: the containment is empty and there is a water source in the light blue room. In this example at first light blue room has to be filled with water, then the water level has to start increasing in a beige room and only when the beige room is full, then the water level should start to increase in a part of containment where wall painted with orange color (see figure 2.7a). Series of figures 2.7b – 2.7e shows desired flooding sequence (for convenience' sake color of the fluid corresponds to the room color). In case when simulation software does not simulate in details the water fall (the overflow of liquid from one sub-room to some other if appropriate limits are achieved) the other approach may be employed. In this approach simulation software on each timestep calculates a water level in each sub-room, so if rooms are connected the program treats them as a single domain if they are disconnected, then simulation runs there separately. In order to perform this calculations room classification has to be done on the geometry generation step.

In order to perform rooms classification preprocessor saves the information about what parts of the geometry can be flooded separately and which ones are connected. For each sub-room an appropriate number (room color) is assigned. Sub-room which connects two sub-rooms is called *link layer*. In the figure 2.9 several possible sub-rooms configuration are shown. Figure 2.8 shows sub-room colors and flooding process generated with the preprocessor for the geometry which is close to the typical containment pool geometry.

2.2.3 Summary

The preprocessor follows the idea that the creation of the geometry should be simple, as well as the generation of the grid for the computations, and the generated grid should provide preconditions for fast and reliable computations. Let us summarize main features of the preprocessor:

- generation of the wall geometry based on the description provided by the user,
- generation of structured wall meshes for the wall compounds,
- determination of overlapping parts for wall compounds,
- generation of structured fluid mesh,
- room classification concerning the flooding.

Chapter 3

Numerical algorithm for isothermal flows

3.1 Discretization of penalized Navier-Stokes equations

One of the difficulties in the numerical solution of the Navier-Stokes equations is that the velocity and the pressure are coupled. Fractional time step (FTS) methods, first introduced in the works of Chorin [14] and Temam [89], where they were referred to as projection methods, allow to overcome this difficulty. Application of the projection method to the Navier-Stokes equation means that at each timestep only a sequence of decoupled equations for velocity and for pressure have to be solved. After we have a sequence of equations to solve, a spatial discretization procedure has to be chosen and implemented. In section 3.1.2 we shortly describe finite volume discretization on a Cartesian staggered grid.

3.1.1 Fractional time step discretization

Different FTS methods were developed over the years (for the details see [97], [38], [49]). In this section we will present two variants.

Time discretization with special treatment of the penalization (Brinkman) term

Let τ be time step and let us set time $t^n = n\tau$ for $0 \leq n \leq N = [T/\tau]$. By u^0 we denote the exact velocity field at $t = 0$ which is a part of given initial conditions. First we split equation (2.12) in time (here we assume that that $Q = 0$):

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\tau} + (\mathbf{u}^n \cdot \nabla)\,\mathbf{u}^* - \nu \Delta \mathbf{u}^* + \frac{1}{\rho}\nabla p^n + \frac{\nu}{K}\mathbf{u}^* = \mathbf{f}^n \tag{3.1}$$

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^*}{\tau} + \left(\frac{1}{\rho}\nabla p^{n+1} - \frac{1}{\rho}\nabla p^n\right) + \left(\frac{\nu}{K}\mathbf{u}^{n+1} - \frac{\nu}{K}\mathbf{u}^*\right) = 0 \tag{3.2}$$

$$\nabla \cdot \mathbf{u}^{n+1} = 0 \tag{3.3}$$

where $\mathbf{u}^n$, $p^n, \mathbf{u}^*$, $p^*, \mathbf{u}^{n+1}$, p^{n+1} stand for pressure and velocity at time $t^n, t^{n+1/2}, t^{n+1}$ respectively. The intermediate velocity field, $\mathbf{u}^*$, is not divergence-free in general.

Now, let us apply divergence operator $\nabla\cdot$ to equation (3.2):

$$\nabla \cdot \left(\left(\frac{1}{\tau} + \frac{\nu}{K}\right)\mathbf{u}^{n+1}\right) - \nabla \cdot \left(\left(\frac{1}{\tau} + \frac{\nu}{K}\right)\mathbf{u}^*\right) + \nabla \cdot \left(\frac{1}{\rho}\nabla p^{n+1} - \frac{1}{\rho}\nabla p^n\right) = 0 \tag{3.4}$$

Then by dividing the equation (3.4) by $\left(\frac{1}{\tau} + \frac{\nu}{K}\right)$ and applying divergence free condition (3.3) we get the following equation:

$$\nabla \cdot \left(\left(\frac{\rho}{\tau} + \frac{\rho\nu}{K}\right)^{-1} \nabla \delta p\right) = \nabla \cdot \mathbf{u}^*,\ \delta p = p^{n+1} - p^n \tag{3.5}$$

By solving equation (3.5) we can calculate pressure correction δp and then using its value restore pressure and velocity for next timestep. Here is the full algorithm:

Velocity predictor:

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\tau} + (\mathbf{u}^n \cdot \nabla)\,\mathbf{u}^* - \nu \Delta \mathbf{u}^* + \nabla p^n + \frac{\nu}{K}\mathbf{u}^* = \mathbf{f}^n \tag{3.6}$$

Pressure corrector:

$$\nabla \cdot \left(\left(\frac{\rho}{\tau} + \frac{\rho\nu}{K}\right)^{-1} \nabla \delta p\right) = \nabla \cdot \mathbf{u}^*,\ \delta p = p^{n+1} - p^n \tag{3.7}$$

Pressure update:

$$\begin{aligned} p^{n+1} &= p^n + \delta p \\ \mathbf{u}^{n+1} &= \mathbf{u}^n + \left(\frac{\rho}{\tau} + \frac{\rho\nu}{K}\right)^{-1} \nabla \delta p \end{aligned} \tag{3.8}$$

Semi-implicit discretization of the Navier-Stokes equations

Here is another approach which can be used for the time discretization of the Navier-Stokes-Brinkman equations. Let us denote by p^0 the exact pressure field at $t = 0$, and by ϕ^0 an approximation to $\Delta t \partial_t p(0)$. The initial pressure p^0 is not part of the initial data but it can be computed from $\Delta p^0 = \nabla \cdot (\mathbf{f}_0 + \Delta \mathbf{u}_0)\,, \quad \partial_n p^0_{|\partial\Omega} = (\mathbf{f}_0 + \Delta \mathbf{u}_0) \cdot n$ under certain compatibility conditions on the data (see section 2.3 of [40]). The approximation ϕ^0 can then be computed using a non-incremental scheme at the first time step. The scheme proceeds for all $n \geq 0$ as follows:

Pressure predictor:

$$p^{*,n+1/2} = p^{n-1/2} + \phi^{n-1/2} \tag{3.9}$$

Velocity update:

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\tau} + (\mathbf{u}^n \cdot \nabla)\,\mathbf{u}^{n+1} - \frac{1}{2}\nu\Delta(\mathbf{u}^{n+1} + \mathbf{u}^n) + \frac{1}{\rho}\nabla p^{*,n+1} + \frac{\nu}{2K}(\mathbf{u}^{n+1} + \mathbf{u}^n) = \mathbf{f}^{n+1} \tag{3.10}$$

Pressure corrector:

$$\Delta\phi^{n+1} = \frac{\rho\nabla \cdot \mathbf{u}^*}{\tau} \tag{3.11}$$

Pressure update:

$$p^{n+1} = p^n + \phi^{n+1} - \frac{1}{2}\chi\nu\nabla\cdot(\mathbf{u}^{n+1} + \mathbf{u}^n),\ \chi \in [0,1] \tag{3.12}$$

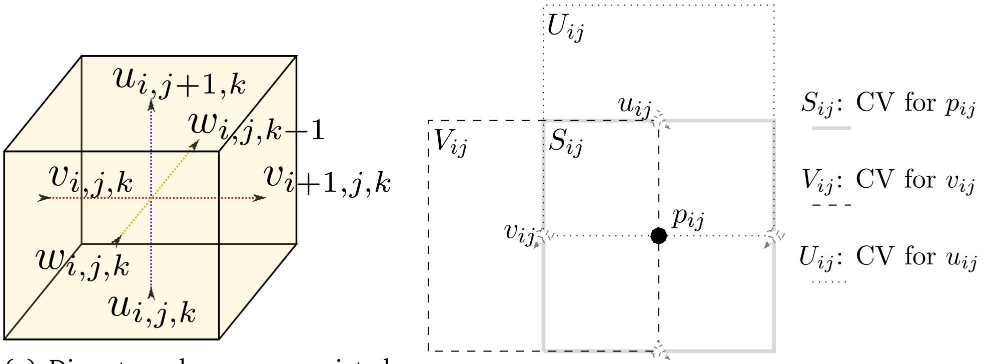
3.1.2 Spatial discretization

The equations obtained after the time discretization procedure, namely (3.6)–(3.8) and (3.9)–(3.12), are discretized in space on a rectangular Cartesian staggered grid using a finite volume method [23] [26]. We will explain the spatial discretization on a set of equations (3.6)–(3.8).

As it was described in chapter 2.2, the domain Ω is partitioned into non-uniform Cartesian control volumes S_{ijk} associated to the scalar grid nodes.

Let's introduce the following notation: with superscript h $(\Omega^h, \Omega^h_f, \Omega^h_p, \Omega^h_s)$ we will denote domains obtained after domain approximation procedure; as $\mathbf{c}_{ijk} = (x_i, y_j, z_k)$ and $\mathbf{h}_{ijk} = (h^x_i, h^y_j, h^z_k)$ we will denote center and size of control volume S_{ijk}.

The discrete value of the pressure p_{ijk} is stored in center $\mathbf{c}_{ijk}$ of a control volume S_{ijk}.



(a) Discrete unknowns associated with grid.

(b) Control volumes (CV) on staggered grid

Figure 3.1: Staggered Cartesian grid arrangement.

In contrast, discrete values of the velocity components ($u_{ijk}, v_{ijk}, w_{ijk}$) are stored at nodes that are staggered with respect to the pressure grid nodes, in other words point of a velocity component is defined on a center of control volume face, so components of velocity $u_{ijk}, v_{ijk}, w_{ijk}$ are defined in points $(x_i + \frac{h_i}{2}, y_j, z_k), (x_i, y_j + \frac{h_j}{2}, z_k), (x_i, y_j, z_k + \frac{h_k}{2})$ respectively. From this location of discrete unknowns follows that each unknown has its own control volume. Figure 3.1 shows described arrangement.

Momentum equation

Momentum equation (3.10) is solved for each component of the velocity independently, so for brevity let us describe the finite volume algorithm for the first component of velocity in case of a 2D domain.

In order to obtain finite volume method formulation for momentum equation we integrate governing equation for x-velocity component (u) over the control volume, corresponding to U_{ij} (rectangle $ABCD$ in the figure 3.2):

$$\iint\limits_{U_{ij}} \left(\frac{\partial u}{\partial t} + (u \cdot \nabla)\, u - \nu \Delta u + \frac{1}{\rho} \nabla p + \frac{\nu}{K} u \right) dU_{ij} = 0 \tag{3.13}$$

For the volume integral of the diffusive term $\nu \Delta u = \nu \nabla \cdot (\nabla u)$ we can apply divergence theorem:

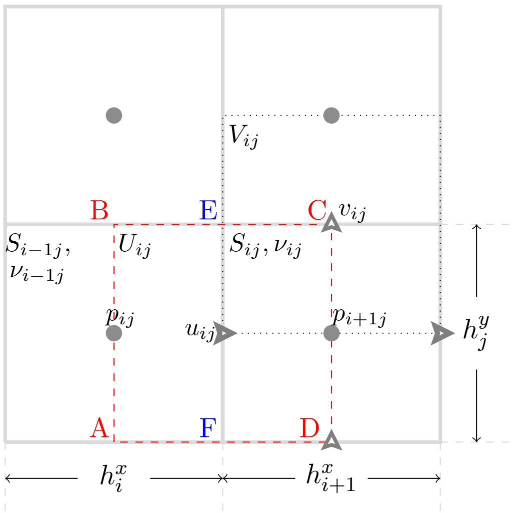


Figure 3.2: Integration over x-component velocity control volume U_{ij}

$$\iint\limits_{V} (\nabla \cdot F)\, dV = \int\limits_{S} (F \cdot n)\, dS \tag{3.14}$$

It will give us the following equation:

$$\iint\limits_{U_{ij}} \left(\frac{\partial u}{\partial t} + (u \cdot \nabla)\, u + \nabla p + \frac{\nu}{K} u \right) dU_{ij} - \int\limits_{\partial U_{ij}} (\nu \nabla u) \cdot n dS = 0 \tag{3.15}$$

Volume integrals discretization.

$$\iint\limits_{U_{ij}} \left(\frac{\partial u}{\partial t} + (u \cdot \nabla)\, u + \nabla p \right) dU_{ij} \approx \left(\frac{\partial u_{ij}}{\partial t} + \left[u \frac{\partial u}{\partial x} + u \frac{\partial v}{\partial y} \right] \bigg|_{(x_i + \frac{h^x ij}{2}, y_j)} + \frac{\partial p}{\partial x} \bigg|_{(x_i + \frac{h^x ij}{2}, y_j)} \right) |U_{ij}|, \tag{3.16}$$

where $|U_{ij}| = h^y_{ij}(\frac{h^x_{ij}}{2} + \frac{h^x_{i+1j}}{2})$ is a volume of the control volume U_{ij}.

Pressure gradient discretization reads as follows:

$$\frac{\partial p}{\partial x} \bigg|_{(x_i + \frac{h^x ij}{2}, y_j)} \approx \frac{p_{i+1,j} - p_{ij}}{\frac{h^x_{ij}}{2} + \frac{h^x_{i+1j}}{2}} \tag{3.17}$$

For the discretization of non-linear convective terms we are applying upwind scheme, where velocity values for the derivative discretization is taken from the upstream direction:

$$u\frac{\partial u}{\partial x}\bigg|_{(x_i+\frac{h^xij}{2},y_j)} \approx \begin{cases} u_{ij}\dfrac{u_{ij}-u_{i-1j}}{h^x_{ij}} \text{ if } u_{ij}>0, \\ u_{ij}\dfrac{u_{i+1j}-u_{ij}}{h^x_{i+1j}} \text{ if } u_{ij}<0, \end{cases} \tag{3.18}$$

The other convective terms have also to be also approximated. It is necessary to take into account the fact that different velocity components are located at different places. The convective term $v\frac{\partial u}{\partial y}$ has to be approximated in the momentum equation related to u. As it was stated earlier, velocity component u is located in a point with the coordinates $(x_i+\frac{h^x_{ij}}{2}, y_j)$ for the control volume U_{ij} . Therefore if we want to get an approximation for the convective term we have to find the value of the velocity component v in the point $(x_i+\frac{h^x_{ij}}{2}, y_j)$. In order to do it we will first find values of v in points $(x_i+\frac{h^x_{ij}}{2}, y_j+\frac{h^y_{ij}}{2})$ and $(x_i+\frac{h^x_{ij}}{2}, y_j-\frac{h^y_{ij}}{2})$ (see point E and F respectively on the Fig. 3.2):

$$v\bigg|_{(x_i+\frac{h^x_{ij}}{2},y_j+\frac{h^y_{ij}}{2})} \approx \frac{v_{i+1j}h^x_{ij}+v_{ij}h^x_{i+1j}}{h^x_{ij}+h^x_{i+1j}} \tag{3.19}$$

$$v\bigg|_{(x_i+\frac{h^x_{ij}}{2},y_j-\frac{h^y_{ij}}{2})} \approx \frac{v_{i+1j-1}h^x_{ij}+v_{ij}h^x_{i+1j}}{h^x_{ij-1}+h^x_{i+1j}} \tag{3.20}$$

Then, since the velocity at point $(x_i+\frac{h^x_{ij}}{2}, y_j)$ is located at the middle of the face EF, the velocity v in this point can be calculated by linear interpolation of the values from (3.19) and (3.20):

$$v\bigg|_{(x_i+\frac{h^x_{ij}}{2},y_j)} \approx \frac{v\Big|_{(x_i+\frac{h^x_{ij}}{2},y_j-\frac{h^y_{ij}}{2})}+v\Big|_{(x_i+\frac{h^x_{ij}}{2},y_j+\frac{h^y_{ij}}{2})}}{2} \tag{3.21}$$

The sign of $v\Big|_{(x_i+\frac{h^x_{ij}}{2},y_j)}$ is used for the upwind approximation of $v\frac{\partial u}{\partial y}$:

$$v\frac{\partial u}{\partial y}\bigg|_{(x_i+\frac{h^xij}{2},y_j)} \approx \begin{cases} v\Big|_{(x_i+\frac{h^x_{ij}}{2},y_j)}\dfrac{u_{ij+1}-u_{ij}}{\frac{h^y_{ij}}{2}+\frac{h^y_{ij+1}}{2}} \text{ if } v\Big|_{(x_i+\frac{h^x_{ij}}{2},y_j)}>0, \\ v\Big|_{(x_i+\frac{h^x_{ij}}{2},y_j)}\dfrac{u_{ij}-u_{ij-1}}{\frac{h^y_{ij}}{2}+\frac{h^y_{ij-1}}{2}} \text{ if } v\Big|_{(x_i+\frac{h^x_{ij}}{2},y_j)}<0, \end{cases} \tag{3.22}$$

Special care has to be taken for the integral over Brinkman term $\frac{\nu}{K}u$, because the coefficient $\frac{\nu}{K}$ is a constant on control volumes for scalar variables such as S_{ij}, S_{i+1j}, and therefore it can have different different values for the left sub-volume $U_{ij}^{-1/2} = ABEF$ and for the right sub-volume $U_{ij}^{+1/2} = FECD$ of the control volume U_{ij}:

$$\begin{gathered}\iint\limits_{U_{ij}} \left(\frac{\nu}{K}u\right) dU_{ij} = \left(\iint\limits_{U_{ij}^{-1/2}} \frac{\nu}{K} dU_{ij}^{-1/2} + \iint\limits_{U_{ij}^{+1/2}} \frac{\nu}{K} dU_{ij}^{+1/2} \right) u_{ij} = \left(\frac{\nu_{ij}}{K_{ij}}|U_{ij}^{-1/2}| + \frac{\nu_{i+1j}}{K_{i+1j}}|U_{ij}^{+1/2}|\right) u_{ij} \\ |U_{ij}^{-1/2}| = h^y_{ij}\frac{h^x_{ij}}{2},\ |U_{ij}^{-1/2}| = h^y_{ij}\frac{h^x_{i+1j}}{2}\end{gathered} \tag{3.23}$$

Surface integrals discretization.

Surface integrals are decomposed into four integrals over faces of the control volume U_{ij}:

$$\int\limits_{\partial U_{ij}} \underbrace{(\nu \nabla u)}_{F_1} \cdot n dS = \int\limits_{AB} F_1 \cdot n dS + \int\limits_{BC} F_1 \cdot n dS + \int\limits_{CD} F_1 \cdot n dS + \int\limits_{DA} F_1 \cdot n dS \tag{3.24}$$

Faces AB and CD are located inside the control volumes for pressure (S_{ij}, S_{i+1j}), where the value of the viscosity ν is constant. First, let us consider the integral over CD:

$$\int\limits_{CD} F_1 \cdot n dS = F_1 \cdot n|_{i+1,j} \cdot |CD| = \left(\nu_{i+1j}\frac{u_{i+1j} - u_{ij}}{h^x_{i+1}}\right) h^y_{ij} \tag{3.25}$$

Similar for other integrals:

$$\int\limits_{AB} F_1 \cdot n dS = -\left(\nu_{ij}\frac{u_{ij} - u_{i-1j}}{h^x_i}\right) h^y_{ij} \tag{3.26}$$

$$\int\limits_{BC} F_1 \cdot n dS = \left(\nu_{i+\frac{1}{2}j+\frac{1}{2}}\frac{u_{ij+1} - u_{ij}}{\frac{h^y_{ij+1}}{2} + \frac{h^y_{ij}}{2}}\right)\left(\frac{h^x_{ij}}{2} + \frac{h^x_{i+1j}}{2}\right) \tag{3.27}$$

$$\int\limits_{DA} F_1 \cdot n dS = -\left(\nu_{i+\frac{1}{2}j-\frac{1}{2}}\frac{u_{ij-1} - u_{ij}}{\frac{h^y_{ij-1}}{2} + \frac{h^y_{ij}}{2}}\right)\left(\frac{h^x_{ij}}{2} + \frac{h^x_{i+1j}}{2}\right) \tag{3.28}$$

Pressure correction equation

After explaining the discretization of the momentum equations, we have to describe the discretization of the pressure correction equation (3.7). As before, we integrate the governing equation over control volume associated with the unknown variable (the pressure in this case).

$$\iint\limits_{S_{ij}} \left(\nabla \cdot \left(\left(\frac{\rho}{\tau} + \frac{\rho\nu}{K}\right)^{-1} \nabla \delta p\right)\right) dS_{ij} = \iint\limits_{S_{ij}} \left(\nabla \cdot u^*\right) dS_{ij} \tag{3.29}$$

Then by applying divergence theorem (3.14) we get the following equation:

$$\int\limits_{\partial S_{ij}} \left(\left(\frac{\rho}{\tau} + \frac{\rho\nu}{K}\right)^{-1} \nabla \delta p\right) \cdot n dS = \int\limits_{\partial S_{ij}} u^* \cdot n dS \tag{3.30}$$

As before we decompose the surface integral into four integrals over individual faces of S_{ij}. Since the procedure is very similar to what we have done for momentum equation, let us stop only on one difficult point: how we should deal with the coefficient $\left(\frac{\rho}{\tau} + \frac{\rho\nu}{K}\right)^{-1}$ on the faces between two pressure cells. As example, let us consider the face EF. The density ρ and the permeability K may take different values in points $(x_i, y_j), (x_{i+1}, y_i)$ and when this points belong to different media the jump in the coefficient can be significant. Let us denote $\left(\frac{\rho}{\tau} + \frac{\rho\nu}{K}\right)^{-1}$ as K^B, so that $K^B(x_i, y_j) = K^B_{ij}$, $K^B(x_{i+1}, y_j) = K^B_{i+1j}$ and K^B_{EF} is the value of K^B interpolated on the face EF. To get value of K^B on the face EF we use harmonic interpolation:

$$K^B_{EF} = \frac{h^x_{ij} + h^x_{i+1j}}{\frac{h^x_{ij}}{K^B_{i+1j}} + \frac{h^x_{i+1j}}{K^B_{ij}}} \tag{3.31}$$

Mass sinks and sources.

Let us consider the pressure correction equation (3.7) in case non-zero mass source Q: $\nabla \cdot \left(\left(\frac{\rho}{\tau} + \frac{\rho\nu}{K}\right)^{-1} \nabla \delta p\right) = Q + \nabla \cdot \mathbf{u}^*$. On the discrete level every scalar finite volume cell S_{ij} can act like a source (it adds some amount of liquid to the domain) or like a sink (it removes some amount of liquid from the domain). To satisfy the incompressibility constraint, the sinks and the sources have to be consistent with each other, and to be consistent with the boundary conditions. In a spatial discretization the mass term appears in the following form:

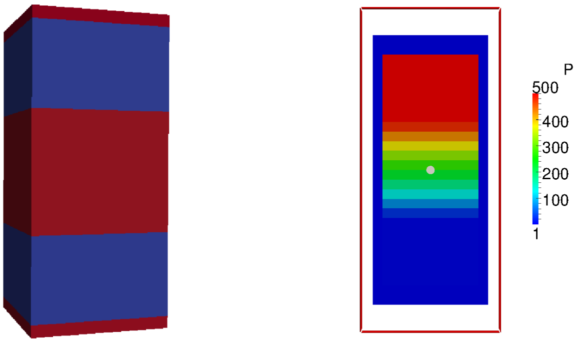


Figure 3.3: Channel filter geometry and pressure drop.

$$\iint\limits_{S_{ij}} Q_{ij} dS_{ij} = Q_{ij}|S_{i}j|, \; |S_{i}j| = h_i^x h_j^y \tag{3.32}$$

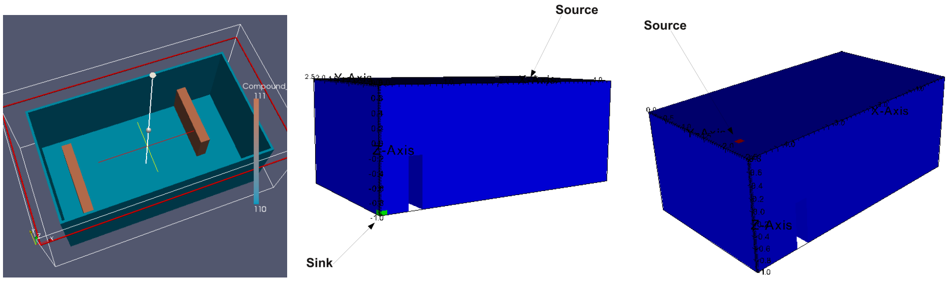
3.1.3 Numerical experiments

Two short examples are presented at the end of this Chapter just in order to demonstrate the capability of the developed algorithm to handle different flow regimes using the same mathematical model. The first example was used also for validation, because analytical solution is available in this case. In the second case, one and the same model, Navier-Stokes-Brinkman equations, (as expected) gives very different solutions for different values of the permeability.

Flow in a channel with a porous block

In this example let us consider a parallelepiped $(x, y, z) : 0 \le x \le 3, 0 \le y \le 3, 0 \le z \le 6$ with a single porous layer $(x, y) : 0 \le x \le 3, 0 \le y \le 3, 2 \le z \le 4$ The following problem parameters are specified: $k_p = 10^{-4}, \nu = 0.01, \rho = 998.21$. In the inlet velocity $w = -1$ is prescribed.

In this example we want to see if our models are able to catch pressure drop effects. Both models (3.6)–(3.8) and (3.9)–(3.12) have shown similar behaviour. Pressure drop in a slice is shown on the left panel of the figure 3.3.



(a) Simulation domain. (b) Sink position. (c) Source position.

Figure 3.4: Flow in room with internal walls. Simulation domain.

Flow in a room with internal walls

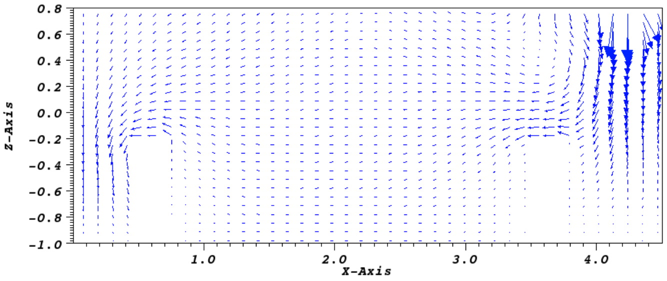
In the course of an accident, leakages may occur in different parts of a containment. These leakages may act as sinks or as sources. With this example we want to demonstrate the influence of the mass sinks and sources terms, combined with the influence of the permeability of the obstacles on the flow regimes.

The computational domain is a parallelepiped room with two internal walls. The room is filled with water, see figure 3.4a for details on the geometry. We've placed a source at the top of the room (see figure 3.4b) and a sink at the bottom of the room (see figure 3.4c). The volume of water generated by the source equals to the volume of water which is withdrawn into the sink, therefore the water level doesn't change, but an intensive water flow may exist, as it is illustrated on figure 3.5a and figure 3.5b.

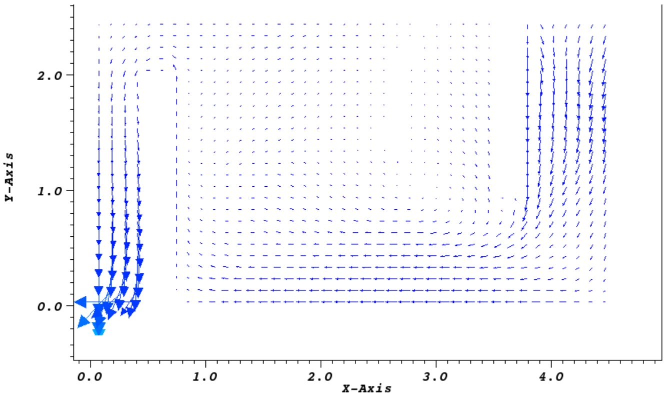
3.2 Alternating direction implicit method for the pressure correction equation

As it was stated in the introduction one of the important requirement for the developed models and software tool is an ability to gain the result fast.

As in the case of the classical Navier-Stokes equations, the solution of the Poisson pressure correction equation can often be a very computationally intensive task. To circumvent this difficulty [37] proposed to use a perturbation of the continuity equation based on a direction-wise factorized operator instead of the classical Laplace operator which allows for the use of a fast tridiagonal direct solver. Let us introduce this idea here.



(a) Velocity in vertical slice.



(b) Velocity in horizontal slice.

Figure 3.5: Velocity vectors in slices.

In [80] was pointed that Chorin's projection method can be interpreted as a pressure stabilization Petrov-Galerkin method, in which the pressure term in the momentum equation is treated explicitly:

$$\begin{cases} \frac{\partial \mathbf{u}_\epsilon}{\partial t} - \nu \Delta \mathbf{u}_\epsilon + \nabla p_\epsilon = f \text{ in } \Omega \times [0,T] \\ -\epsilon \Delta p_\epsilon + \nabla \cdot \mathbf{u}_\epsilon = 0, \text{ in } \Omega \times [0,T] \\ \mathbf{u}_\epsilon|_{\partial\Omega} = 0, \ \partial_n p_\epsilon|_{\partial\Omega} = 0 \text{ in } [0,T], \text{ and} \\ \mathbf{u}_\epsilon|_{t=0} = \mathbf{u}_0, \ p_\epsilon|_{t=0} = p_0 \text{ in } \Omega \end{cases} \tag{3.33}$$

In [42] it is stated that Laplace operator Δ in the second equation of the system 3.33 can be replaced by any operator A which possess specific property: A and its domain $D(A)$ are such that the bilinear form $a(p,q) := \int\limits_\Omega qAp\mathrm{d}\mathbf{x}$ satisfies conditions:

$$a \text{ is symmetric}, \qquad \text{and} \qquad \|\nabla q\|^2_{\mathbf{L}^2} \leq a(q,q), \quad \forall q \in D(A). \tag{3.34}$$

In [37] it is proven that if A is chosen in the factorized form

$$A := (I - \partial_{xx})(I - \partial_{yy})(I - \partial_{zz}),$$

then its bilinear form satisfies the properties (3.34).

Semi-implicit time discretization of the Navier-Stokes equations with direction-wise factorized operator on domain Ω and time interval $[0,T]$ reads as follows:

$$\begin{aligned} &\textit{Pressure predictor} \\ &\qquad p^{*,n+1} = p^n + \phi^n \\ &\textit{Velocity update} \\ &\qquad \frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\tau} + (\mathbf{u}^n \cdot \nabla)\mathbf{u}^n - \Delta \mathbf{u}^{n+1} + \nabla p^{*,n+1} = 0 \\ &\textit{Pressure corrector:} \\ &\qquad A\phi^{n+1} = -\frac{\nabla \cdot \mathbf{u}^{n+1}}{\Delta t} \\ &\textit{Pressure update:} \\ &\qquad p^{n+1} = p^n + \phi^{n+1} \end{aligned} \tag{3.35}$$

As shown in [42] this scheme is unconditionally stable for the unsteady Stokes problem and is first order accurate in time if the domain has a simple rectangular/parallelepiped shape. In our situation, the shape of the domain is much more complex and therefore the operator A supplied with zero Neumann conditions may not satisfy the conditions required to maintain the stability and accuracy of the algorithm (see [42]). In order to explain the difficulties arising in the pressure problem in case of complex geometries, let us assume for a moment that the problem is two-dimensional. As discussed in [42], if Ω has a simple rectangular shape, we need to use homogeneous Neumann boundary condition in each direction when we invert the operator A. This is because we need that the bilinear form associated with it, (Ap, q) for some p, q in the domain of A, be symmetric and coercive. Then it is easy to show that the solution of pressure corrector step of the scheme (3.35) has a gradient controlled by the L^2 norm of the velocity at the new time level because $||\nabla q||^2 \leq (Aq, q), \quad \forall q \in D(A)$. This in turn guarantees the stability and accuracy of the scheme. To understand the difficulty arising in the case of a complex geometry let us first consider the bilinear form in $\Omega = [0,1] \times [0,1]$ and integrate by parts the second order derivatives as follows

$$\begin{aligned}
\int_\Omega qAp\mathbf{dx} &= \int_\Omega q(1-\partial_{xx})\psi\mathbf{dx} = \int_\Omega (q\psi + \partial_x q\partial_x\psi)\mathbf{dx} \\
&= \int_\Omega (q(1-\partial_{yy})\phi + \partial_x q\partial_x(1-\partial_{yy})\phi)\mathbf{dx} \\
&= \int_\Omega (q\phi + \partial_y q\partial_y\phi + \partial_x q\partial_x\phi - \partial_x q\partial_{xyy}\phi)\mathbf{dx} = \\
&= \int_\Omega (q\phi + \partial_y q\partial_y\phi + \partial_x q\partial_x\phi + \partial_{xy} q\partial_{xy}\phi)\mathbf{dx} \\
&- \int_{\{y=1\}} \partial_x q(x,1)\partial_{xy}\phi(x,1)\mathrm{d}x + \int_{\{y=0\}} \partial_x q(x,0)\partial_{xy}\phi(x,0)\mathrm{d}x.
\end{aligned} \tag{3.36}$$

The boundary terms are zero if the following boundary conditions are imposed: $\partial_x\psi(1,y) = 0$ and $\partial_x\psi(0,y) = 0$ for all $y \in (0,1)$, $\partial_y\phi(x,1) = 0$ and $\partial_y\phi(x,0) = 0$ for all $x \in (0,1)$. Note that the last two conditions imply that $\partial_{xy}\phi(x,1) = \partial_x(\partial_y\phi(x,1)) = 0$, and $\partial_{xy}\phi(x,0) = \partial_x(\partial_y\phi(x,0)) = 0$ for all $x \in (0,1)$ and so the last boundary terms in (3.36) vanish. However, if the boundary has a more complex shape, the mixed derivative $\partial_{xy}\phi_{|\partial\Omega}$ does not necessarily vanish and the non-zero boundary terms of the type $\partial_x q\partial_{xy}\phi$ imply that the bilinear form looses its positive definiteness. This in turn destabilizes the algorithm. However, we can make the following observation. Suppose that we cover the domain Ω with a finite number of quadrilaterals in 2D or parallelepipeds in 3D, whose sides are orthogonal to one of the coordinate axes. Then if we use the operator A with zero Neumann boundary conditions on each piece of the boundary for computation of the pressure on the so extended domain, it is clear that we can integrate by parts as in (3.36) on each of the rectangles/parallelepipeds. Moreover, all the boundary terms will vanish since on the external pieces of boundary we enforce zero Neumann conditions, and on all internal interfaces between the subdomains the continuity of the fluxes implies that all the boundary terms on those interfaces would cancel each other as well.

In other worlds, we need to extend our problem to a rectangular/parallelepiped domain containing Ω entirely. Note, that this is exactly what the fictitious domain method is designed for. An attentive reader may remembered that in the introduction (see 1) we discussed two possibilities of how complex-shaped domains and complex-shaped obstacles can be treated: immersed boundary method (see 1.2.1) and fictitious domain method (see 1.2.2). Now it is clear that if we want to use fast direction-wise factorized operator for the pressure correction step as an alternative to the slow Poisson pressure correction equation, than we better use fictitious domain method instead of one of the variants of the immersed boundary methods.

3.2.1 Numerical algorithm

Here we are discussing adjustment of the algorithms proposed in [42] to the penalized Navier-Stokes equations.

Time discretization

In [37] it was shown that if the domain of the problem has a simple shape, it is convenient to perturb the continuity equation to the form $\prod_{i=1}^{d} (I - \partial_{x_i x_i})\, \phi = -\nabla{\cdot}\mathbf{u}/\Delta t$ with ϕ being either the pressure itself (for first order schemes) or its time increment (for higher order schemes). Therefore, it would be also convenient to apply the same direction-splitting procedure to the momentum equation. However, since the permeability is space-dependent, the direction splitting of the momentum equation, in case of an implicit treatment of the Brinkman term $\nu\mathbf{u}/k$, is not very straightforward. To understand the problem, let us consider the Douglas splitting for the first equation in (3.10) (see [45]) in the following factorized form

$$\prod_{i=1}^{d}\left(b(\mathbf{x})I - \frac{\nu\Delta t}{2b(\mathbf{x})}\partial_{x_ix_i}\right)\frac{\mathbf{u}^{n+1}-\mathbf{u}^{n}}{\Delta t} - \nu\Delta\mathbf{u}^{n} + \frac{\nu}{k}\mathbf{u}^{n} + \nabla p^{*,n+1/2} = \mathbf{f}^{n+1/2}, \tag{3.37}$$

with $b(\mathbf{x}) = (1 + \nu\Delta t/(2k))^{1/d}$ and I being the identity operator. It is clear that since $b(\mathbf{x})^{-1}\partial_{x_1x_1}$ and $b(\mathbf{x})^{-1}\partial_{x_2x_2}$ do not commute in general, the product operators of the type $b(\mathbf{x})^{-1}\partial_{x_1x_1}b(\mathbf{x})^{-1}\partial_{x_2x_2}$ are not necessarily positive and self-adjoint which affects the stability of the scheme. As proven in [83], section 2.2.3, the Douglas scheme is unconditionally stable in 2D but the stability in 3D cannot be guaranteed. Therefore, here we do not consider the possibility to split direction-wise the momentum equation. As a result, the full 2D/3D momentum problem is solved by means of an iterative procedure. The reader is referred to [2] for some possibilities for a direction splitting of the momentum equation in the non-commutative case, however, no theory for the case of Navier-Stokes-Brinkman flows has been developed in the case of non-commutative splitting of the momentum equation.

Therefore in order to obtain full algorithm we need to change pressure correction step (3.11) in the scheme (3.9–3.12) in a following way:

Pressure-corrector:

$$A\phi^{n+1/2} = -\tfrac{1}{\Delta t}\nabla\cdot\mathbf{u}^{n+1}, \tag{3.38}$$

where $A = \prod\limits_{i=1}^{d}(I - \partial_{x_ix_i})$ together with homogeneous Neumann boundary conditions on $\partial\Omega$. This operator is factorized alongside the spatial directions and is therefore much easier to invert than the usual for projection methods Laplace operator. The solution to problem (3.11) can in fact be computed as the following sequence of one dimensional problems:

$$\begin{aligned}
&\psi - \partial_{xx}\psi = -\frac{\nabla\cdot\mathbf{u}^{n+1}}{\Delta t},\\
&\eta - \partial_{yy}\eta = \psi,\\
&\phi - \partial_{zz}\phi = \eta.
\end{aligned}$$

The stability of the scheme is guaranteed by the following theorem.

Theorem 3.2.1. *Assume that the solution to* (3.10) *is smooth enough. There exist* c*, independent of* τ*, such that for all* T *and all* $\chi \in [0,1]$ *the solution to* (3.9), (3.10), (3.38), (3.12) *satisfies the following stability estimate:*

$$\begin{aligned}&\|\boldsymbol{u}\|^2_{\ell^\infty(0,T;\boldsymbol{L}^2)} + \frac{\tau}{2}\|\nu^{1/2}\nabla\boldsymbol{u}\|^2_{\ell^\infty(\tau,T;\boldsymbol{L}^2)} + \tau^2\|p\|_{\ell^\infty(\tau,T-\tau,D(A))}+\\ &\frac{\tau}{2}\|\left(\frac{\nu}{k}\right)^{1/2}\boldsymbol{u}\|^2_{\ell^\infty(\tau,T;\boldsymbol{L}^2)} \leq c(\|\boldsymbol{u}^0\|^2_{\boldsymbol{L}^2} + \tau^2\|p^{-1/2}\|^2_A+\\ &\frac{\tau}{2}\|\nu^{1/2}\nabla\boldsymbol{u}^0\|^2_{\boldsymbol{L}^2} + \frac{\tau}{2}\|\left(\frac{\nu}{k}\right)^{1/2}\boldsymbol{u}^0\|^2_{\boldsymbol{L}^2} + \nu^{-1}\|\mathbf{f}\|_{\ell^2(\tau,T;\boldsymbol{H}^{-1})}).\end{aligned} \tag{3.39}$$

Proof. The proof follows along the same lines as the proof of stability of the direction splitting scheme in the case of Navier-Stokes equations given in [37]. Nevertheless, there are certain differences and we provide a sketch of the proof for the case $\chi = 0$. This stability estimate can then be used to study the accuracy of the scheme similarly to the case of the rotational pressure correction method studied in [39]. We first multiply (3.10) by $2\tau\mathbf{u}^{n+1}$, integrate over Ω, and use the identity $2(a-b,a) = \|a\|^2 + \|a-b\|^2 - \|b\|^2$ and the Young's inequality to obtain

$$\begin{aligned}&\|\mathbf{u}^{n+1}\|^2_{\mathbf{L}^2} + \|\mathbf{u}^{n+1} - \mathbf{u}^n\|^2_{\mathbf{L}^2} + \frac{\tau}{2}\|\nu^{1/2}\nabla\mathbf{u}^{n+1}\|^2_{\mathbf{L}^2}\\ &+ 2\tau(\nabla p^{*,n+1}, \mathbf{u}^{n+1}) + \frac{\tau}{2}\|\left(\frac{\nu}{k}\right)^{1/2}\mathbf{u}^{n+1}\|^2_{\mathbf{L}^2} \leq\\ &\|\mathbf{u}^n\|^2_{\mathbf{L}^2} + \frac{\tau}{2}\|\nu^{1/2}\nabla\mathbf{u}^n\|^2_{\mathbf{L}^2} + \frac{\tau}{2}\|\left(\frac{\nu}{k}\right)^{1/2}\mathbf{u}^n\|^2_{\mathbf{L}^2} + 2\Delta t\int\limits_\Omega \mathbf{f}^{n+1}\cdot\mathbf{u}^{n+1}d\Omega\end{aligned} \tag{3.40}$$

Now we use the properties (3.34) of the operator A to deduce that the pressure correction $(p^{n+1} - p^n) \in D(A)$ solves the following problem for $n \geq 0$:

$$a(p^{n+1} - p^n, q) = -\tau^{-1}(\nabla\cdot\mathbf{u}^{n+1}, q), \qquad \forall q \in D(A). \tag{3.41}$$

Testing it with $2\tau^2 p^{*,n+1} := 2\tau^2(2p^n - p^{n-1})$ (in the case $\chi = 0$) and using the symmetry and coercivity of $a(\cdot,\cdot)$ we obtain that

$$\begin{aligned}-2\tau(\nabla\cdot\mathbf{u}^{n+1}, p^{*,n+1}) = \tau^2\big(\|p^{n+1}\|^2_A -\\ \|p^n\|^2_A + \|\delta p^n\|^2_A - \|\delta^2 p^{n+1}\|^2_A\big).\end{aligned} \tag{3.42}$$

where $\delta p^n = p^n - p^{n-1}$ is the usual difference operator. The control on $\|\delta^2 p^{n+1}\|^2_A$ is obtained subtracting (3.41) at time t^n from (3.41) at time t^{n+1} and by testing the result with $\tau\delta^2 p^{n+1}$,

$$\begin{aligned}\tau\|\delta^2 p^{n+1}\|_A^2 &= -(\nabla\cdot(\mathbf{u}^{n+1}-\mathbf{u}^n),\delta^2 p^{n+1}) = (\mathbf{u}^{n+1}-\mathbf{u}^n,\nabla\delta^2 p^{n+1})\\ &\le \|\mathbf{u}^{n+1}-\mathbf{u}^n\|_{\mathbf{L}^2}\|\nabla\delta^2 p^{n+1}\|_{\mathbf{L}^2}.\end{aligned} \tag{3.43}$$

Then the coercivity property of the bilinear form a implies that

$$\tau\|\nabla\delta^2 p^{n+1}\|_{\mathbf{L}^2}\|\delta^2 p^{n+1}\|_A \le \|\mathbf{u}^{n+1}-\mathbf{u}^n\|_{\mathbf{L}^2}\|\nabla\delta^2 p^{n+1}\|_{\mathbf{L}^2},$$

which yields the inequality $\tau^2\|\delta^2 p^{n+1}\|_A^2 \le \|\mathbf{u}^{n+1}-\mathbf{u}^n\|_{\mathbf{L}^2}^2$. This bound together with (3.42) gives the following bound on the pressure gradient term in (3.40)

$$\begin{aligned}\tau^2\left(\|p^{n+1}\|_A^2 + \|\delta p^{n-1}\|_A^2 - \|p^{n-1}\|_A^2\right) \le\\ -2\tau(\nabla\cdot\mathbf{u}^{n+1},p^{*,n+1}) + \|\mathbf{u}^{n+1}-\mathbf{u}^n\|_{\mathbf{L}^2}^2\end{aligned} \tag{3.44}$$

which after summing with (3.40) gives

$$\begin{aligned}&\|\mathbf{u}^{n+1}\|_{\mathbf{L}^2}^2 + \tau^2\|p^{n+1}\|_A^2 + \frac{\tau}{2}(\|\nu^{1/2}\nabla\mathbf{u}^{n+1}\|_{\mathbf{L}^2}^2 + \frac{\tau}{2}\|\left(\frac{\nu}{k}\right)^{1/2}\mathbf{u}^{n+1}\|_{\mathbf{L}^2}^2\\ &\quad\le \|\mathbf{u}^n\|_{\mathbf{L}^2}^2 + \tau^2\|p^{n-1}\|_A^2 + \frac{\tau}{2}\|\nu^{1/2}\nabla\mathbf{u}^n\|_{\mathbf{L}^2}^2 + \frac{\tau}{2}\|\left(\frac{\nu}{k}\right)^{1/2}\mathbf{u}^n\|_{\mathbf{L}^2}^2 + \nu^{-1}\|\mathbf{f}\|_{\mathbf{H}^{-1}}.\end{aligned} \tag{3.45}$$

The final estimate is obtained by summing (3.45) over the time levels from 0 to $N-1 = T/\tau - 1$. ☐

This stability estimate can be used to derive a first order error estimate on the velocity and the pressure in the L^2 norm similarly to theorem 3.1 of [40]. However, the main problem that prevents the derivation of an optimal second order estimate on the velocity in the fully split case of the Stokes equations considered in [40], remains in the present case too. In essence, it is due to the fact that the norm generated by the factorized operator A is not equivalent to the H^1 norm which in turn does to allow to apply the usual duality argument for obtaining an optimal estimate. Thus in the present case, as in the case of the Stokes problem with split momentum equation considered in [40], using the same arguments, we can currently derive only estimates for velocity error. In L^2 norm it is of order $\Delta t^{3/2}$ in the case of $\chi > 0$ (see also theorem 4.2 of [40]). The improvement in the present case is that this result is valid irrespectively of the spatial dimension because the momentum equation is not discretized with a direction splitting scheme. Overall, the extension of all results for the Stokes problem in simple domains obtained in [40] to the Brinkman case is quite straightforward, and the purpose of the present paper is mainly to show that the factorized perturbation of the incompressibility constraint (3.38) works in the Brinkman case independently of the domain shape i.e. the pressure can be extended outside of the fluid domain with a penalty approach and this would not hamper the stability of the splitting.

3.2.2 Computational cost of the direction splitting operator vs. Laplace operator in the pressure correction equation

For simplicity, let us consider a cubic domain covered with a grid containing $n \times n \times n$ nodes. Under the reasonable assumption that the cost for assembling the matrices is small compared to the cost of solving the systems of linear equations, let us evaluate the number of operations needed for computing the solution for the pressure *per time step*.

Direction splitting operator. In 3D, the Thomas algorithm for tri-diagonal matrices has to be employed $3n^2$ times (i.e., n^2 times in each direction). Each Thomas algorithm requires $5n$ multiplications/divisions and $3n$ summations/subtractions. So, the total of $15n^3 + 9n^3$ operations are required.

Laplace operator. Suppose now that we use an iterative method to solve the pressure Poisson equation[1]. Each iteration requires at least one matrix-vector multiplication, which for a seven-diagonal matrix will require $7n^3$ multiplications and $6n^3$ summations, thereby $7n^3 + 6n^3$ is the total number of operation for one matrix-vector multiplication.

These results clearly indicate that even if the iterative method converges with only a couple of iterations, the solution of pressure equation with a direction splitting operator and a direct tri-diagonal solver requires less operations than the iterative solution of the pressure Poisson equation.

---

[1] Note that the Poisson equation can be solved using fast Fourier transform (FFT). The parallel performance of the FFT algorithm, however, is much worse than that of a Thomas-based Schur complement approach (see [37])

3.2.3 Numerical experiments

Preliminaries

The performance of the two approximations to the incompressibility constraint discussed here: the pressure Poisson equation, and the directionally factorized perturbation (3.38), is compared in a domains involving fluid and porous regions. The domain of the first problem is a channel with sudden contraction, containing a porous subdomain with a given permeability. If the permeability is very large, the Darcy term in the Brinkman equations tends to zero, and therefore the flow in the porous subdomain is unrestricted similarly to the rest of the fluid domain. For very small values of the permeability, the Darcy term dominates in the porous subdomain, and may completely prevent the flow through it. In fact, in this case the Stokes-Brinkman equations become a Fictitious Domain (penalty) formulation for the Stokes equations with no-slip boundary conditions on the boundary of the subdomain in which an extremely low porosity is defined. The domain of the second problem is a channel with two porous subdomains. In this case time dependent boundary conditions are applied.

The main goal of these simulations is to compare the results for the velocity and pressure computed with the classical Poisson equation for the pressure increment (classical incremental projection scheme in a rotational form; see [36]), and the factorized operator A defined above. Therefore, we define $\mathbf{u}_\Delta, p_\Delta$ to be the velocity and pressure calculated using the classical projection scheme, and $\mathbf{u}_A, p_A$ – the velocity and pressure calculated using the scheme with the operator A in the pressure correction step.

In all test cases presented below the viscosity is set to $\nu = 10^{-6} m^2 \cdot s^{-1}$ and the parameter χ is set to one. The momentum equation of both schemes, and the pressure-Poisson equation in the classical projection scheme are solved by a generalized minimal residual method with ILU preconditioner (see [84] for implementation details). All simulations were performed on a machine with a dual core Intel Xeon 5148LV with 8 GB RAM.

Flow in a channel with a sudden contraction and a porous obstacle

As a first test case, consider the flow in a channel with a sudden contraction and a porous obstacle (see figure 3.6). No-slip boundary conditions are prescribed on the entire boundary except for the inlet AB and the outlet CD where a parabolic profile for the velocity is prescribed:

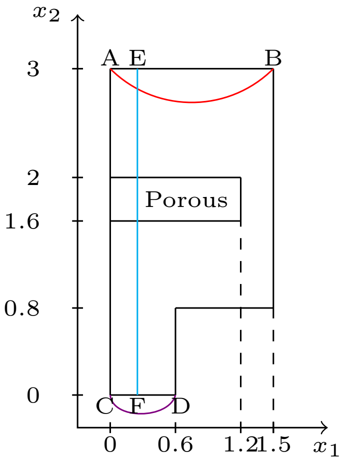


Figure 3.6: Sketch of a channel with a sudden contraction and a porous obstacle.

$$\begin{aligned} u_{AB} = &\quad \alpha_1 x_1(x_1 - 1.5), \quad 0 \le x_1 \le 1.5, \quad x_2 = 3 \\ u_{CD} = &\quad \alpha_2 x_1(x_1 - 0.6), \quad 0 \le x_1 \le 0.6, \quad x_2 = 0 \end{aligned}$$

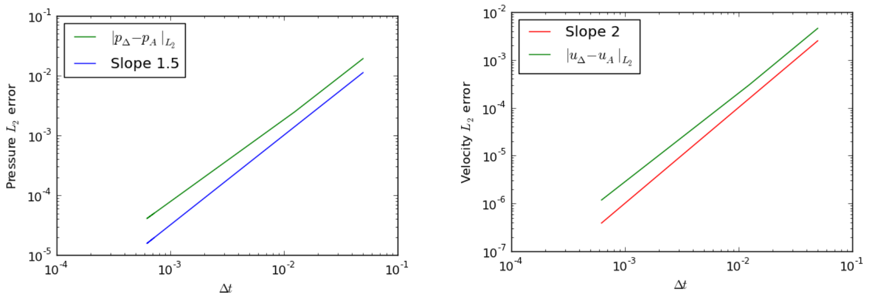
The coefficients α_1 and α_2 are specified so that the flow rates at both ends of the channel are equal:

$$\int_0^{1.5} \alpha_1 x(x - 1.5)dx = \int_0^{0.6} \alpha_2 x(x - 0.6)dx = 0.015.$$

The usual zero Neumann boundary condition on the entire boundary is imposed on the pressure correction.

In figure 3.7 we show the L^2 norm of the difference of the two velocities and the pressures, $\|u_\Delta - u_A\|_{L_2}$ and $\|p_\Delta - p_A\|_{L_2}$, as a function of Δt. Clearly, the rate of decrease of both errors is similar to the theoretical estimates for the convergence error of the classical incremental projection scheme in a rotational form (see [39]): second order for the velocity and order $3/2$ for the pressure error in the L^2 norm. At the same time, the simulations with the directional splitting approach (using the factorized operator A) are significantly faster (see table 3.1).

To compare CPU time usage, we've taken into account that the computational time for the iterative solvers depends on the number of iterations, and the latter depends on the prescribed tolerance, as well as on the used time step (this information is summarized in table 3.1).



(a) Pressure L^2 error: $\|p_\Delta - p_A\|_{L_2}$ (b) Velocity L^2 error: $\|u_\Delta - u_A\|_{L_2}$

Figure 3.7: Pressure and velocity error; test case of section 3.2.3 ($h = 0.005$).

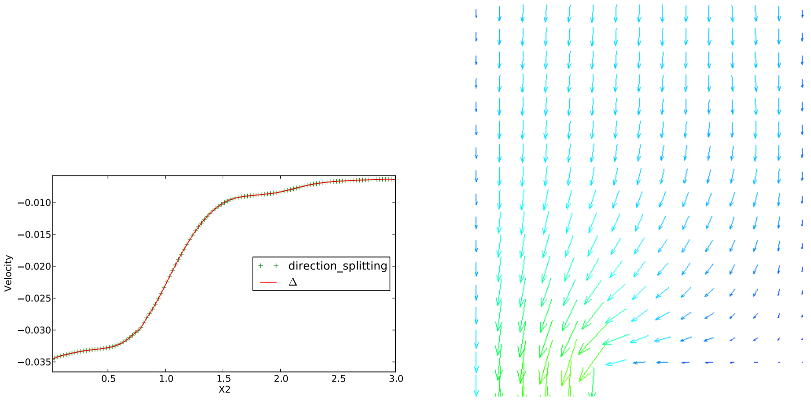
In the left panel of figure 3.8 we display the horizontal profiles of the velocity along the vertical segment EF (see figure 3.6) at various permeabilities. The corresponding velocity vectors are displayed in the right panel of figure 3.8.

Table 3.1: CPU time comparison; test case of section 3.2.3

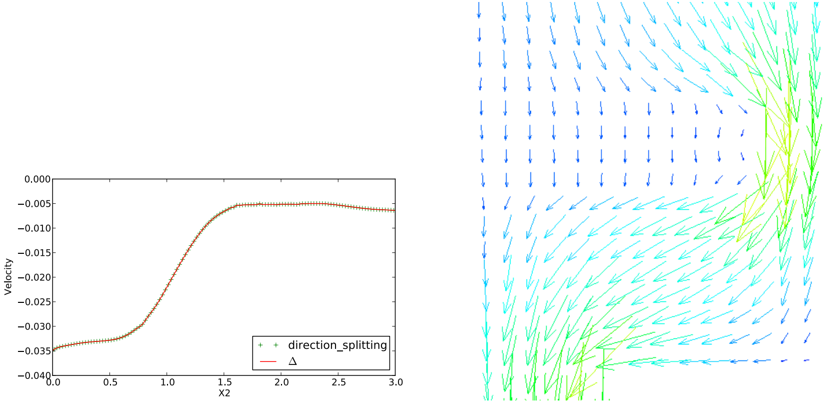
h/N	Δt	Tolerance for iterative solver	Time for Laplace	Time for DS operator	Speed up
0.01/37800	1e-5	1e-8	18.18	2.35	7.7
0.01/37800	1e-5	1e-12	34.12	2.35	14.52
0.01/37800	1e-7	1e-8	22.14	2.35	9.42
0.01/37800	1e-7	1e-12	28.89	2.35	12.3
0.005/151200	1e-5	1e-8	86.12	6.57	13.1
0.005/151200	1e-5	1e-12	97.43	6.57	14.8
0.005/151200	1e-7	1e-8	78.16	6.57	11.8
0.005/151200	1e-7	1e-12	84.25	6.57	12.8

Flow in a channel with two porous obstacles and time dependent boundary conditions

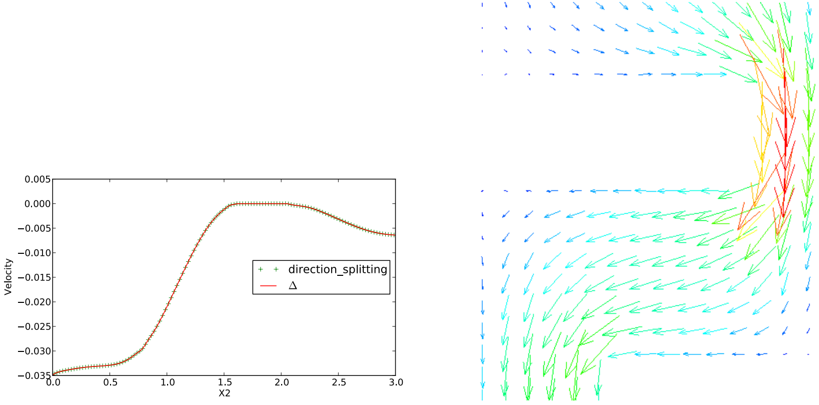
To further illustrate the properties of the factorized scheme, consider the flow in a vertical channel with two porous blocks (see figure 3.9). No-slip boundary conditions are prescribed on the solid walls AC and BD, while at the inlet AB and the outlet CD time a dependent



High permeability ($k = 10^{-4}$).



Moderate permeability ($k = 10^{-6}$).



Extremely low permeability ($k = 10^{-8}$).

Figure 3.8: Velocity profile in the segment EF and the corresponding velocity field for different values of permeability; test case of section 3.2.3 ($h = 0.025, \Delta t = 0.01$).

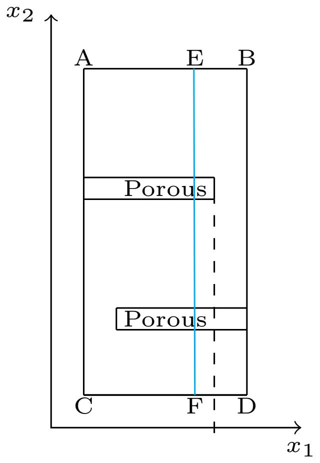
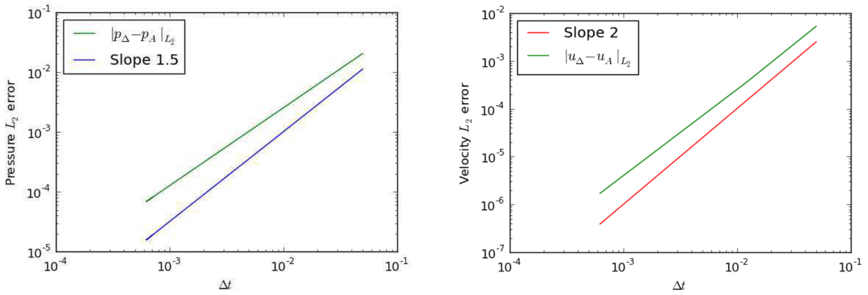


Figure 3.9: Sketch of the channel with two porous obstacles.

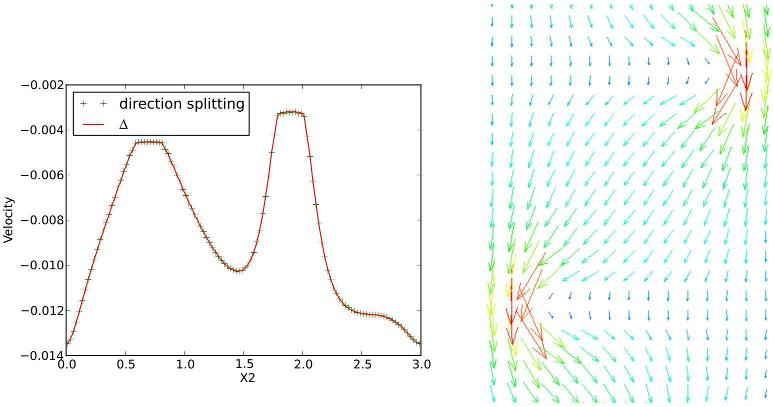
profile for the velocity is prescribed: $-(\pi + \sin(5t))$. The domain is given by: $0 \leq x_1 \leq 1.5,\ 0 \leq x_2 \leq 1$.

In figure 3.10 we show the norm of the difference of the velocities and the pressures, $\|u_\Delta - u_A\|_{L_2}$ and $\|p_\Delta - p_A\|_{L_2}$, as a function of Δt, and in figure 3.11, the vertical profiles of the velocity along the vertical segment EF, as well as the velocity vectors in the entire domain. As in the previous test case, the velocity and pressure difference has a very similar convergence rate to the corresponding convergence errors of the classical incremental projection scheme in a rotational form.

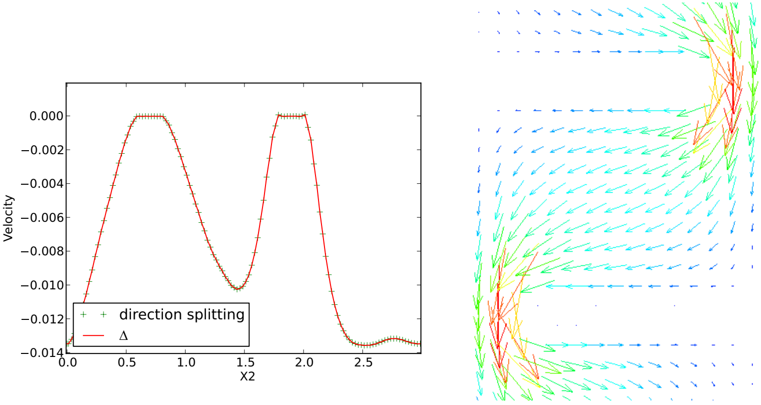


(a) Pressure L^2 error: $\|p_\Delta - p_A\|_{L_2}$ (b) Velocity L^2 error: $\|\mathbf{u_\Delta} - \mathbf{u_A}\|_{\mathbf{L_2}}$

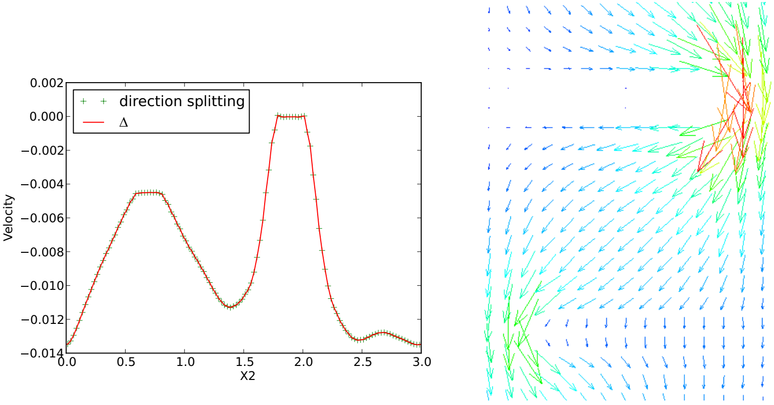
Figure 3.10: Pressure and velocity error; test case of section 3.2.3 ($h = 0.005$).



Moderate permeability in both porous areas ($k_{upper} = k_{lower} = 10^{-6}$).



Low permeability in both porous areas ($k_{upper} = k_{lower} = 10^{-8}$).



Low permeability in the upper area ($k_{upper} = 10^{-8}$), and moderate permeability in the lower area ($k_{lower} = 10^{-6}$)

Figure 3.11: Velocity in the slice and corresponding velocity vectors for different values of permeability; test case of section 3.2.3 ($h = 0.025, \Delta t = 0.01$)

Table 3.2 shows the CPU time comparison for the two schemes, confirming again that the simulations with the direction splitting approach are significantly faster .

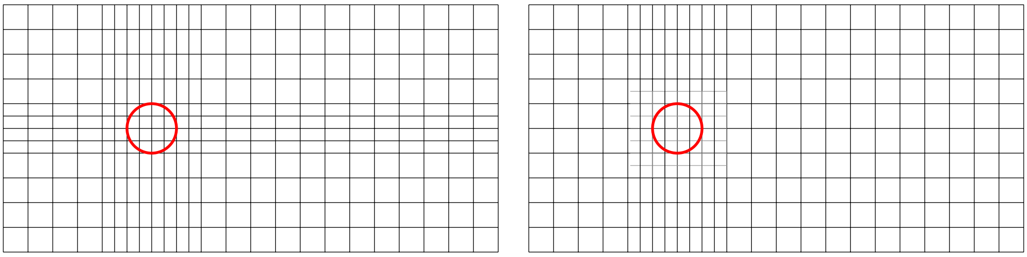
Table 3.2: CPU time comparison; test case of section 3.2.3.

h/N	Δt	Tolerance for iterative solver	Time for Laplace	Time for DS operator	Speed up
0.01/45000	1e-5	1e-8	20.24	2.4	8.61
0.01/45000	1e-5	1e-12	38.12	2.4	16.2
0.01/45000	1e-7	1e-8	27.14	2.4	11.54
0.01/45000	1e-7	1e-12	33.89	2.4	14.42
0.005/180000	1e-5	1e-8	88.15	6.8	12.9
0.005/180000	1e-5	1e-12	99.4	6.8	14.6
0.005/180000	1e-7	1e-8	79.25	6.8	11.6
0.005/180000	1e-7	1e-12	85.13	6.8	12.5

3.2.4 Summary

The results presented in this section demonstrate that the direction-factorized perturbation of the incompressibility constraint of the Navier-Stokes equations proposed in [37] can be applied to the Navier-Stokes-Brinkman equations yielding an unconditionally stable scheme. We do not consider here the possibility for a direction splitting of the momentum equations since it would generally yield non-commutative one-dimensional operators which significantly complicates the stability estimate. Although some possibilities for such splitting of the momentum equation in the non-commutative case are discussed in [2] the stability of the overall algorithm in the Navier-Stokes or Navier-Stokes-Brinkman case is still an open problem.

The presented numerical results clearly demonstrate on two test cases involving fluid and porous areas, that the difference between the velocities and pressures computed with the current direction-slitting approach, and the classical incremental projection scheme in



(a) Resolving a cylinder by changing a mesh size.

(b) Resolving a cylinder by partial grid refinement.

Figure 3.12: Some approaches for resolving a region of interest in the simulation domain.

a rotational form decrease as $O(\Delta t^2)$ and $O(\Delta t^{3/2})$ respectively. In other words, the convergence rates in time of the two algorithms are close to the theoretical estimates for the incremental projection scheme in a rotational form.

3.3 Alternating direction implicit method for a domain with partially matching grid

In modeling of physical processes, especially when we are dealing with NPP safety evaluation, it is often the case when some parts of simulation domain are more interesting than others. Different approaches can be used in order to resolve a more interesting part of the geometry. Let us consider flow around circular cylinder (the setup for this problem was introduced in section 1.2.3). Assuming we are interested in resolving in details an area near cylinder. One simple way how we can do it, it is by decreasing a spatial step in all directions near cylinder like it is shown in the figure 3.12a. This approach has obvious disadvantage: not only area near cylinder is resolved, but also some extra area, especially in this example when channel can be sufficiently long overhead may be noticeable. Another approach is presented in the figure 3.12b: only part near cylinder which we are interested can be resolved. In this section we present a method which can be used in order to deal with a domain from figure 3.12b. For example, in [21] was proposed local refinement approach for structured grids, here we also want to present similar approach but in our case we also want to take some benefits from alternating direction implicit technique.

3.3.1 Solution of a parabolic equation on a domain with partially matching grid

Solution algorithm for a 2D heat equation with Dirichlet boundary conditions by the Peaceman-Rachford method

In order to explain an approach which can be used in order to treat non-matching grid let us use some simplified problem. Let us take two dimensional heat equation on a domain $\Omega = [0, X] \times [0, Y]$ and use finite difference method in order to discretize it:

$$\frac{\partial \phi}{\partial t} = \frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} + f(x, y, t) \text{ in } \Omega \times (0, T] \tag{3.46}$$

With following boundary and initial condition:

$$\begin{aligned} &\phi(x, y, 0) = g(x, y, 0) \text{ in } \Omega \\ &\phi(x, 0, t) = g(0, y, t) = g_{0y}\,, \quad \phi(X, y, t) = g(X, y, t) = g_{Xy} \text{ in } [0, Y] \times (0, T] \\ &\phi(0, y, t) = g(x, 0, t) = g_{x0}\,, \quad \phi(M, y, t) = g(x, Y, t) = g_{xY} \text{ in } [0, X] \times (0, T] \end{aligned} \tag{3.47}$$

Let us introduce alternating direction implicit numerical solution proposed by Peaceman-Rachford. Allowing that we have a uniform for each direction grid on domain $\Omega \times [0, T]$, which is given by the step h_x in x-th coordinates and by the step h_y in the y-th coordinates and also by the temporal step k:

$$\begin{aligned} x_i &= i * h_x,\ i = 0, 1, \ldots, n_x \\ y_j &= j * h_y,\ j = 0, 1, \ldots, n_y \\ t_n &= n * k,\ n = 0, 1, \ldots, t \end{aligned} \tag{3.48}$$

By $\phi^n_{i,j} \approx \phi(x_i, y_j, t_n) = \phi(ih_x, jh_y, nk)$ we denote the value of the numerical solution at a grid point (x_i, y_j). Central difference operators of the second order have a following form: $\delta^2_x \phi_{i,j} = \phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j}$ and $\delta^2_y \phi_{i,j} = \phi_{i,j+1} - 2\phi_{i,j} + \phi_{i,j-1}$.

Five point finite difference Crank-Nicolson [17] scheme method which is unconditionally stable for problem (3.46-3.47) reads as follows:

$$\left(1 - (\frac{r_x}{2}\delta^2_x + \frac{r_y}{2}\delta^2_y)\right)\phi^{n+1} = \left(1 + (\frac{r_x}{2}\delta^2_x + \frac{r_y}{2}\delta^2_y)\right)\phi^n + (\frac{r_x}{2} + \frac{r_y}{2}) f^{n+1/2} + \mathcal{O}(k^3 + kh^2), \tag{3.49}$$

where

$$f^{n+1/2} = \frac{1}{2}(f^n + f^{n+1}), r_x = \frac{k}{h_x^2}, r_y = \frac{k}{h_y^2}, \tag{3.50}$$

To get the alternating direction implicit Peaceman-Rachford [68] method additional profile $\phi^{n+1/2}$ has to be introduced and the equation (3.49) should be split:

$$\begin{aligned} (I - B_1)\phi^{n+1/2} &= (1 + B_2)\phi^n + \frac{r_x}{2} f^{n+1/2} \\ (I - B_2)\phi^{n+1} &= (1 + B_1)\phi^{n+1/2} + \frac{r_y}{2} f^{n+1/2} \end{aligned} \tag{3.51}$$

where $B_1\phi = \frac{r_x}{2}\delta_x^2\phi$, $B_2\phi = \frac{r_y}{2}\delta_y^2$.

Now let us formulate solution algorithm for a 2D heat equation with Dirichlet boundary conditions by the Peaceman-Rachford method. Suppose that the solution ϕ^n has been computed at the n^{th} time level. The following steps should be performed on each time step.

Step 1 (x-sweep).

For each $l = 1, \cdots, L-1$ solve the tridiagonal system

$$(I - B_{1h})\Phi_l^{n+1/2} = (1 + B_{2h})\Phi_l^n + \frac{r}{2} F_l^{n+1/2}, \tag{3.52}$$

where

$$\begin{aligned} \Phi_l^{n+1/2} &= (\phi_{1,l}^{n+1/2}, \phi_{2,l}^{n+1/2}, \cdots, \phi_{M-1,l}^{n+1/2}, \phi_{M-1,l}^{n+1/2})^T \\ \Phi_l^n &= (\phi_{1,l}^n, \phi_{2,l}^n, \cdots, \phi_{M-2,l}^n, \phi_{M-1,l}^n)^T \\ F_l^{n+1/2} &= (f_{1,l}^{n+1/2} + g_{y0}, f_{2,l}^{n+1/2}, \cdots, f_{M-2,l}^{n+1/2}, f_{M-1,l}^{n+1/2} + g_{yM})^T \end{aligned} \tag{3.53}$$

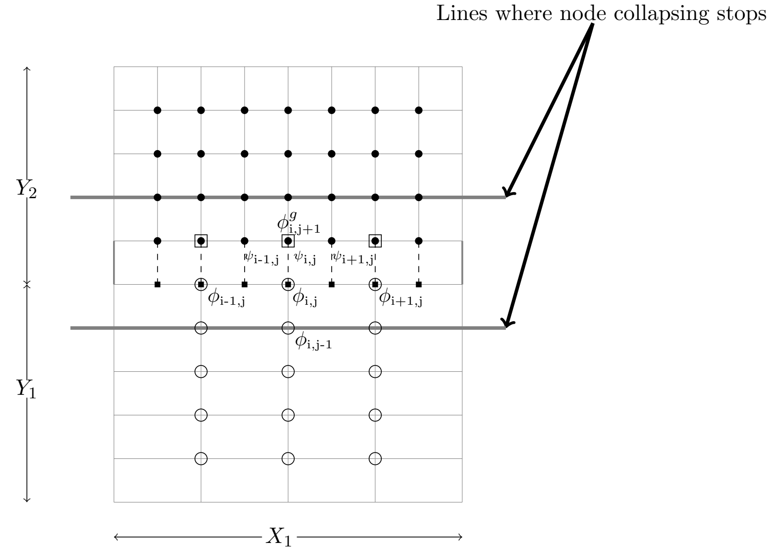
Step 2 (y-sweep).

For each $m = 1, \cdots, M-1$ solve the tridiagonal system

$$(I - B_{2h})\Phi_m^{n+1} = (1 + B_{1h})\Phi_m^{n+1/2} + \frac{r}{2} F_m^{n+1} \tag{3.54}$$

where

$$\begin{aligned} \Phi_m^{n+1} &= (\phi_{m,1}^{n+1}, \phi_{m,2}^{n+1}, \cdots, \phi_{L-1,l}^{n+1}, \phi_{L-1,l}^{n+1})^T \\ \Phi_m^{n+1/2} &= (\phi_{m,1}^{n+1/2}, \phi_{m,2}^{n+1/2}, \cdots, \phi_{M-2,l}^{n+1/2}, \phi_{L-1,l}^{n+1/2})^T \\ F_m^{n+1} &= (f_{m,1}^{n+1} + g_{x0}, f_{m,2}^{n+1}, \cdots, f_{m,L-2}^{n+1}, f_{L-1,l}^{n+1} + g_{xL})^T \end{aligned} \tag{3.55}$$



Legend:

○ – internal points of lower domain

● – internal points of upper domain

□ – lower domain ghost points

▪ – upper domain ghost points

Figure 3.13: Domain with partially matching grid.

Thus on each step we solve set of independent equations ($L-2$ equations for x-sweep and $M-2$ equation for y-sweep) which can be solved independently (e.g in parallel).

Domain with partially matching grid

Let us consider domain with a grid from figure 3.13. Its feature is that all points from lower sub-domain match with points in upper sub-domain, but not all points from upper sub-domain have a match in a lower one. So in order to deal with domains like this we introduce interpolation procedure.

Interpolation procedure

Let us denote unknowns from lower domain and upper domain ϕ and ψ respectively. For the domain on the figure 3.13 only operator δ_y^2 requires interpolation. Let us describe interpolation procedure for the following points: $\phi_{i,j}$, $\psi_{i,j}$, $\psi_{i-1,j}$:

$$\begin{aligned}
\delta_y^2\phi_{i,j} &= \underbrace{(\frac{1}{4}\psi_{\mathrm{i-1,j}} + \frac{1}{2}\psi_{\mathrm{i,j}} + \frac{1}{4}\psi_{\mathrm{i+1,j}})}_{\phi^g_{i,j+1}} - 2\phi_{i,j} + \phi_{i,j-1} \\
\delta_y^2\psi_{i,j} &= \psi_{i,j+1} - 2\psi_{i,j} + \underbrace{(\frac{1}{2}\phi_{i-1,j} + \frac{1}{2}\phi_{i+1,j})}_{\psi^g_{i,j-1}} \\
\delta_y^2\psi_{i-1,j} &= \psi_{i-1,j+1} - 2\psi_{i-1,j} + \underbrace{(\frac{1}{2}\phi_{i-1,j} + \frac{1}{2}\phi_{i,j})}_{\psi^g_{i-1,j-1}}
\end{aligned} \tag{3.56}$$

Disadvantage of this interpolation is that it spoils main feature of ADI methods: ability to solve system in parallel (in case of domain from figure 3.13, y-sweep lines aren't independent anymore). Following subsection will describe how we can overcome limitation imposed by interpolation procedure.

Unknowns collapsing

Lower sub-domain.

So, we have $m = 1, \cdots, M_1 - 1$ lines in lower domain and for first $l_1 = 1, \cdots, L_1 - 3$ unknowns we can write following system (equation for these unknowns has three point structure):

$$\begin{cases}
b_{m,1}\phi_{m,1} + c_{m,1}\phi_{m,2} = F_{m,1} \\
a_{m,2}\phi_{m,1} + b_{m,2}\phi_{m,2} + c_{m,2}\phi_{m,3} = F_{m,2} \\
\dots \\
a_{m,l_1}\phi_{m,l_1-1} + b_{m,l_1}\phi_{m,l_1} + c_{m,l_1}\phi_{m,l_1+1} = F_{m,l_1} \\
\dots \\
a_{m,L_1-3}\phi_{m,L_1-4} + b_{m,L_1-3}\phi_{m,L_1-3} + c_{m,L_1-3}\phi_{m,L_1-2} = F_{m,L_1-3}
\end{cases} \tag{3.57}$$

Let us start to express the unknowns. First we can express ϕ_2 through ϕ_1 using first equation of system (3.57):

$$\phi_{m,2} = K_{m,1}\phi_{m,1} + F'_{m,1}, \tag{3.58}$$

where $K_1 = -\frac{b_1}{c_1}$, $F'_1 = \frac{F_1}{c_1}$. Then we can express ϕ_3 through ϕ_1 using second equation of system (3.57) and equation (3.58). Processing like this we can derive the following recurrent formulation:

$$\begin{cases} \phi_{m,i} = K_{m,i-1}\phi_{m,1} + F^{'}_{m,i-1} \\ K_{m,i-1} = -\frac{a_{m,i-1}}{c_{m,i-1}}K_{m,i-3} - \frac{b_{m,i-1}}{c_{m,i-1}}K_{m,i-2} \\ F^{'}_{m,i-1} = \frac{1}{c_{m,i-1}}F_{m,i-1} - \frac{a_{m,i-1}}{c_{m,i-1}}F^{'}_{m,i-3} - \frac{b_{m,i-1}}{c_{m,i-1}}F^{'}_{m,i-2} \end{cases} \tag{3.59}$$

Upper sub-domain.

For lines $m = 1, \cdots, M_2 - 1$ in upper domain and for last $l_2 = 2, \cdots, L_2$ unknowns we can write the following system:

$$\begin{cases} a_2\phi_1 + b_2\phi_2 + c_2\phi_3 = F_2 \\ \cdots \\ a_{l_2}\phi_{l_2-1} + b_{l_2}\phi_{l_2} + c_{l_2}\phi_{l_2+1} = F_{l_2} \\ \cdots \\ a_{L_2-2}\phi_{L_2-3} + b_{L_2-2}\phi_{L_2-2} + c_{L_2-2}\phi_{L_2-1} = F_{L_2-2} \\ a_{L_2-1}\phi_{L_2-2} + b_{L_2-1}\phi_{L_2-1} = F_{L_2-1} \end{cases} \tag{3.60}$$

In the same manner as for lower domain we can obtain the following recurrent formulation:

$$\begin{cases} \phi_i = K_{i+1}\phi_1 + F^{'}_{i+1} \\ K_{i+1} = -\frac{c_{i+1}}{c_{i+1}}K_{i-3} - \frac{b_{i+1}}{a_{i+1}}K_{i-2} \\ F^{'}_{i+1} = \frac{1}{a_{i+1}}F_{i-1} - \frac{c_{i+1}}{a_{i+1}}F^{'}_{i-3} - \frac{b_{i+1}}{a_{i+1}}F^{'}_{i-2} \end{cases} \tag{3.61}$$

Therefore we can formulate the following algorithm which employs advantages given by collapsing procedure:

1. Calculate and store coefficient for lower and upper sub-domains using equations (3.57) and (3.61) respectively.
2. Discretize and solve system for $M_1 \times L_1$ points, where interpolation procedure should be applied.
3. Restore values of all points using coefficients from step 1.

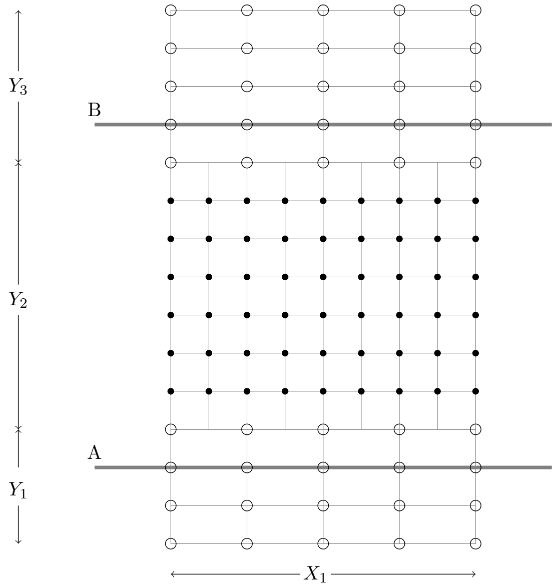


Figure 3.14: Domain with several non-matching boundaries

Domain with several partially matching interfaces

Let us consider domain from figure 3.14, if we are using solution procedure with node collapsing, which was proposed above, we can only collapse node for lower sub-domain until line A and from line B for upper sub-domain. For the points from line A to line B we should solve full system which leads to impossibility of taking advantage of direction splitting for internal domain with fine grid. Iterative solution procedure can be used in order to deal with this problem.

Iterative solution procedure

Iterative solution procedure relies on matrix decomposition. In order to explain the idea let us again consider domain from figure 3.13.

Matrix decomposition.

$$\left(\begin{array}{c|c|c} A_{11} & A_{12} & 0 \\ \hline B_1 & B_2 & B_3 \\ \hline 0 & A_{21} & A_{22} \end{array}\right) \left(\begin{array}{c} \Phi \\ \hline W \\ \hline \Psi \end{array}\right) = \left(\begin{array}{c} F_1 \\ \hline F_{12} \\ \hline F_2 \end{array}\right), \tag{3.62}$$

In order to construct such decomposition we should define only shape of matrix A_{11} and matrix A_{22}, because forms of other sub-matrices naturally depend on those two:

- both matrices are square, let assume that A_{11} has size $n_1 \times n_1$, A_{22} has size $n_2 \times n_2$
- n_1 equals to number of point in the lower sub-domain until line A (including points in this line)
- n_2 equals to number of point in the upper sub-domain from line B (including points in this line)

Iterative algorithm.

Iterative procedure consists of following steps:

1. Start with initial guess for Φ_{12}^0 (superscript denotes the iteration level).
2. Setup a threshold. Threshold indicates when we should stop iterations.
3. Solve equation for internal unknowns:

$$\begin{cases} A_{11}\Phi^{k+1} = F_1 - A_{12}W^k \\ A_{22}\Psi^{k+1} = F_2 - A_{21}W^k \end{cases} \tag{3.63}$$

4. Solve equation for boundary unknowns (restore value of W for next iteration level):

$$B_2W^{k+1} = F_{12} - B_1\Phi^{k+1} - B_3\Psi^{k+1} \tag{3.64}$$

5. If any value in difference between vector $(\Phi^{k+1}, W^{k+1}, \Psi^{k+1})$ and (Φ^k, W^k, Ψ^k) is greater than threshold then go to step 3.

h	$\|\phi_h - \phi\|_{L_\infty}$	$\|\phi_h - \phi\|_{L_2}$
0.1	0.069	0.001
0.05	0.032	0.0002
0.025	0.013	4.2e-5

Table 3.3: Errors between analytic solution and numerical solution.

Shur-complement solution

Instead of using iterative procedure we can get value for W using an algorithm based on idea of Shur-complement [101]:

$$(B_2 - B_1 A_{11}^{-11} A_{21} - B_3 A_{22}^{-1} A_{21})W = F_{12} - B_1 A_{11}^{-1} F_1 - B_3 A_{22}^{-1} F_2 \tag{3.65}$$

Then solve the system in order to restore values for Φ and Ψ for the next time step:

$$\begin{cases} A_{11}\Phi = F_1 - A_{12}W \\ A_{22}\Psi = F_2 - A_{21}W \end{cases} \tag{3.66}$$

Numerical experiments

Let us test method using following analytic solution:

$$\omega(x, y, t) = exp(-t)sin(\pi x)sin(\pi y) \tag{3.67}$$

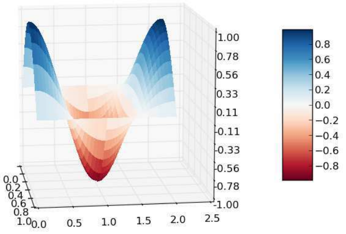
We will perform test on two geometries:

- geometry from figure 3.13 ($X_1 = 1, Y_1 = 1., Y_2 = 1.$)
- geometry from figure 3.14 ($X_1 = 1, Y_1 = 1., Y_2 = 1., Y_3 = 1.$).

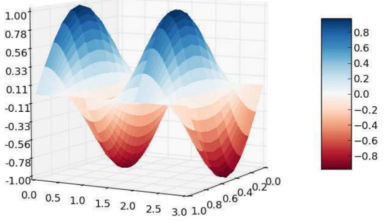
Figures 3.15a, 3.15b show numerical solution for both geometries. Table 3.3 shows maximum norm of error between analytic solution and numerical solution:

3.3.2 Application to the Navier-Stokes equations.

Now we want to apply described method to the time-dependent Navier-Stokes equations (2.1) (assuming $Q = 0$). In order to apply proposed method to the (2.12) we will solve



(a) Numerical solution of problem (3.46) on domain 3.13.



(b) Numerical solution of problem (3.46) on domain 3.14.

equations (2.1) with fractional time step direction splitting method proposed in [42], where momentum equation is also solved in a splitting manner:

Pressure predictor.

Compute a pressure predictor by setting $p^{\frac{1}{2}} = 0$ and for $n \geq 0$:

$$p^{*,n+\frac{1}{2}} = p^{n-\frac{1}{2}} + \phi^{n+\frac{1}{2}} \tag{3.68}$$

Velocity update.

Updating the velocity field by using Douglas [20] direction splitting technique. Initialize the algorithm with $\mathbf{u}^0 = \mathbf{u}_0$, and for $n \geq 0$ velocity is updated as follows:

$$\begin{aligned}
&\frac{\xi^{n+1} - \mathbf{u}^{n+1}}{\Delta t} - \nu \Delta \mathbf{u}^n + \nabla p^{*,n+\frac{1}{2}} = f(t^{n+\frac{1}{2}}) \\
&\frac{\sigma^{n+1} - \xi^{n+1}}{\Delta t} + \frac{\nu}{2} \partial_{xx}(\sigma^{n+1} - \mathbf{u}^n) = 0 \\
&\frac{\mathbf{u}^{n+1} - \sigma^{n+1}}{\Delta t} + \frac{\nu}{2} \partial_{yy}(\mathbf{u}^{n+1} - \mathbf{u}^n) = 0
\end{aligned} \tag{3.69}$$

Pressure update.

The pressure is updated by solving $A p^{n+\frac{1}{2}} = \dfrac{1}{\Delta t} \nabla \cdot u^{n+1}$, where A is a direction splitting operator of following form $A = (1 - \partial_{xx})(1 - \partial_{yy})$.

$$\begin{aligned}
&\psi - \partial_{xx}\psi = -\frac{1}{\Delta t} \nabla \cdot u^{n+1} \\
&p^{n+\frac{1}{2}} - \partial_{yy} p^{n+\frac{1}{2}} = \psi
\end{aligned} \tag{3.70}$$

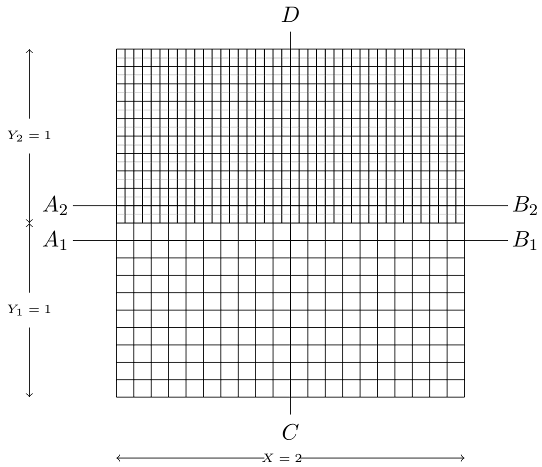


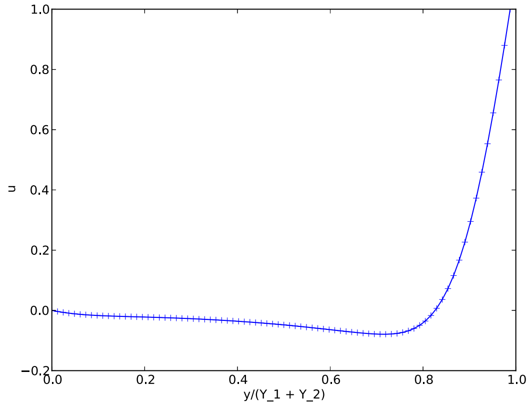
Figure 3.16: Lid driven cavity on domain with partially matching grid.

By applying this algorithm to the equations (2.1) we'll get set of one dimensional equations on each step, so algorithm from previous subsection can be applied without significant changes.

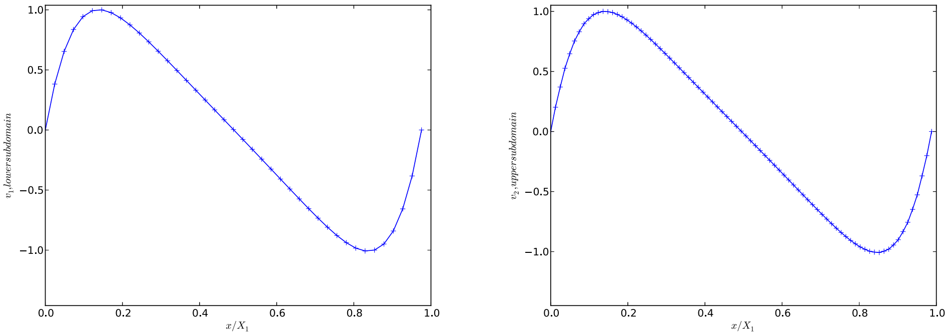
Numerical experiment: lid driven cavity problem.

As numerical experiment we have chosen classical 2D lid driven cavity problem. The computational domain is $\Omega = (0, 2)^2$. The boundary conditions are $u|_{x=0,1,y=0} = 0$, $u|_{y=1} = 1$ and $v|_{\partial\Omega} = 0$. The computation is done at Reynolds number $Re \approx 1$ with time step $\tau = 0.001$, with the end time $T = 2$. Lower part of the domain has coarse grid $40{\times}40$ and upper domain has finer grid $80{\times}40$.

We show in Figure 3.17 the horizontal and vertical profiles of the velocity alongside the vertical/horizontal lines through the center of the cavity (see lines A_1B_1, A_2B_2, CD in the Figure 3.16).



(a) First velocity component u in a slice CD.



(b) Second velocity component v in a slice A_1B_1.

(c) Second velocity component v in a slice A_2B_2.

Figure 3.17: Velocity slices in a lid driven cavity problem.

Chapter 4

Numerical algorithm for non-isothermal flows

4.1 Fractional time step discretization of the Oberbeck-Boussinesq equations

First, we recall a classical fractional time step discretization approach, and after that will introduce an efficient directional splitting algorithm.

Fractional time step discretization of Navier-Stokes equations and heat equation [14, 35, 41]. The equations (2.22), (2.23) are discretized in time using a first order fractional time step technique which can be summarized as follows: given the values for $\mathbf{u}^n$, p^n and the pressure-correction at time t^n, ϕ^n, the transition to the next time level, t^{n+1}, is done as follows:

Temperature update:

$$\begin{aligned} \frac{T^{n+1}-T^n}{\tau} + (\mathbf{u}^n \cdot \nabla) T^n - \nabla \cdot (\lambda \nabla T^{n+1}) = 0, \\ T^{n+1}|_{\partial\Omega} = T_b. \end{aligned} \tag{4.1}$$

Pressure predictor:

$$p^{*,n+1} = p^n + \phi^n \tag{4.2}$$

Velocity update:

$$\frac{\mathbf{u}^{n+1}-\mathbf{u}^n}{\tau} + (\mathbf{u}^n \cdot \nabla)\mathbf{u}^n - \nu \Delta \mathbf{u}^{n+1} + \frac{1}{\rho} \nabla p^{*,n+1} = -\beta \mathbf{g} T^{n+1} \tag{4.3}$$

Pressure corrector:

$$\Delta\phi^{n+1} = -\frac{\rho\nabla\cdot\mathbf{u}^{n+1}}{\Delta t} \tag{4.4}$$

Pressure update:

$$p^{n+1} = p^n + \phi^{n+1} - \nu\chi\nabla\cdot\mathbf{u}^{n+1}, \text{ for } \chi\in[0,1] \tag{4.5}$$

Although we consider unsteady Navier-Stokes equation, at each time step we have to solve the elliptic equation (4.4), what requires significant amount of CPU time, even when advanced methods like multigrid are used. Therefore, here we also want to employ fast direction splitting algorithm, which was described in a previous section 3.2. As before in order to do it we need to introduce model which works also in simple domains, thus we formulate penalized version of the Navier-Stokes equation with Oberbeck-Boussinesq approximation (2.22) in a same fashion as it was done in Section 2.1.3:

$$\begin{aligned}&\frac{\partial\mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\,\mathbf{u} - \nu\Delta\mathbf{u} + \frac{1}{\rho}\nabla p + \frac{\nu}{K}\mathbf{u} = -\beta\mathbf{g}T, \text{ in } \Omega\times[0,T]\\ &\nabla\cdot\mathbf{u} = 0 \text{ in } \Omega\times[0,T],\end{aligned} \tag{4.6}$$

where value for K depends on the domain (see system (2.11)).

The heat equation (2.23) has also to be modified. It looks the same except that now the expression for the thermal diffusivity in formula is space dependent:

$$\lambda(t,\mathbf{x}) = \begin{cases}\tilde{\lambda} & \mathbf{x}\in\Omega\setminus\Omega_f\\ \lambda & \mathbf{x}\in\Omega_f\end{cases} \tag{4.7}$$

The choice of $\tilde{\lambda}$ depends on the type of the boundary conditions (Dirichlet or Neumann): $\tilde{\lambda}=\lambda$ for Dirichlet boundary conditions, and $\tilde{\lambda}\to\infty$ in the case of Neumann boundary conditions. For details see, e.g. [47].

Fractional time step discretization of Navier-Stokes equations exploiting direction splitting [42].

We change pressure correction step (4.4) in the fractional time step scheme proposed above to the following equation:

$$A\phi^{n+1} = \frac{\rho\nabla\cdot\mathbf{u}^{n+1}}{\Delta t} \tag{4.8}$$

The solution to problem (4.8) can in fact be computed as the following sequence of one dimensional problems

$$\begin{aligned}
&\psi - \partial_{xx}\psi = -\frac{\rho\nabla\cdot\mathbf{u}^{n+1}}{\Delta t},\\
&\eta - \partial_{yy}\eta = \psi,\\
&\phi - \partial_{zz}\phi = \eta.
\end{aligned}$$

Discretization of a penalized momentum equation (4.6) in the velocity update step reads as:

$$\frac{\mathbf{u}^{n+1}-\mathbf{u}^{n}}{\tau} + (\mathbf{u}^{n}\cdot\nabla)\mathbf{u}^{n} - \nu\Delta\mathbf{u}^{n+1} + \nabla p^{*,n+1} = -\beta T^{n+1}\mathbf{g} + \frac{1}{2K}(\mathbf{u}^{n+1}+\mathbf{u}^{n}) \qquad (4.9)$$

Thus fractional time step discretization of Navier-Stokes equations exploiting direction splitting consists from the following equations (4.1), (4.9), (4.8), (4.5).

4.2 Numerical experiments

4.2.1 Preliminaries

Computer simulations which are presented here have as a main goal the demonstration that the directional splitting approach (4.1–4.9, 4.8, 4.5) is efficient in simulating the fluid flow and heat transfer in the containment pool of nuclear reactor. It is faster then the algorithm based on the classical Poisson equation for the pressure correction step, and at the same time provide practically the same accuracy. In fact, this is expected, because it is known that the accuracy of the directional splitting approach depends on the time step, which is the perturbation parameter. In our case small time step has to be used due to the moving interfaces. The developed algorithm is validated in comparison with measurement results for carefully selected realistic cases. This is subject of a separate forthcoming paper.

The two sets of simulations are as follows:
i. side heated cavity, solved with and without Fictitious Domain Method;
ii. heat and mass transfer in ThAI experimental facility (THAI [88] is experimental facility build especially for performing experiments for the needs of nuclear reactor safety).

Simulations were performed using two sets of equations which we will refer as FD-M-DS and SD-SPS:

- FD-M-DS formulation: directional splitting algorithm as described by equations ((4.1), (4.9), (4.8),(4.5)), solved on a domain, which is augmented to cuboid ($\Omega \cup \Omega_f$)
- SD-SPS: equations (4.1 – 4.5) with standard Poisson-type equation on pressure correction step (4.4) on standard domain (Ω).

Parallelization

In order to reduce computational time we've created parallel implementation of the FD-M-DS model:

1. We solve decoupled momentum equation, therefore on each timestep equations for different velocity components can be computed in parallel.
2. While solving equation (3.11) we solve set of independent one dimensional equations.

Our parallel implementation are based on threads (we have used Boost implementation of threads in order to be platform independent).

For the first use case from enumeration above we run equation for each component of velocity in separate thread. For the second use case we implemented thread pool pattern in order to distribute tasks (solution of one dimensional equations) between threads. Numbers of threads in this case depend on underlying hardware configuration.

All simulations were performed on a machine with a dual core Intel Xeon 5148LV with 8 GB RAM.

4.2.2 Side heated 3D cavity.

The side heated cavity is a classical problem for studying and validation of the flow and heat transfer.

Simulation domain is shown in the figure 4.1, no-slip velocity boundary conditions ($u = v = w = 0$) are applied on the walls of the square cavity. For temperature Dirichlet boundary conditions are enforced on the left (hot wall) and right (cold wall) vertical walls. As there is no transfer of heat through the horizontal walls a Neumann boundary condition is applied.

Our goal for this simulation is to obtain steady state flow patterns of the buoyancy-driven cavity for the following range of Rayleigh number $Ra = 10^3, 10^4$ and 10^5. We used expression 2.24 in order to compute dimensionless Rayleigh number using simulation setup

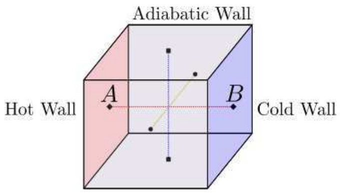


Figure 4.1: Simulation domain for side heated cavity problem.

data. In the figure 4.2 computed flow patterns are shown. In the left panel we display the vertical profiles of the velocity along the horizontal segment AB from figure 4.1, with dots we display velocity calculated with FD-M-DS model, using line we display velocity calculated with SD-SPS model. Comparison of this two models demonstrates good agreement between them. In the middle and right panel we show velocity vectors and isothermal lines respectively.

For the low Rayleigh number we can see that velocity vectors form a single circulating eddy, but as the Rayleigh number increases vectors tend to divide into counter rotating eddies. For $Ra = 10^3$ the isotherms tend to be parallel to the vertical walls, however, as the Rayleigh number increases the isotherms tend to become parallel to the adiabatic walls. This results are in a good agreement with an expected physical behavior.

In this set of simulations we also addressed a stability problem, so we performed tests with wide range of time steps and grid sizes have shown stable behavior of proposed scheme.

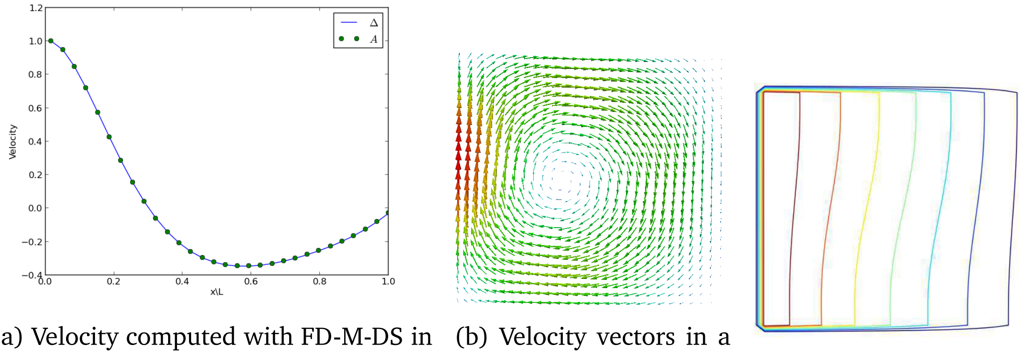
4.2.3 ThAI facility

The ThAI facility is a technical scale experimental facility for research in the area of nuclear reactor containment safety. The ThAI test vessel dimensions are: height of $9.2m$, diameter of $3.2m$, free volume of 60 m^3. Depending on experimental needs there is also can be an inner cylinder (see 4.3b) [88].

Simulation 1

In this simulation container is filled with water at a temperature of 20 degrees, water at a temperature of 25 degrees comes from inlet, with the same speed water drain out through outlet (we prescribe non zero third component of velocity $w = 0.5$). We use no-slip velocity boundary condition and Neumann for pressure and temperature on other walls (fig. 4.3b).

$Ra = 10^3$

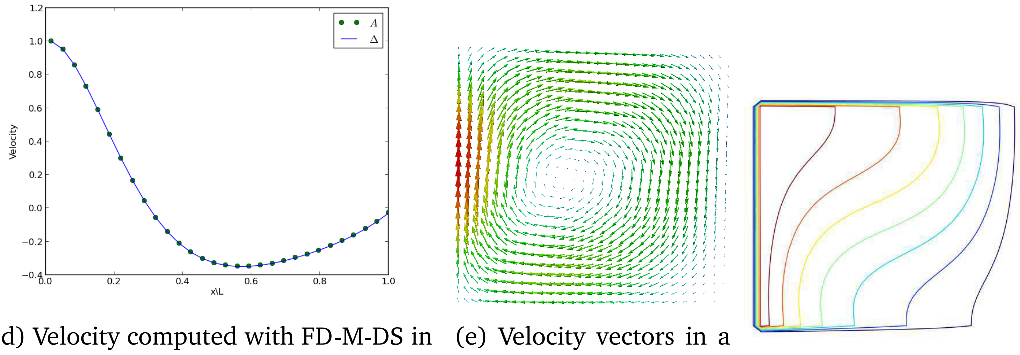


(a) Velocity computed with FD-M-DS in comparison with SD-SPS

(b) Velocity vectors in a slice

(c) Isotherms in a slice

$Ra = 10^4$

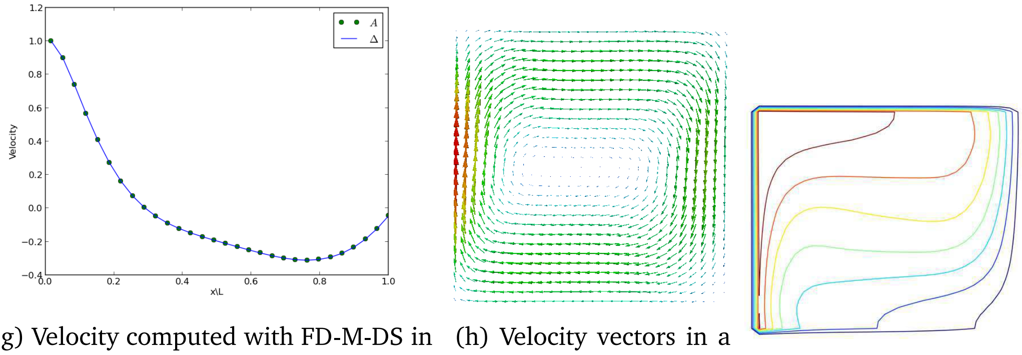


(d) Velocity computed with FD-M-DS in comparison with SD-SPS

(e) Velocity vectors in a slice

(f) Isotherms in a slice

$Ra = 10^5$

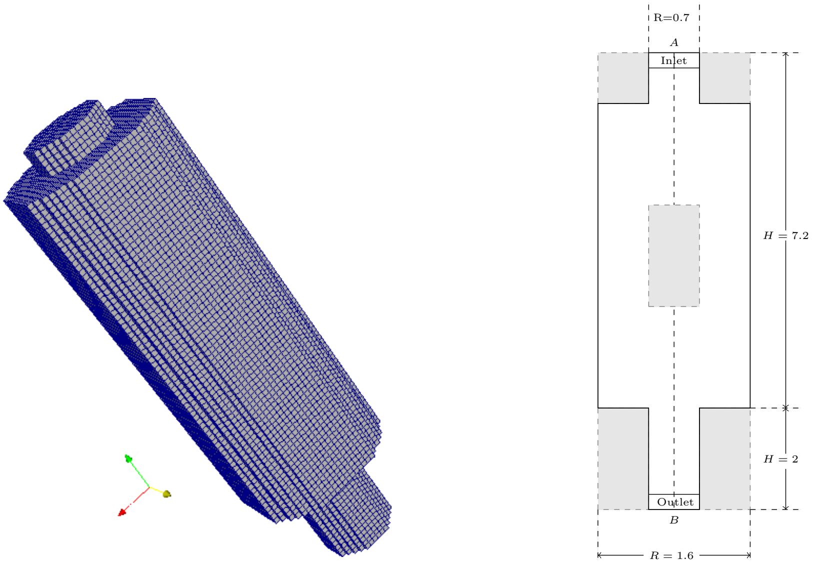


(g) Velocity computed with FD-M-DS in comparison with SD-SPS

(h) Velocity vectors in a slice

(i) Isotherms in a slice

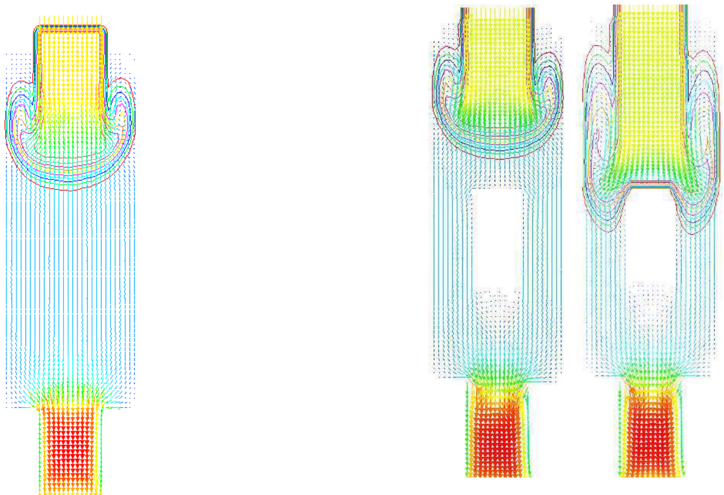
Figure 4.2: Convection patterns for different Rayleigh numbers.



(a) 3D model of ThAI.

(b) Shaded areas on the right panel belong to $\Omega \setminus \Omega_f$. Shaded cylinder in a central part is used only in the second simulations from the Section 4.2.3

Figure 4.3: ThAI facility.



(a) Simulation results for setup from section 4.2.3.Velocity vectors and temperature contours in a slice.

(b) Simulation results for setup from section 4.2.3. Velocity vectors and temperature contours in a slice for two different moments in time.

Figure 4.4: Simulation results for ThAI facility.

Table 4.1: CPU time comparison for experiments from section 4.2.3

Number of points (FD-M-DS)	78300	671800
Number of points (SD-SPS)	55700	453300
Time (FD-M-DS)	0.6	5.1
Time (FD-M-DS, parallel code)	0.35	3.0
Time (SD-SPS)	1.65	24.3

In the figure 4.4a we show simulation results: velocity vectors and temperature contours in a slice. Behavior of temperature contours tells us that convection in this simulation is mixed: one part of hot water flows down, another part of hot water flows up.

In the table 4.1 we show CPU time comparison results for FD-M-DS and SD-SPS models. Simulations with FD-M-DS model is significant faster than simulation with SD-SPS model.

Simulation 2

In this simulation let us consider ThAI geometry with inner cylinder (see fig. 4.3b). Boundary Prescribed boundary conditions are similar to the previous simulation. For our simulation with direction splitting operator (FD-M-DS model) inner cylinder belongs to fictitious part of domain (Ω_f).

In right panel of the figure 4.4b we show velocity vectors and temperature contours in a slice.

It is not a surprise that in this simulation FD-M-DS model was also significantly faster than SD-SPS model, CPU time was similar to the time presented in the previous section.

4.3 Conclusion

In this section the direction splitting model for the incompressibility constraint was successfully applied for simulation fluid flow and heat transfer, which have been modeled with set of Navier-Stokes equations with Oberbeck-Boussinesq approximation and heat equation.

Main characteristic of proposed algorithm is significant computational time reduction in comparison with standard way of dealing with incompressibility constraint in Navier-Stokes equations.

Chapter 5

Summary

This thesis substantially contributes to developing customized algorithms and software for simulation of hydrodynamic processes in containment pools of Nuclear Power Plants. Approaches and algorithms described in this thesis became a part of a software tools named CoPool, which is currently used by researchers and engineers working on evaluation of NPP safety. The following main building blocks are considered:

- Evaluation of existing mathematical models and discretization techniques and their classification according to the specific needs of containment pool simulation;
- Development of customized pre-processor;
- Development of fast directional splitting numerical algorithms for solving isothermal and non-isothermal Navier-Stokes-Brinkman equations in complicated domains;
- Careful analysis of the proposed models and algorithms, and study of their performance and stability.

The main contributions of this thesis concern:

- The overview, classification and analysis of models and discretization techniques in accordance with the specific requirements for containment pool simulation;
- The fast directional splitting numerical algorithm for efficient simulation of isothermal fluid flow based on a penalized Navier-Stokes equations;
- The fast directional splitting numerical algorithm for efficient simulation of non-isothermal fluid flow and temperature stratification in containment pools;
- Simulation of accident scenarios for realistic geometries.

Further results include contribution to developing the pre-processor and contribution to the testing of the overall software.

Nomenclature

$\mathbf{u}$, p, $T^{'}$	velocity, pressure and temperature respectively – are the main unknowns in the problems we consider.
u, v, w	first, second and third components of velocity vector **u**
T	time, indicates end of simulation
τ	time step
Ω	computational domain
Ω_p	porous part of the computational domain
Ω_s	solid part of the computational domain
Ω_f	fluid part of the computational domain
d	dimension of the problem $d = 2$ (2D) or $d = 3$ (3D)
μ	kinematic viscosity $[kg/ms]$
ρ	density $[kg/m^3]$
$\mathbf{f}$	body forces
Q	mass
Pr	Prandtl number
Ra	Rayleigh number
Pe	Peclet number
β	thermal expansion coefficient
λ	thermal conductivity $[W/mK]$

Acronyms

IB	Immersed boundary
FD	Fictitious Domain
FTS	Fractional Time Step
ADI	Alternating Direction Implicit
$ThAI$	Thermal-hydraulics, Hydrogen, Aerosols and Iodine.

Bibliography

[1] H.-J. Allelein. International Standard Problem ISP-47 on Containment Thermal Hydraulics., 2007.

[2] Ph. Angot, J. Keating, and P.D. Minev. A direction splitting algorithm for incompressible flow in complex geometries. *Comp. Meth. Appl. Mech. Engng*, 2012.

[3] Philippe Angot. Analysis of singular perturbations on the Brinkman problem for fictitious domain models of viscous flows. *Mathematical Methods in the Applied Sciences*, 22(16):1395–1412, 1999.

[4] Philippe Angot. A fictitious domain model for the Stokes/Brinkman problem with jump embedded boundary conditions. *Comptes Rendus Mathematique*, 348(1112):697 – 702, 2010.

[5] ANSYS Fluent. `http://www.ansys.com/Products/Simulation+Technology/Fluid+Dynamics/Fluid+Dynamics+Products/ANSYS+Fluent`. [Online; accessed 2013-08-21].

[6] Petter A. Berthelsen and Odd M. Faltinsen. A local directional ghost cell approach for incompressible viscous flow problems with irregular boundaries. *J. Comput. Phys.*, 227(9):4354–4397, April 2008.

[7] F. Bertrand, P.A. Tanguy, and F. Thibault. A three-dimensional fictitious domain method for incompressible fluid flow problems. *Int. J. Numer. Meth. Fluids*, 25:719–736, 1997.

[8] J. Bestele and W. Klein-Heling. Cpa-module containment thermal hydraulics and aerosol- and fission product behaviour - user guidelines. *ASTEC V0/DOC/00-14, GRS*, 2000.

[9] R. P. Beyer and R. J. Leveque. Analysis of a one-dimensional model for the immersed boundary method. *SIAM Journal on Numerical Analysis*, 29:332–364, 1992.

[10] J. Boussinesq. *Théorie de L'Écoulement Tourbillonnant Et Tumultueux Des Liquides Dans Les Lits Rectilignes a Grande Section*. Number v. 1. BiblioBazaar, 2010.

[11] H.C. Brinkman. A calculation of the viscous force exerted by a flowing fluid on a dense swarm of particles. *Applied Scientific Research*, 1(1):27–34, 1949.

[12] Yoann Cheny and Olivier Botella. The ls-stag method: A new immersed boundary/level-set method for the computation of incompressible viscous flows in complex moving geometries with good conservation properties. *Journal of Computational Physics*, 229(4):1043 – 1076, 2010.

[13] Jung-Il Choi, Roshan C. Oberoi, Jack R. Edwards, and Jacky A. Rosati. An immersed boundary method for complex incompressible flows. *Journal of Computational Physics*, 224(2):757 – 784, 2007.

[14] A.J. Chorin. Numerical solution of the Navier-Stokes equations. *Math. Comp.*, 22:745–762, 1968.

[15] COMSOL Multiphysics. The Platform for Physics-Based Modeling and Simulation. `http://www.comsol.de/comsol-multiphysics`. [Online; accessed 2013-08-21].

[16] Containment building. `http://en.wikipedia.org/wiki/Containment\_building`. [Online; accessed 2013-08-21].

[17] J. Crank and P. Nicolson. A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. *Advances in Computational Mathematics*, 6(1):207–226, 1996.

[18] Clarke D., Salas M., and Hassan H.. Euler calculations for multi-element airfoils using cartesian grids. *AIAA J.*, (24):112835, 1986.

[19] H. Darcy. Les Fontaines Publiques de la Ville de Dijon. 1856.

[20] Jr. Douglas, Jim. Alternating direction methods for three space variables. *Numerische Mathematik*, 4(1):41–63, 1962.

[21] PA Durbin and G Iaccarino. An approach to local refinement of structured grids. *Journal of Computational Physics*, 181(2):639–653, 2002.

[22] Paul Durbin, Sekhar Majumdar, and Gianluca Iaccarino. RANS solvers with adaptative structured boundary non conforming grids. *Annual Research Briefs*, pages 353–364, 2001.

[23] R. Eymard, T. R. Gallout, and R. Herbin. The finite volume method. *Handbook of Numerical Analysis*, VII, 2000.

[24] E.A. Fadlun, R. Verzicco, P. Orlandi, and J. Mohd-Yusof. Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations. *Journal of Computational Physics*, 161(1):35 – 60, 2000.

[25] Lisa J. Fauci and Charles S. Peskin. A computational model of aquatic animal locomotion. *J. Comput. Phys.*, 77(1):85–108, July 1988.

[26] J. H. Ferziger and M. Peric. *Computational Methods for Fluid Dynamics*. Springer, Berlin, 1999.

[27] K. Fischer. Numerische Modelle fuer Tide und Salzgehaltsverteilung im Muendungsbereich der Ems. pages 29–39, 1979.

[28] Aaron L Fogelson. A mathematical model and numerical method for studying platelet adhesion and aggregation during blood clotting. *Journal of Computational Physics*, 56(1):111 – 134, 1984.

[29] Roberto Verzicco Gianluca Iaccarino. Immersed boundary technique for turbulent flow simulations. *Appl Mech Rev*, 56(3), May 2003.

[30] Frederic Gibou, Ronald P. Fedkiw, Li-Tien Cheng, and Myungjoo Kang. A second-order-accurate symmetric discretization of the poisson equation on irregular domains. *Journal of Computational Physics*, 176(1):205 – 227, 2002.

[31] Roland Glowinski, Tsorng-Whay Pan, Todd I. Hesla, Daniel D. Joseph, and Jacques Priaux. A distributed lagrange multiplier/fictitious domain method for flows around moving rigid bodies: Application to particulate flow. *International Journal for Numerical Methods in Fluids*, 30(8):1043–1066, 1999.

[32] D. Goldstein, R. Handler, and L. Sirovich. Modeling a no-slip flow with an external force eld. *Journal of Computational Physics*, 105, 1993.

[33] D. Goldstein, R. Handler, and L. Sirovich. Direct numerical simulation of turbulent flow over a modelled riblet covered surface. *J. Fluid Mech.*, 302, 333, 1995.

[34] T. Gornak, J.L. Guermond, O. Iliev, and P.D. Minev. A direction splitting approach for incompressible Brinkman flow. *Int. J. Numer. Analysis Modeling*, 4(1–13), 2013.

[35] P.M. Gresho and S.T. Chan. On the theory of semi-implicit projection methods for viscous incompressible flow and its implementation via finite element method that also introduces a nearly consistent mass matrix. part I and part II. *Int. J. Numer. Methods Fluids*, 11:587–620, 621–659, 1990.

[36] J.-L. Guermond, P. Minev, and J. Shen. An overview of projection methods for incompressible flows. *Comput. Methods Appl. Mech. Engrg.*, 195:6011–6054, 2006.

[37] J.-L. Guermond and P.D. Minev. A new class of splitting methods for the incompressible Navier-Stokes equations using direction splitting. *Comput. Methods Appl. Mech. Engrg.*, 200:2083–2093, 2011.

[38] J.-L. Guermond and Abner Salgado. A splitting method for incompressible flows with variable density based on a pressure poisson equation. *Journal of Computational Physics*, 228(8):2834 – 2846, 2009.

[39] J. L. Guermond and J. Shen. On the error estimates for the rotational pressure-correction projection methods. *Math. Comp.*, 73(248):1719–1737, 2004.

[40] Jean-Luc Guermond, Peter D. Minev, and Abner J. Salgado. Convergence analysis of a class of massively parallel direction splitting algorithms for the navier-stokes equations in simple domains. *Math. Comput.*, 81(280), 2012.

[41] J.L. Guermond and P.D. Minev. Analysis of a projection/characteristic scheme for incompressible flow. *Comm. Numer. Meth. Engng.*, 19:535, 2003.

[42] J.L. Guermond and P.D. Minev. A new class of fractional step techniques for the incompressible Navier-Stokes equations using direction splitting. *C.R.Acad.Sci Paris, Ser. I*, 348:581–585, 2010.

[43] Bandringa Henry. Immersed boundary methods. `http://www.math.rug.nl/~veldman/Scripties/Bandringa-MasterTechWisk.pdf`, 2010. [Online; accessed 2013-10-23].

[44] Wei-Xi Huang and Hyung Jin Sung. Improvement of mass source/sink for an immersed boundary method. *International Journal for Numerical Methods in Fluids*, 53(11):1659–1671, 2007.

[45] J. Douglas Jr. Alternating direction methods for three space variables. *Numer. Math.*, 4:41–63, 1962.

[46] Seongwon Kang, Gianluca Iaccarino, Frank Ham, and Parviz Moin. Prediction of wall-pressure fluctuation in turbulent flows with an immersed boundary method. *J. Comput. Phys.*, 228(18):6753–6772, October 2009.

[47] K. Khadra, Ph. Angot, S. Parneix, and J.-P. Caltagirone. Fictitious region approach for numerical modelling of NavierStokes equations. *International Journal for Numerical Methods in Fluids*, page 651684, 2000.

[48] Khodor Khadra, Philippe Angot, Sacha Parneix, and Jean-Paul Caltagirone. Fictitious domain approach for numerical modelling of navierstokes equations. *International Journal for Numerical Methods in Fluids*, 34(8):651–684, 2000.

[49] J. Kim and P. Moin. Application of a fractional-step method to incompressible Navier-Stokes equations. *J. Comput. Phys.*, 59:308–323, 1985.

[50] Jungwoo Kim, Dongjoo Kim, and Haecheon Choi. An immersed-boundary finite-volume method for simulations of flow in complex geometries. *Journal of Computational Physics*, 171(1):132 – 150, 2001.

[51] M.P. Kirkpatrick, S.W. Armeld, and J.H. Kent. A representation of curved boundaries for the solution of the navier-stokes equations on a staggered three-dimensional cartesian grid. *Journal of Computational Physics*, (184(1)):136, 2003.

[52] Murata K.K., Williams D.C., Griffith R.O. Tills J., Gido R.G., Tadios E.L., Davis F.J., Martinez G.M., and Washington K.E. Code Manual for CONTAIN 2.0: A Computer Code for Nuclear Reactor Containment Analysis. *NUREG/CR-65, SAND97-1735, Sandia National Laboratories, USA*, 1997.

[53] Klein-Hessling, S W. Arndt, and G. Weber. COCOSYS user manual. *Programmdokumentation GRS-P-3/1, Gesellschaft fuer Anlagen- und Reaktorsicherheit (GRS) mbh, Koeln*, 2000.

[54] W. Klein-Hessling. Benchmark Analysis With COCOSYS. COCOSYS Calculations for Benchmark Exercise 1. Appendix F., 2002.

[55] M. C. Lai and C. S. Peskin. An immersed-boundary method with formal second-order accuracy and reduced numerical viscosity. *Journal of Computational Physics*, 160, 200.

[56] L.D. Landau and E.M. Lifshitz. *Course of Theoretical Physics:Hydrodynamics*, volume 6. Nauka, Moskva, 1988.

[57] Chao-An Lin. Immersed boundary method based Lattice Boltzmann method to simulate 2d and 3d complex geometry flows. 2006.

[58] Andreas Mark and Berend G. M. van Wachem. Derivation and validation of a novel implicit second-order accurate immersed boundary method. *J. Comput. Phys.*, 227(13):6660–6680, June 2008.

[59] R. Mittal, C. Bonilla, and H.S. Udaykumar. Cartesian grid methods for simulating flows with moving boundaries. *Computational Methods and Experimental Measurements*, 2003.

[60] R. Mittal, H. Dong, M. Bozkurttas, F. M. Najjar, A. Vargas, and A. von Loebbecke. A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries. *J. Comput. Phys.*, 227(10):4825–4852, May 2008.

[61] R. Mittal and G. Iaccarino. Computational modeling and analysis of biomimetic flight mechanisms. *AIAA Pap.*, 37, 2002.

[62] R. Mittal and G. Iaccarino. Immersed boundary methods. *Ann. Rev. Fluid Mech*, 37:239261, 2005.

[63] R. Mittal, V. Seshadri, S. Sarma, and H.S. Udaykumar. Computational modeling of fluidic micro-handling processes. *In Tech. Proc. 5th Int. Conf. Model. Simul. Microsyst.*, page 38891, 2002.

[64] R. Mittal, V. Seshadri, and H.S. Udaykumar. Flutter, tumble and vortex induced autorotation. *Theor. Comput. Fluid Dyn*, 3(17), 2004.

[65] J. Mohd-Yusof. Combined Immersed-Boundary/B-spline methods for simulations of flow in complex geometries. Annual research briefs, Center for Turbulence Research, 1997.

[66] A. Oberbeck. Ueber die wrmeleitung der flssigkeiten bei bercksichtigung der strmungen infolge von temperaturdifferenzen. *Annalen der Physik*, 243(6):271–292, 1879.

[67] Dartzi Pan and Tzung-Tza Shen. Computation of incompressible flows with immersed bodies by a simple ghost cell method. *International Journal for Numerical Methods in Fluids*, 60(12):1378–1401, 2009.

[68] D. W. Peaceman and Rachford. The numerical solution of parabolic and elliptic differential equations. *Journal of the Society for Industrial and Applied Mathematics*, 3(1):pp. 28–41, 1955.

[69] Charles S. Peskin. Flow patterns around heart valves: A numerical method. *Journal of Computational Physics, 10(2):252271*, (10(2)):25227, 1972.

[70] Charles S. Peskin. Numerical analysis of blood flow in the heart. *Journal of Computational Physics*, 25(3):220 – 252, 1977.

[71] Charles S. Peskin. The fluid dynamics of heart valves: experimental, theoretical, and computation methods. *Annu. Rev. Fluid Mech.*, (14):235259, 1982.

[72] Charles S. Peskin and David M. McQueen. Modeling prosthetic heart valves for numerical analysis of blood flow in the heart. *Journal of Computational Physics*, 37(1):113 – 132, 1980.

[73] Charles S. Peskin and David M. McQueen. Computer-assisted design of pivoting disc prosthetic mitral valves. *The American journal of physiology*, 86(1):126–135, 1983.

[74] Charles S. Peskin and David M. McQueen. A three-dimensional computational method for blood flow in the heart i. immersed elastic fibers in a viscous incompressible fluid. *Journal of Computational Physics*, 81(2):372 – 405, 1989.

[75] Charles S. Peskin and David M. McQueen. A three-dimensional computational method for blood flow in the heart i. immersed elastic fibers in a viscous incompressible fluid. *Journal of Computational Physics*, 81(2):372 – 405, 1989.

[76] Charles S. Peskin, David M. McQueen, and E.L Yellin. Fluid dynamics of the mitral valve: physiological aspects of a mathematical model. *The American journal of physiology*, 242(6):1095–1110, 1982.

[77] Charles S. Peskin and Beth Feller Printz. Improved volume conservation in the computation of flows with immersed elastic boundaries. *Journal of Computational Physics*, 105(1):33 – 46, 1993.

[78] I. Petrowsky. New proof of the existence of a solution of Dirichlet's problem by the method of finite differences. pages 161–170, 1941.

[79] S.L. Randall. MELCOR 1.8.3 computer code users' guide. *Sandia National Laboratories, Albuquerque, MN87185, USA*, 1994.

[80] R. Rannacher. *On Chorin's projection method for the incompressible Navier-Stokes equations*. Lecture Notes in Mathematics, vol. 1530, 1992.

[81] Safety of Nuclear Power Reactors. `http://www.world-nuclear.org/info/Safety-and-Security/Safety-of-Plants/Safety-of-Nuclear-Power-Reactors`. [Online; accessed 2013-09-15].

[82] E. M. Saiki and S. Biringen. Numerical simulation of a cylinder in uniform flow: application of a virtual boundary method. *Journal of Computational Physics*, 123:450–465, 1996.

[83] A. Samarskii and A Vabishchevich. *Additive schemes for problems of Mathematical Physics (in Russian)*. Nauka, Moskva, 1999.

[84] Scipy: Python-based ecosystem of open-source software for mathematics, science, and engineering. [Online; accessed 2013-09-28].

[85] Jung Hee Seo and Rajat Mittal. A sharp-interface immersed boundary method with improved mass conservation and reduced spurious pressure oscillations. *Journal of Computational Physics*, (230):73477363, 2011.

[86] A.F. Shinn, M.A. Goodwin, and S.P. Vanka. Immersed boundary computations of shear- and buoyancy-driven flows in complex enclosures. *International Journal of Heat and Mass Transfer*, 52(1718):4082 – 4089, 2009. ¡ce:title¿Special Issue Honoring Professor D. Brian Spalding¡/ce:title¿.

[87] Sh. Smagulov. *Fictitious domain method for the Navier-Stokes equations (in russian)*. Preprint CS SO USSR, N 68, 1979.

[88] Becker Technologies. ThAI-Facility and Program. `http://www.becker-technologies.com/pdf/thai.pdf`. [Online; accessed 2013-01-11].

[89] R. Temam. Sur l'approximation de la solution des équations de Navier-Stokes par la méthode des pas fractionnaires ii. *Arch. Rat. Mech. Anal.*, 33:377–385, 1969.

[90] T.Gornak and O.Iliev. A goal oriented survey on immersed boundary methods. *ITWM-Bericht: Nr. 235*, 2013.

[91] Yu-Heng Tseng and Joel H. Ferziger. A ghost-cell immersed boundary method for flow in complex geometry. *Journal of Computational Physics*, 192(2):593 – 623, 2003.

[92] S. Turek and M. Schäfer. Benchmark computations of laminar flow around cylinder. In E. Hirschel, editor, *Flow Simulation with High-Performance Computers II*, volume 52 of *Notes on Numerical Fluid Mechanics*, pages 547–566. Vieweg, January 1996.

[93] H.S. Udaykumar, H.C. Kan, W. Shyy, and R. Tran-Son-Tay. Multiphase dynamics in arbitrary geometries on xed cartesian grids. *Journal of Computational Physics*, (137(2)):366405, 1997.

[94] H.S. Udaykumar, R. Mittal, and W. Shyy. Computation of solid-liquid phase fronts in the sharp interface limit on xed grids. *Journal of Computational Physics*, (153(2)):535574, 1999.

[95] P.N. Vabishchevich. *The Method of Fictitious Domains in Problems of Mathematical Physics (in russian)*. Izdatelstvo Moskovskogo Universiteta, 1991.

[96] Reinout vander Meulen. The immersed boundary method for the (2d) incompressible Navier-Stokes equations. `http://www.lr.tudelft.nl/fileadmin/Faculteit/`

`LR/Organisatie/Afdelingen_en_Leerstoelen/Afdeling_AEWE/Aerodynamics/Contributor_Area/Secretary/M._Sc._theses/doc/2006_1_01.pdf`, 2006. [Online; accessed 2013-10-24].

[97] N. N. Yanenko. *The method of fractional steps. The solution of problems of mathematical physics in several variables*. Springer-Verlag, New York, 1971.

[98] T. Ye, R. Mittal, H.S. Udaykumar, and W. Shyy. An accurate cartesian grid method for viscous incompressible flows with complex immersed boundaries. *Journal of Computational Physics*, (156(2)):209240, 1999.

[99] A. Zemitis, O.Iliev, T.Gornak, and B. Schmidtmann. Copool: Preprocessor's manual. `http://www.itwm.fraunhofer.de/fileadmin/ITWM-Media/Abteilungen/SMS/Pdf/CoPrepUserManual.pdf`, 2012. [Online; accessed 2013-08-21].

[100] A. Zemitis, V. Starikovichus, O.Iliev, and K. Steiner. CoPool: reduced 3D flow model and solution method. *Fraunhofer ITWM, Kaiserslautern*, 2010.

[101] F. Zhang. *The Schur Complement and Its Applications*. Numerical Methods and Algorithms. Springer, 2010.

Lebenslauf

1995 – 1998	Schule No. 1, Krasnogorsk, Russland.
1998 – 2003	Schule No. 4, Krasnogorsk, Russland.
2003 – 2005	Schule No. 79, Moskau, Russland.
2005 – 2010	Lomonossow-Universität, Fakultät für Rechenmathematik und Kybernetik, Moskau, Russland
2010 – 2011	Project studies in Advanced Technologies, Technische Universität Kaiserslautern, Kaiserslautern, Deutschland
Ab Feb. 2011	Doktorand bei Technische Universität Kaiserslautern/Fraunhofer Institut für Techno und Wirtschaftsmathematik (ITWM), Kaiserslautern, Deutschland