signature supervisor

TECHNISCHE
UNIVERSITÄT
WIEN
Vienna University of Technology

DIPLOMA THESIS

Space-time finite volume method in OpenFOAM®

executed to obtain the academic degree
Master of Science (MSc.) under the supervision of

Univ.Prof. Dipl.-Phys. Dr.-Ing. Andreas Otto

at the
Institute of Production Engineering and Photonic Technologies
Getreidemarkt 9, BA,
9th floor

submitted to the Vienna University of Technology
Faculty of Mechanical and Industrial Engineering

by

Tobias Florian, BSc

2. Februar 2023

signature student

Acknowledgements

I would like to express my sincerest gratitude to my thesis advisor, Prof. Andreas Otto, for his invaluable guidance, support and encouragements throughout the course of this project. I am grateful for his expertise and willingness to expand my scientific knowledge.

I would also like to thank the members of the Institute of Production Engineering and Photonic Technologies, for their valuable feedback and constructive suggestions. Special thanks are due to my co-advisor and friend, Constantin Zenz, whose ideas and insights significantly contributed to the outcome of this thesis.

Furthermore, I would like to thank Prof. Stefanie Elgeti to share her expertise and knowledge in space-time simulations.

Finally, I want to thank my girlfriend Vanessa, my family and friends for their steady assistance.

Thank you all for your support.

Declaration in Lieu of Oath

I hereby declare to be the sole author of this thesis and that no part of my work has been previously published or submitted for publication. I certify, to the best of my knowledge, that this thesis does not infringe upon anyone's copyright nor violate any other material from the work of others. All knowledge arising from external sources has been fully acknowledged below under standard referencing practice.

Tobias Florian, BSc

Vienna, 2. Februar 2023

Abstract

The interest in numerical simulations in various fields of science and engineering is rising steadily. Because of the broad community and the easy way to generate custom solvers, the open source CFD (computational fluid dynamics) software OpenFOAM® has taken a leading role in the development of finite volume codes. Generally, the software is designed to solve multi-physical field problems via the conservative finite volume method and, in case of non-stationary problems, is then proceeded in time via a finite difference scheme in order to receive a transient solution. In this thesis, we investigate the idea of a space-time finite volume formulation, where both space and time are simultaneously discretized using a finite volume approach. Apart from the benefit of a fully conservative formulation, the big advantages of such an approach are the possibility for local refinement in both space and time and the convenient treatment of moving boundaries.

Within the scope of the thesis, this method is deduced, some basic OpenFOAM® solvers are transformed into the space-time formulation, and various one- and two-dimensional test cases are then validated and compared to the original solvers.

Kurzfassung

Das Interesse an numerischen Simulationen in verschiedenen Bereichen der Wissenschaft und Technik nimmt stetig zu. Aufgrund der breiten Community und der einfachen Handhabung, eigene Solver zu erstellen, hat die Open-Source-CFD-Software OpenFOAM® eine führende Rolle bei der Entwicklung von Finite-Volumen-Codes übernommen. Generell ist die Software zur Lösung multiphysikalischer Feldprobleme unter Verwendung der konservativen Finiten-Volumen-Methode ausgelegt. Im Falle nicht-stationärer Probleme erfolgt dann die zeitliche Entwicklung mit Hilfe eines Finiten-Differenzen-Schemas. In dieser Arbeit untersuchen wir die Idee einer Raum-Zeit-Finite-Volumen-Formulierung, bei der sowohl die räumliche als auch die zeitliche Ausdehnung der Simulation gleichzeitig mit Hilfe eines Finite-Volumen-Ansatzes diskretisiert werden. Neben dem Vorteil einer vollständig konservativen Formulierung, sind die großen Vorzüge eines solchen Ansatzes die Möglichkeit der lokalen Netzverfeinerung sowohl in Raum als auch in Zeit und die praktische Behandlung von sich bewegenden Randbedingungen.

Im Rahmen der vorliegenden Arbeit wird diese Methode hergeleitet, einige grundlegende OpenFOAM® Löser in die Raum-Zeit-Formulierung transformiert, mit verschiedenen ein- und zwei-dimensionalen Testfällen validiert und mit den ursprünglichen Solvern verglichen.

List of Figures

Nomenclature

$\mathbf{I}$	Identity matrix
$\boldsymbol{\Psi}$	Vectorial or scalar quantity
$\boldsymbol{\Psi}_{\mathrm{D}}$	Downwind cell value of quantity $\boldsymbol{\Psi}$
$\boldsymbol{\Psi}_{\mathrm{f}}$	Face value of quantity $\boldsymbol{\Psi}$
$\boldsymbol{\Psi}_{\mathrm{N}}$	Neighbour cell value of quantity $\boldsymbol{\Psi}$
$\boldsymbol{\Psi}_{\mathrm{P}}$	Owner cell value of quantity $\boldsymbol{\Psi}$
$\boldsymbol{\Psi}_{\mathrm{U}}$	Upwind cell value of quantity $\boldsymbol{\Psi}$
$\boldsymbol{\sigma}$	Stress tensor
$\boldsymbol{\tau}$	Shear stress tensor
$\boldsymbol{b}$	Body force per unit mass
$\boldsymbol{S}^*$	Space surface area vector
$\boldsymbol{S}_f$	Surface area vector
$\boldsymbol{S}_s$	Space surface area vector
$\boldsymbol{S}_t$	Time surface area vector
$\boldsymbol{u}$	Velocity
$\boldsymbol{u}'$	Temporary velocity not fulfilling continuity equation
$\boldsymbol{u}^*$	Space-Time velocity
$\boldsymbol{u}_f$	Face velocity

Δ	Laplace operator
Γ	Diffusion tensor or scalar diffusion coefficient
γ^*	Space-Time tensor
κ	Thermal conductivity
$\mathbf{A}$	Coefficient matrix
$\mathbf{b}$	Source vector
$\mathbf{C_f}$	Face center
$\mathbf{C}$	Cell center
$\mathbf{n_f}$	Face unit normal vector
$\mathbf{n}$	Normal vector
$\mathbf{n}^*$	Space-Time normal vector
$\mathbf{n}_s$	Space normal vector
$\mathbf{n}_t$	Time normal vector
$\mathbf{S_f}$	Face area vector
μ	Dynamic viscosity
∇	Nabla operator
∇_n	Normal gradient
∇^*	Space-Time nabla operator
ν	Kinematic viscosity
ϕ_f	Volumetric face flux
ϕ_f	Volumetric flux
ρ	Density

$a_{i,j}$	Matrix coefficients
b_i	Source vector coefficients
Co	Courant number
F	0uter force
p	Pressure
Re	Reynold's number
T	Temperature
w	Interpolation weighting factor

Contents

«People like us who believe in physics know that the distinction between past, present and future is only a stubbornly persistent illusion. Time, in other words, is an illusion.»

— Albert Einstein

1 Introduction

1.1 Motivation

Numerical simulations have become a well-established method to investigate physical phenomena and behaviours in various fields of science and engineering. The enormous progress in computer science and processing power brings along the scientists' hunger to simulate bigger domains on the one hand and more accurately, on the other hand. The most commonly used methods are the finite difference, the finite element and the finite volume method, each, having its advantages and disadvantages. Because of its conservation property, the latter is suited very well for predicting fluid flows and related problems. The overall aim of the method is to transform physical partial differential equations into algebraic equations, which the computer can solve and display. Usually, the course of action has been to divide the spatial domain into finite control volumes, then, integrate the underlying equations over these control volumes and subsequently apply Gauss' theorem in order to transform volume integrals into surface integrals, which can then be approximated using suitable numerical schemes, such as upwind scheme, midpoint scheme, and so on. If the problem is time-dependent, the time derivative is discretized via a finite difference method such as Euler or Runge-Kutta of arbitrary order. This procedure leads to one algebraic equation per control volume, per differential equation, per time step. In order to follow the above steps from meshing all the way to solving the huge systems of equations, the open source CFD software OpenFOAM® provides all the necessary functionalities and different solvers for a broad range of physical problems. Furthermore, the C++-based code enables editing and generating custom solvers.

The aim of this work is to use the OpenFOAM® functionalities and transform several solvers into space-time finite volume solvers. The basic idea of this approach is to not only discretize the spatial domain into finite volumes, but rather divide the whole space-time domain into space-time finite volumes. To clarify the idea, imagine a one-dimensional bar along the x-axis with a heat peak in the middle that transiently diffuses from the middle to the side ends. The classical way is to discretize the bar along the x-axis and process time step per time step. The

space-time analogy discretizes the bar along the x-axis, the time along the t-axis and by pulling the time derivative into the first spatial derivative then solves the transient problem as one mathematically static problem. In doing so, we increase the computational dimension of the problem by one, as in the example case, the 1D setup is transformed into a 2D ($\widehat{=}$1D+t) problem. The advantages of this method are its fully conservative behaviour, its treatment with moving boundaries and its ability to have local mesh refinement in space and time. The local refinement in space and time potentially makes large-scale simulations on clusters more feasible, since also parallelisation will be enabled in both, space and time.

The following investigates the space-time finite volume approach to solve fluid dynamical problems and the suitability of OpenFOAM® to serve as a base program to transform existing finite volume code into space-time.

The thesis is organized as follows. After an overview of the state of the art in Sec. 1.2, the second chapter covers the theoretical part, more precisely, the description of the conventional and the space-time finite volume method, the derivation and transformation of fundamental equations into space-time and an explanation of the time slab method in OpenFOAM®. Within the third chapter, the space-time OpenFOAM® implementations are validated by means of some 1D+t and 2D+t test cases. Herein, the possibility to deal with moving boundary conditions and also local mesh refinement is shown. The fourth chapter summarises the results and finally, the fifth chapter provides a critical discussion and an outlook on future studies and reveals what needs to be done to drive the method forward.

1.2 State of the art

The space-time approach solving fluid flow problems appears to be first linked with finite element methods. Therein, also higher order schemes are introduced in order to increase numerical accuracy. The approach also takes root in the DG (discontinuous Galerkin) community. In general, the approach is applied more widely in conjunction with finite elements than with finite differences or finite volumes. Zwart [13] derives the integrated space-time finite volume method and calculates free surface problems. Rendall *et al.* [12] uses the technique to perform aerodynamic, store separation and rotor simulations in 2D+t. Herein, the moving boundaries are in the foreground. Since the space-time finite volume description brings along conservation in space and time, there is no need to apply the Leibnitz integration rule[13]. In this respect, conventional finite volume solvers incorporate the multiple reference frame (MRF) method [9], the immersed boundary conditions [6] or, using the *Arbitrary Lagrangian-Eulerian* method, have to implement

geometrical conservation rules [3]. Those methods demand special treatments of the underlying equations.

Indeed, space-time simulations are still only performed up to order 2D+t using a 3D mesh. Advancing the method to 3D+t is currently under development, but to date, there are no 4D mesh generators available. [12]

In this work, the 3D mesh generating tool that comes with OpenFOAM® is used to execute 1D+t and 2D+t simulations. Also, the mesh refinement libraries can be employed one-to-one.

2 Theory

This section covers the theoretical background of this work. It starts with an introduction of the fundamental equations needed to solve the test cases, such as the *incompressible* Navier-Stokes equations and the passive scalar transport equation. The latter is then used to derive the classical finite volume method. Afterwards, the space-time finite volume approach is presented. This starts with an illustration of a space-time grid and continues with the full description of the space-time finite volume method.

2.1 Governing equations

2.1.1 Scalar transport equation

In fluid dynamics, most processes are described by equations of the type

$$\underbrace{\frac{\partial \boldsymbol{\Psi}}{\partial t}}_{\text{temporal rate of change}} + \underbrace{\nabla \cdot (\boldsymbol{u}\boldsymbol{\Psi})}_{\text{convection}} = \underbrace{\Gamma \nabla^2 \boldsymbol{\Psi}}_{\text{diffusion}} + \underbrace{F}_{\text{sources}}, \tag{2.1}$$

the general transport equation of any quantity $\boldsymbol{\Psi}$. Considering the temperature T as the underlying quantity, Eqn. 2.1 becomes the heat transport equation

$$\underbrace{\frac{\partial T}{\partial t}}_{\text{temporal rate of change}} + \underbrace{\boldsymbol{u} \cdot \nabla T}_{\text{convection}} = \underbrace{\kappa \nabla^2 T}_{\text{diffusion}} + \underbrace{F}_{\text{heat sources/sinks}}, \tag{2.2}$$

that specifies the evolution of the temperature at all points $\boldsymbol{x}$ of a finite domain at all times $t > t_0$ based on an initial condition $T_0(\boldsymbol{x}) = T_0(\boldsymbol{x}, t_0)$. Therein, $\frac{\partial T}{\partial t}$ provides the temporal change of the temperature at a fixed position $\boldsymbol{x}$ at a certain time t. The next term, $\boldsymbol{u} \cdot \nabla T$ gives the negative rate of change due to convection with a prescribed velocity $\boldsymbol{u}$. The first term on the *RHS*, $\kappa \nabla^2 T$, is called the diffusion term with κ being the thermal diffusivity. By means of the examined

example, it balances the local temperature differences. Eventually, the last term $F(\boldsymbol{x}, t)$ describes sources and sinks of the present quantity T.[7]

In case the temperature field does not affect the convection velocity field $\boldsymbol{u}$, (2.2) is called a passive scalar transport equation. The respective OpenFOAM® solver which handles the problem and is later transferred into space-time is called *scalarTransportFoam*[11]. In Sec. 3, this solver will be used to test pure diffusion, pure advection and advection-diffusion in a prescribed velocity setup.

2.1.2 Navier-Stokes equation

Typically, the velocity field is not known *a priori* and is, in its most simplified version, described by the famous Navier-Stokes equation for Newtonian, *incompressible* fluids

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\frac{1}{\rho} \nabla p + \nu \nabla^2 \boldsymbol{u} + \boldsymbol{F} \tag{2.3}$$

which is by itself a transport equation. The difference to (2.2) is that now, the momentum per mass, $\boldsymbol{u}$, which is a *vectorial* quantity, is transported. In (2.3) the convective term $\boldsymbol{u} \cdot \nabla \boldsymbol{u}$ brings along a non-linearity and with that, a number of numerical challenges.[7]

The following shows the derivation steps to yield (2.3) and which additional equations and simplifications are necessary in order to solve for the flow field $\boldsymbol{u}$. Additionally, the PISO[1] algorithm is shortly explained, which is implemented in the OpenFOAM® solver *icoFoam*[5] and transformed into space-time in Sec. 3.

Derivation of the Navier-Stokes equation

The conservation of momentum can be written as

$$\frac{\partial \rho \boldsymbol{u}}{\partial t} + \nabla \cdot (\rho \boldsymbol{u} \boldsymbol{u}) = \nabla \cdot \boldsymbol{\sigma} + \rho \boldsymbol{b} \tag{2.4}$$

where $\boldsymbol{b}$ is any body force per unit mass, for instance the gravitational force, and $\boldsymbol{\sigma}$ represents the symmetric stress tensor. Let us have a closer look at the term $\nabla \cdot \boldsymbol{\sigma}$ describing forces within a fluid. This is the point where Newton's fluid model[2] takes action, which states the linear relation between shear stress τ_{xy} and the

[1] *Pressure-Implicit with Splitting of Operators*
[2] or linear viscosity model

velocity gradient, by multiplication of the dynamic viscosity μ. For *incompressible* fluids, it can be written as [7]

$$\boldsymbol{\tau} = \mu \left[\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^{\mathrm{T}}\right] \tag{2.5}$$

and

$$\boldsymbol{\sigma} = \boldsymbol{\tau} - p\mathbf{I}. \tag{2.6}$$

Substitution into 2.4 yields

$$\begin{aligned} \frac{\partial \rho \boldsymbol{u}}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}\boldsymbol{u}) - \nabla \cdot (\mu \nabla \boldsymbol{u}) \\ - \nabla \cdot \left[\mu (\nabla \boldsymbol{u})^{\mathrm{T}}\right] = -\nabla p + \rho \mathbf{b}. \end{aligned} \tag{2.7}$$

Further, we denote the conservation of mass in its general form

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = 0. \tag{2.8}$$

Assuming an *incompressible* fluid, *i.e.* $\rho = constant$, it reduces to

$$\nabla \cdot \boldsymbol{u} = 0. \tag{2.9}$$

With that, $\nu = \mu/\rho$ being the kinematic viscosity, p being the kinematic pressure and the identity

$$\nabla \cdot \left[\nu (\nabla \boldsymbol{u})^{\mathrm{T}}\right] \equiv \nu \nabla (\nabla \cdot \boldsymbol{u}) + (\nabla \nu) \cdot (\nabla \boldsymbol{u}), \tag{2.10}$$

Eqn. 2.7 can be written as the Navier-Stokes equation for homogeneous, *incompressible*, Newtonian fluids

$$\begin{aligned} \frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot (\boldsymbol{u}\boldsymbol{u}) - \nabla \cdot (\nu \nabla \boldsymbol{u}) \\ - (\nabla \nu) \cdot (\nabla \boldsymbol{u}) = -\nabla p + \mathbf{b}. \end{aligned} \tag{2.11}$$

Thus, we end up with an equation with one unknown vector field $\boldsymbol{u}$ and one unknown scalar field p. In order to yield a solvable system, we use another identity

$$\nabla \cdot (\nabla \cdot \nabla \boldsymbol{u}) \equiv \nabla^2 (\nabla \cdot \boldsymbol{u}), \tag{2.12}$$

plug (2.9) into (2.11), apply the divergence operator and receive an equation for the pressure

$$\nabla^2 p + \nabla \cdot [\nabla \cdot (\boldsymbol{u}\boldsymbol{u})] = 0. \tag{2.13}$$

For the sake of simplicity, the body force per unit mass $\mathbf{b}$ is assumed to be divergence free and time independent. In this *incompressible* setup, the conservation of mass is a steady equation, and thus, it can be interpreted as a constraint to the flow field.[4]

PISO algorithm

Since it is intended to solve transient problems with the space-time approach, the PISO algorithm is shortly explained in this section and transformed into space-time in Sec.3. This algorithm is well suited for solving homogeneous, *incompressible*, transient flow fields. For the sake of simplicity, no thermal coupling, no outer forces and constant viscosity is assumed. Therefore, we recall the previously derived and simplified momentum equation and pressure equation

$$\text{I:} \quad \frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot (\boldsymbol{u}\boldsymbol{u}) - \nabla \cdot (\nu \nabla \boldsymbol{u}) = -\nabla p. \tag{2.14}$$

and

$$\text{II:} \quad \nabla^2 p + \nabla \cdot [\nabla \cdot (\boldsymbol{u}\boldsymbol{u})] = 0. \tag{2.15}$$

The aim is to evaluate the vector field $\boldsymbol{u}$ and the pressure field p for every time step $n+1$. The PISO algorithm belongs to the family of pressure-velocity coupling algorithms. Thereby, the basic idea is that at the beginning of each time step $n+1$, the momentum equation I is solved for $\boldsymbol{u}'$ with p^n and the one $\boldsymbol{u}^n$ of the non-linear term $\nabla \cdot (\boldsymbol{u}\boldsymbol{u})$ from the previous time step n. The $\boldsymbol{u}'$ denotes, that this solution does, in general, not fulfill the continuity equation (2.9) anymore. Therefore, $\boldsymbol{u}'$ is substituted into the pressure equation II, which is then solved for p. This p is then used to update $\boldsymbol{u}'$ in order to match the continuity equation (2.9) again. Now, let us have a closer look at the equations. For the sake of clarity, we will use the following notation in order to rewrite for example I, without the pressure gradient

$$\boldsymbol{A} \cdot \boldsymbol{u} \equiv A\boldsymbol{u} - \mathbf{H}(\boldsymbol{u}) \equiv \frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot (\boldsymbol{u}\boldsymbol{u}) - \nabla \cdot (\nu \nabla \boldsymbol{u}), \tag{2.16}$$

where $A\boldsymbol{u}$ is linear in $\boldsymbol{u}$ and $\mathbf{H}(\boldsymbol{u})$ denotes a function of $\boldsymbol{u}$. This way, the momentum equation can be written as

$$\boldsymbol{A} \cdot \boldsymbol{u} = -\nabla p, \tag{2.17}$$

and is well known as the momentum predictor. The pressure equation becomes

$$\nabla \cdot \frac{1}{A} \nabla p = \nabla \cdot \left[\frac{\mathbf{H}(\boldsymbol{u})}{A} \right] \tag{2.18}$$

and the equation, that corrects $\boldsymbol{u}'$ (*momentum corrector*) becomes

$$\boldsymbol{u} := \frac{\mathbf{H}(\boldsymbol{u})}{A} - \frac{1}{A} \nabla p. \tag{2.19}$$

The essence of the PISO algorithm is that after the momentum corrector, before heading to the next time step, the pressure equation (2.18) is again solved with

the corrected momentum of (2.17)[4]. This procedure can be executed several times. Now, since the pressure, and thus, the pressure gradient has changed, the momentum equation is not fulfilled anymore. Hence, we could start the whole procedure again at the momentum predictor, now with a better guess of $\boldsymbol{u}$ and p - this can be called the outer iteration loop, which can also be seen in Fig. 2.1. Once $\boldsymbol{u}$ and p converge, the next time step can be started.[7]

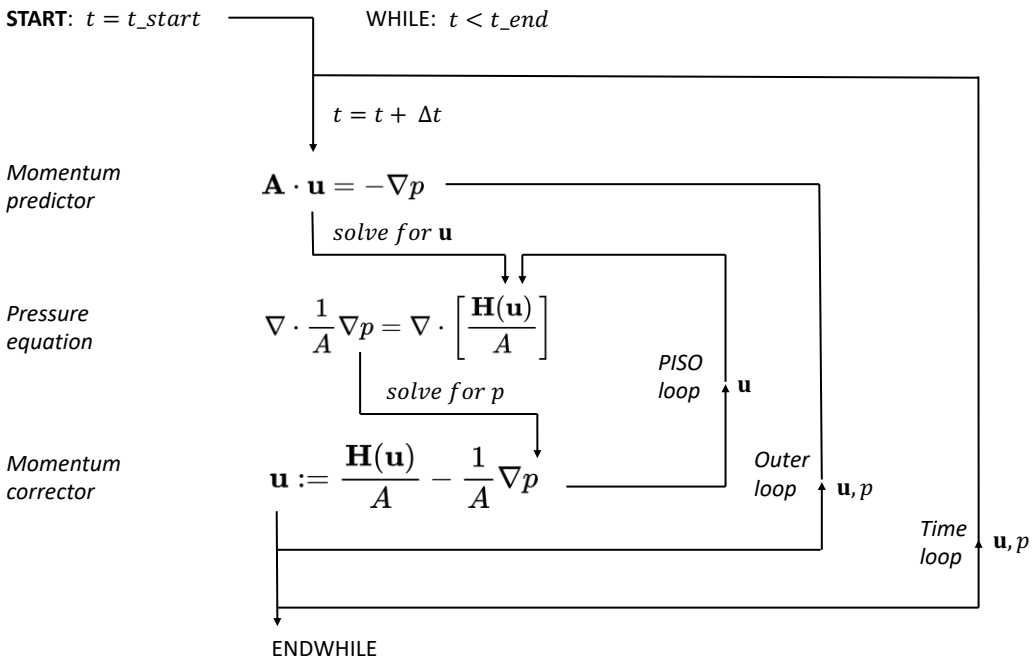


Figure 2.1: Solution algorithm (derived from [4])

In most cases, having small enough time steps, proceeding the outer loop once and the PISO loop twice delivers sufficient accuracy[7].

2.2 Classical finite volume formulation

As already mentioned in the introduction, the overall goal of the finite volume method is to transform algebraic equations, mainly partial differential equations, into linear systems of equations, which the computer can solve. Having derived the governing equations to solve for example a fluid flow in the previous section,

the following describes how the different terms are treated and transformed into solvable systems of equations. The derivation is mainly based on the ideas of Greenshields and Weller [4].

2.2.1 General concept

As relating methods like the finite element method or finite difference method, also the finite volume method attempts to represent continuous physical phenomena by discrete values. Thereby, time is divided into time intervals Δt, the physical space is split into finite volumes, which take discrete values of quantities like velocity $\boldsymbol{u}$ or pressure p, and algebraic equations become sets of linear equations. The following image (Fig.2.2) summarizes the general concept.[4]

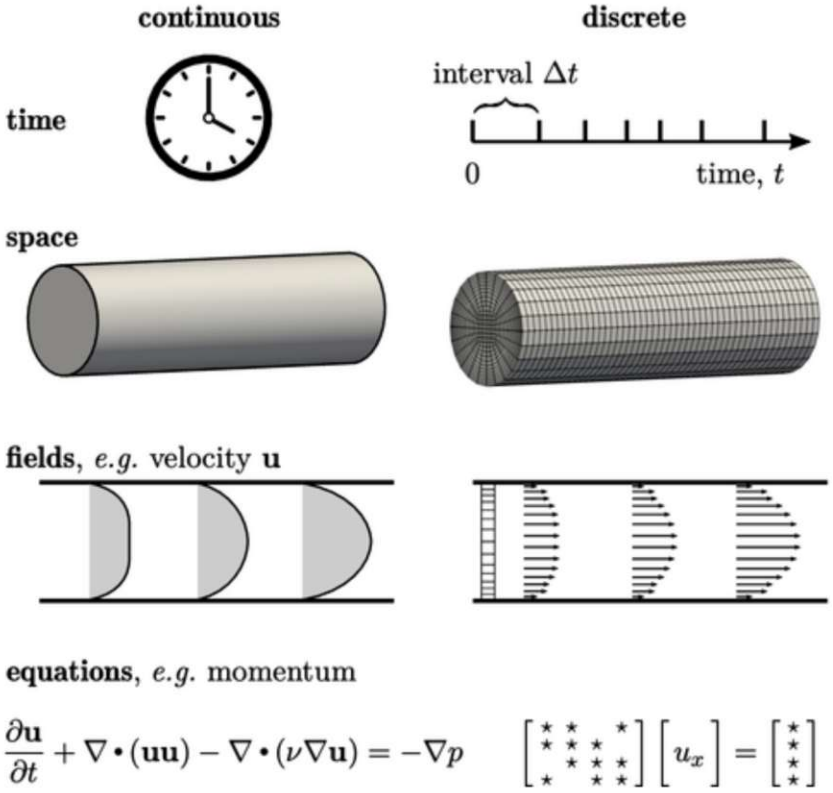


Figure 2.2: Finite volume concept [4]

2.2.2 Finite volume mesh

In Fig. 2.2, the second row "space", already shows how a solution domain, here a pipe, can be split into cells or finite volumes. These can take various shapes, such as in the figure below (Fig.2.3). Therein, the entities cell volume $\mathbf{V}$, face area vector $\mathbf{S_f}$, face unit normal vector $\mathbf{n_f} = \mathbf{S_f}/|\mathbf{S_f}|$ and the relation $d\mathbf{S_f} = \mathbf{n_f}dS$ will play a huge role, as the finite volume method deals with volume and surface integrals and assigns quantities like pressure p to cell centers $\mathbf{C}$ and face centers $\mathbf{C_f}$.[4]

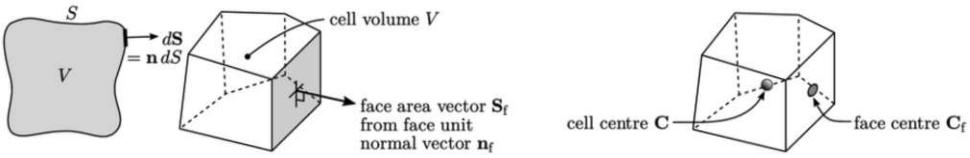


Figure 2.3: Finite volume [4]

2.2.3 Equation discretisation and matrix construction

Discretisation is the process that transforms continuous differential equations into sets of linear equations. For example, let us assume a clip of a mesh containing N cells (Fig. 2.4). Thus, each cell holds a discrete value, *e.g.* pressure p_i.

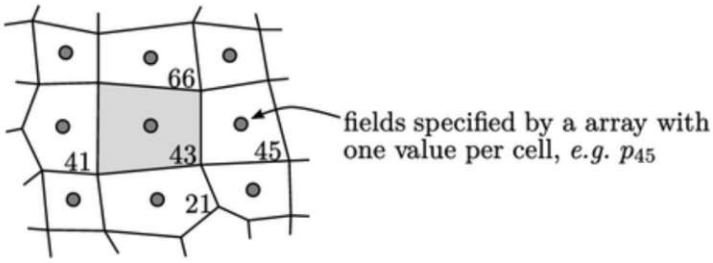


Figure 2.4: Clip of mesh [4]

In particular, discretisation of the p-equation for the exemplary cell 43 might yield following equation

$$a_{43,21}p_{21} + a_{43,41}p_{41} + \mathbf{a_{43}}_{,43}\mathbf{p}_{43} + a_{43,45}p_{45} + a_{43,66}p_{66} = b_{43}. \tag{2.20}$$

This way, we obtain one linear equation per cell (*e.g.* cell number 43), per equation to solve(*e.g.* the p-equation). Herein, $a_{i,j}$ correspond to coefficients derived from discretisation schemes and b_i are coefficients corresponding to sources. By sorting the equations, the system of equations can be rewritten as

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,N} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,N} \\ a_{3,1} & a_{3,2} & a_{3,3} & \cdots & a_{3,N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{N,1} & a_{N,2} & a_{N,3} & \cdots & a_{N,N} \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ \vdots \\ p_N \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_N \end{bmatrix} \tag{2.21}$$

or

$$[\mathbf{A}][p] = [\mathbf{b}] \tag{2.22}$$

where the coefficient matrix $\mathbf{A}$ becomes a sparse matrix since for each diagonal entry $a_{i,i}$ only neighbour cells contribute a non-zero entry. The solution method for the matrix equations of type (2.22) can be chosen freely, for example *Gauss-Seidel* or *Conjugate Gradients etc.* and is not under investigation in this work. The evaluation of the coefficients is dependent on the differential equation, the boundary conditions and on the numerical schemes. Fig.2.5 , for example, shows how the different terms in the transport equation of a quantity $\mathbf{\Psi}$ build up the coefficient matrix and the source vector, where the stars illustrate a non-zero entry.[4]

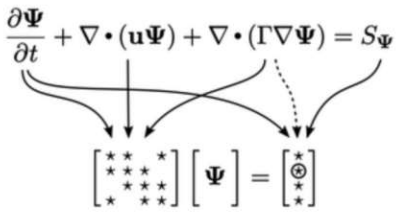


Figure 2.5: Matrix construction of transport equation [4]

The following examines discretisation of

1. Laplacian term
2. advection term
3. gradient
4. source term
5. time derivative

In general, the course of action is to integrate the underlying partial differential equation over the finite volumes to receive volume integrals. Subsequently, apply *Gauss divergence theorem* to transform some volume integrals into surface integrals, *i.s.* all the integrals containing the divergence operator, *e.g.* the Laplacian term $\nabla \cdot (\Gamma \nabla \mathbf{\Psi})$ or the convective term $\nabla \cdot (\mathbf{u}\mathbf{\Psi})$ in a transport equation. Subsequently, these surface integrals can be approximated by a summation over the faces of one finite volume, *s.t.*

$$\int_S (d\mathbf{S} \cdot \mathbf{\Psi}) \rightarrow \sum_{\mathrm{f}} \mathbf{S}_{\mathrm{f}} \cdot \mathbf{\Psi}_{\mathrm{f}}, \tag{2.23}$$

with $d\mathbf{S} = \mathbf{n}dS$. Since the surface vector $\mathbf{S}_{\mathrm{f}}$ is known from the mesh information, $\mathbf{\Psi}_{\mathrm{f}}$ between two cells still has to be computed via an interpolation scheme.[4]

Discretisation of Laplacian term

Let us start with the Laplacian term $\nabla \cdot (\Gamma \nabla \mathbf{\Psi})$ of the transport equation in Fig. 2.5. Integrating over the finite volume and applying *Gauss' divergence theorem* yields

$$\int_V \nabla \cdot (\Gamma \nabla \mathbf{\Psi}) dV = \int_S (\Gamma \nabla_{\mathrm{n}} \mathbf{\Psi}) d\mathbf{S} \rightarrow \sum_{\mathrm{f}} |\mathbf{S}_{\mathrm{f}}| \Gamma_{\mathrm{f}} \nabla_{\mathrm{n}} \mathbf{\Psi}_{\mathrm{f}}, \tag{2.24}$$

with known mesh data $\mathbf{S}_{\mathrm{f}}$ and yet to obtain diffusivity at faces Γ_{f} and surface normal gradient ∇_{n}. As previously described, field values like Γ or $\mathbf{\Psi}$ are stored at cell centers, whereas quantities with subscript f have to be determined at cell faces between two cells according to

$$\mathbf{\Psi}_{\mathrm{f}} = w\mathbf{\Psi}_{\mathrm{P}} + (1 - w)\mathbf{\Psi}_{\mathrm{N}}, \tag{2.25}$$

with P, N and w denoting the owner cell, the neighbour cell and the weighting factor, respectively. The most commonly used and most intuitive interpolation scheme is linear interpolation with weighting factors

$$w = \frac{d_{\mathrm{fN}}}{d_{\mathrm{PN}}} = \frac{\mathbf{n} \cdot \mathbf{d}_{\mathrm{N}}}{\mathbf{n} \cdot (\mathbf{d}_{\mathrm{P}} + \mathbf{d}_{\mathrm{N}})} \tag{2.26}$$

according to Fig. 2.6.

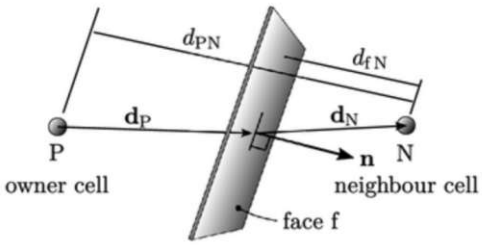


Figure 2.6: Linear interpolation [4]

Further, the surface normal gradient ∇_n can be discretised as

$$\nabla_n \Psi_f = \frac{(\Psi_N - \Psi_P)}{|\Delta d|}, \tag{2.27}$$

according to Fig. 2.7, for orthogonal meshes, *i.e.* $\theta = 0$.[4]

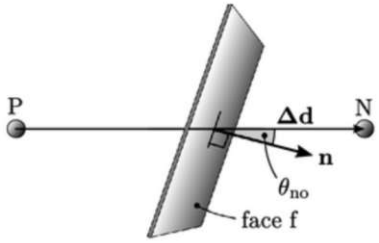


Figure 2.7: Surface normal gradient ∇_n [4]

Discretisation of advection term

Now, let us examine the discretisation of the advection term $\nabla \cdot (\boldsymbol{u}\Psi)$ of the transport equation in Fig. 2.5. Again, integrating over the finite volumes and applying *Gauss' divergence* theorem, gives

$$\int_V \nabla \cdot (\boldsymbol{u}\Psi) dV = \int_S \boldsymbol{u}\Psi \cdot d\mathbf{S} \rightarrow \sum_f \mathbf{S}_f \cdot \boldsymbol{u}_f \Psi_f \rightarrow \sum_f \phi_f \Psi_f \tag{2.28}$$

with the volumetric flux $\phi_f = \mathbf{S}_f \cdot \boldsymbol{u}_f$, where $\boldsymbol{u}_f$ is conventionally obtained by face interpolation of $\boldsymbol{u}$. The critical, *i.e.* the advected quantity Ψ_f is where advection schemes come into play. There exists a broad number of advection schemes characterised by boundedness/unboundedness, first or higher order accuracy and

different numerical stability. In this work, only the first and second order upwind scheme and the linear interpolation scheme are shortly discussed based on Fig. 2.6 Therein, the determination of the desired face quantity $\boldsymbol{\Psi}_\mathrm{f}$ is dependent on the flow direction. Recalling the linear interpolation scheme, already discussed above, defines $\boldsymbol{\Psi}_\mathrm{f}$ as a linear combination of the upwind cell value $\boldsymbol{\Psi}_\mathrm{U}$ and the downwind cell value $\boldsymbol{\Psi}_\mathrm{D}$, whereas the upwind scheme specifies $\boldsymbol{\Psi}_\mathrm{f}$ only based on the upwind cell value $\boldsymbol{\Psi}_\mathrm{U}$. It can be shown that the linear interpolation scheme is second order accurate but yields unbounded solutions, whereas the upwind scheme is only first order accurate but yields bounded results, if $\nabla \cdot \boldsymbol{u} = 0$. The most significant drawback of the upwind scheme is its diffusive character. In order to see that, let us state the *Taylor's series expansion*

$$\boldsymbol{\Psi}(x + \Delta x) = \boldsymbol{\Psi}(x) + \frac{\partial \boldsymbol{\Psi}}{\partial x}\Delta x + \frac{\partial^2 \boldsymbol{\Psi}}{\partial x^2}\frac{\Delta x^2}{2} + \dots . \tag{2.29}$$

In an 1D setup, application of the upwind scheme yields

$$\frac{\boldsymbol{\Psi}_\mathrm{P} - \boldsymbol{\Psi}_\mathrm{U}}{\Delta x} = \frac{\partial \boldsymbol{\Psi}}{\partial x} + \frac{\partial^2 \boldsymbol{\Psi}}{\partial x^2}\frac{\Delta x}{2} + \dots \tag{2.30}$$

which implies that the upwind scheme diffuses $\boldsymbol{\Psi}$ with $\frac{\partial^2 \boldsymbol{\Psi}}{\partial x^2}\frac{\Delta x}{2}$, *i.e.* a Laplacian term with diffusion coefficient $\frac{\Delta x}{2}$. The numerical diffusion is particularly high when the advection direction is not perpendicular to the cell faces. The extreme case, as shown in Fig. 2.8, where we have a 2D setup, a square mesh, advection velocity inclination of 45° and $\boldsymbol{\Psi} = 1$ at the left boundary and $\boldsymbol{\Psi} = 0$ at the bottom boundary. The dashed line clearly reveals the diffusion of the step function, which is the analytical solution in this case.[4]

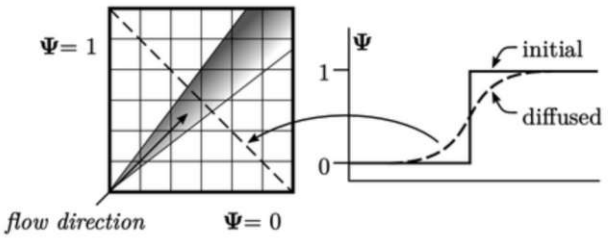


Figure 2.8: Numerical diffusion [4]

Concerning this issue, the second order linear upwind scheme provides remedy. The essence of the scheme is to additionally take the gradient $\nabla \boldsymbol{u}$ of the upwind cell into account. Thus, the upwind cell value $\boldsymbol{\Psi}_\mathrm{U}$ is interpolated to the face in terms of the upwind cell gradient $\nabla \boldsymbol{u}$ and the vector $\mathrm{d_p}$ from the upwind cell

center to the face center according to Fig.2.6. As shown in Fig. 2.9, this method contributes the face value $\mathbf{\Psi}_\mathrm{f}$ represented by $\mathbf{\Psi}_\mathrm{U}$ to the system matrix $[\mathbf{A}]$ (see Eqn. 2.22) and the extrapolation $\mathrm{d_p} \cdot \nabla \boldsymbol{u}$ *explicitly* to the *RHS* $[\mathbf{b}]$.[4]

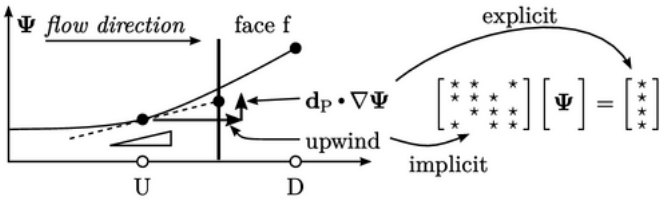


Figure 2.9: Linear upwind scheme [4]

Discretisation of gradients

Discretisation of gradients, like for example the pressure gradient on the *RHS* of the momentum equation (2.7), follows a similar procedure:

$$\int_V \nabla \mathbf{\Psi} dv = \int_S (\mathbf{\Psi} d\mathbf{S}) \rightarrow \sum_\mathrm{f} \mathbf{S}_\mathrm{f} \mathbf{\Psi}_\mathrm{f} \tag{2.31}$$

Where $\mathbf{\Psi}_\mathrm{f}$ can be evaluated by a linear scheme like in Fig.2.6 or *e.g.* by a least square method.[4] Indeed, only linear interpolation is used throughout all simulations in this work.

Discretisation of source terms

Source terms like $S_\mathbf{\Psi}$ in the transport equation in Fig. 2.5 remain a volume integral after integrating over the finite volume and contribute to the *RHS* in the governing system matrix.

Discretisation of time derivative

The last term to be discretised is the time derivative. This will be the crucial difference compared to the space-time formulation, as we will see. Conventionally, a transient simulation over a total time is divided into time steps or intervals of duration Δt and is discretised in a finite difference manner. The simplest method is the first order Euler scheme, illustrated in the following sketch.

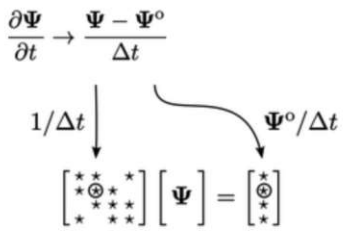


Figure 2.10: Time discretisation [4]

Therein (Fig.2.10), $\mathbf{\Psi^0}$ and $\mathbf{\Psi}$ denote the quantity at the old and the current time level, respectively. This way, the $\mathbf{\Psi^0}$ contributes to the source vector and $\mathbf{\Psi}$ to the system matrix [**A**]. The size of the time step can be evaluated via the Courant number

$$\mathrm{Co} = \frac{u_x \Delta t}{\Delta x} \tag{2.32}$$

where u_x and Δx are the advection velocity and the cell size, respectively. A 1D advection setup with an explicit upwind scheme introduces the Courant-Friedrich-Lewi condition for convergence, $\mathrm{Co} \leq 1$, suggesting that a quantity may not be transported further than one cell per time step. For other cases, the actual convergence limit depends on the spatial and temporal discretisation schemes and may fall below 1. Indeed, there exist also higher order time discretisation schemes like Euler backward scheme or Crank-Nicolson scheme, which are not examined in detail within the scope of this work. Beside the order of the time scheme, another essential characteristic is whether the time scheme in combination with the spatial discretisation schemes is *explicit* or *implicit*. In this context, *explicit* means that, *e.g.* the Laplacian term is discretised at the old time level $\mathbf{\Psi^0}$ and as a consequence only contributes to the source vector [**b**] in the governing system. On the other hand, implicit implies a discretisation of the term at the current time level, yielding contribution to the system matrix [**A**], also to off diagonal entries. An *implicit* simulation usually brings along better stability but more computational effort.[4]

2.2.4 Initial conditions

As described above, processing a simulation in time, values $\mathbf{\Psi^0}$ at the old time level have to be known. Thus, in order to start a simulation, initial conditions have to be set, for example, an initial temperature field.

2.2.5 Boundary conditions

The last ingredient to build up the final matrices is the incorporation of the boundary conditions. At each boundary patch of the computational domain, we have to specify conditions which correctly represent the physical behaviour of the simulated case. Herein, only the two most basic boundary conditions, the *fixed value* and the *fixed gradient* condition are examined. More advanced boundary conditions can be deduced from those two and are not needed for the test cases in this work. Whenever the discretisation of a term demands interpolation to a boundary face, it needs special treatment. Recalling the transport equation of (2.5) and the discretisation schemes above, the advection term and the Laplacian term transform to

- $\nabla \cdot (\boldsymbol{u}\boldsymbol{\Psi}) \rightarrow \sum_{\mathrm{f}} \phi_{\mathrm{f}} \boldsymbol{\Psi}$
- $\nabla \cdot (\Gamma \nabla \boldsymbol{\Psi}) \rightarrow \sum_{\mathrm{f}} |\mathbf{S}_{\mathrm{f}}| \, \Gamma_{\mathrm{f}} \nabla_{\mathrm{n}} \boldsymbol{\Psi}_{\mathrm{f}}$.

Thus, the values $\boldsymbol{\Psi}_{\mathrm{f}}$ and $\nabla_{\mathrm{n}} \boldsymbol{\Psi}_{\mathrm{f}}$ must be prescribed by the following boundary conditions:

- The *fixed value* boundary condition [3] assigns $\boldsymbol{\Psi}_{\mathrm{b}}$ a fixed value, *e.g.* a fixed temperature.
- The *fixed gradient* boundary condition [4] assigns $\nabla_{\mathrm{n}} \boldsymbol{\Psi}_{\mathrm{b}}$ a prescribed gradient value, where $\nabla_{\mathbf{n}} \equiv \mathbf{n} \cdot \nabla$. For example, a *zero gradient* condition suggests no change of the quantity at the boundary.

Depending on the the discretised term and the underlying boundary condition, both, the system matrix [$\mathbf{A}$] and the source vector [$\mathbf{b}$] are effected. [4]

2.3 Space-time finite volume formulation

The overall aim to transform time dependent partial differential equations into systems of linear equations does not change. The crucial difference of the space-time finite volume method is the treatment of the time discretisation. The following investigates the general idea and the necessary steps to transform the previously described finite volume method into space-time.

[3] Also called Dirichlet boundary condition
[4] Also called Neumann condition

2.3.1 General concept

In this model, the finite volumes do not only fill the spatial domain but rather the whole space-time domain. In this way, the time discretisation is no longer treated with a finite difference scheme which makes the method fully conservative. Further, the dimension of the computational domain increases by one, such that *e.g.* a transient 1D problem, where each time step is computed after another, transforms into a 1D+t ($\widehat{=}$2D) problem, which is then solved as one connected algebraic problem.

2.3.2 Finite volume mesh

In order to visualize an example space-time mesh, imagine a 1D beam which is divided into 8 finite volumes. Now assume that we want to compute a transient temperature distribution at this domain where the total simulation time is divided into 8 time steps. Hence, conventionally, 8 time steps would be processed on this 1D domain. If the space-time method is applied on the same spatial domain, the control volume transforms into a 1D+t ($\widehat{=}$2D) mesh, as depicted in Fig. 2.11.

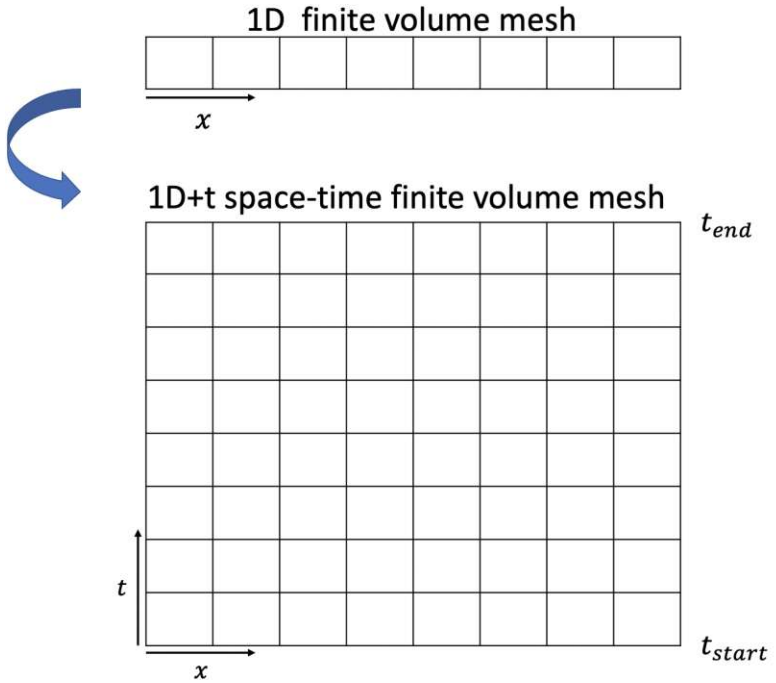


Figure 2.11: Space-time finite volume mesh

2.3.3 Equation discretisation

Above, in Sec.2.2.3, the discretisation methods for the conventional finite volume method have already been investigated. Concerning the transformation into space-time, most of the techniques remain equivalent, and only a few changes have to be done. The most incisive difference is the fusion of the time derivative into the convection term of a transport equation. In order to demonstrate this, let us recall the transport equation (2.2) of a quantity $\boldsymbol{\Psi}$ without sources

$$\underbrace{\frac{\partial \boldsymbol{\Psi}}{\partial t}}_{\text{temporal rate of change}} + \underbrace{\nabla \cdot (\boldsymbol{u}\boldsymbol{\Psi})}_{\text{convection}} = \underbrace{\Gamma \nabla^2 \boldsymbol{\Psi}}_{\text{diffusion}} . \tag{2.33}$$

Further, we introduce the space-time identities $\boldsymbol{u}^*, \nabla^*$ and γ^*:

$$\boldsymbol{u}^* = \begin{pmatrix} u_x \\ u_y \\ u_z \\ 1 \end{pmatrix} \tag{2.34}$$

$$\nabla^* = \begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \\ \frac{\partial}{\partial t} \end{pmatrix} \tag{2.35}$$

$$\gamma^* = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.36}$$

These identities can be used to transform (2.33) into

$$\nabla^* \cdot (\boldsymbol{u}^* \boldsymbol{\Psi}) = \gamma^* \Gamma (\nabla^*)^2 \boldsymbol{\Psi} \tag{2.37}$$

where $\boldsymbol{u}^*$ contains the convection velocities in the spatial dimensions and the *time velocity* being 1 in the time dimension, ∇^* is the nabla operator extended by the time derivative and γ^* ensures that $\boldsymbol{\Psi}$ only diffuses into the spatial directions. Before heading to the next steps, let us first take a look at a 2D+t space-time finite volume[5] (Fig. 2.12).

[5] a 3D+t space-time finite volume is hard to imagine, however, mathematically nothing changes

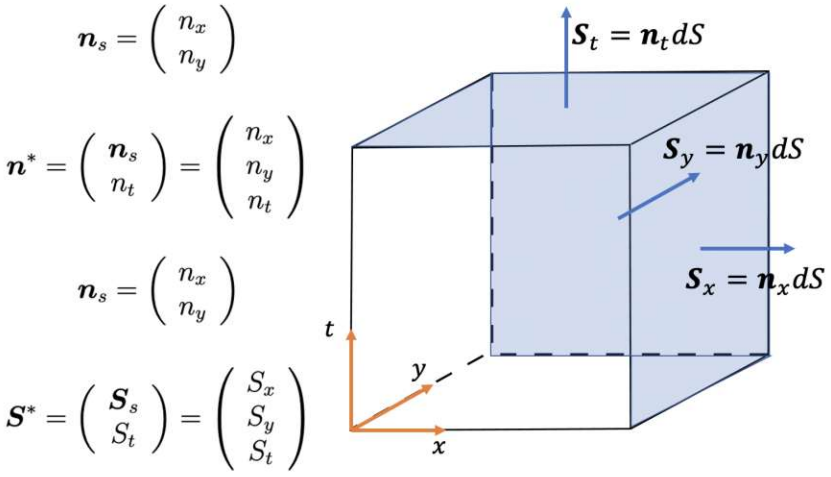


Figure 2.12: Space-time finite volume

Therein, the important quantities are the spatial face normal vectors $\boldsymbol{n}_s$ with their surface area vectors $\boldsymbol{S}_s = \boldsymbol{n}_s dS$ and the time face normal vectors $\boldsymbol{n}_t$ with their surface area vectors $\boldsymbol{S}_t = \boldsymbol{n}_t dS$. Extending these entities to 3D+t we obtain

$$\boldsymbol{n}^* = \begin{pmatrix} n_x \\ n_y \\ n_z \\ n_t \end{pmatrix} \tag{2.38}$$

and

$$\boldsymbol{S}^* = \begin{pmatrix} S_x \\ S_y \\ S_z \\ S_t \end{pmatrix} \tag{2.39}$$

with the important relation

$$\boldsymbol{S}^* = \boldsymbol{n}^* dS \tag{2.40}$$

where the cell surface encloses the space-time volume V^*. Integrating (2.37) over those volumes

$$\int_{V^*} \nabla^* \cdot (\boldsymbol{u}^* \boldsymbol{\Psi}) dV^* = \int_{V^*} \gamma^* \Gamma (\nabla^*)^2 \boldsymbol{\Psi} dV^* = \int_{V^*} \gamma^* \nabla^* \cdot (\Gamma \nabla^* \boldsymbol{\Psi}) dV^* \tag{2.41}$$

and applying *Gauss' divergence theorem* gives

$$\int_{S^*} (\boldsymbol{u}^* \boldsymbol{\Psi}) \boldsymbol{n}^* \cdot dS^* = \int_{S^*} \gamma^* (\Gamma \nabla^* \boldsymbol{\Psi}) \boldsymbol{n}^* \cdot dS^*. \tag{2.42}$$

Now, using (2.40) and the surface integral approximation of Sec.2.2.3

$$\int_{S^*} (d\mathbf{S}^* \cdot \boldsymbol{\Psi}) \rightarrow \sum_{\mathrm{f}} \mathbf{S^*}_{\mathrm{f}} \cdot \boldsymbol{\Psi}_{\mathrm{f}}, \tag{2.43}$$

the diffusion term of (2.42) becomes

$$\gamma^* \int_V \nabla^* \cdot (\Gamma \nabla^* \boldsymbol{\Psi}) dV = \gamma^* \int_S (\Gamma \nabla^*_{\mathrm{n}} \Psi) d\mathbf{S}^* \rightarrow \gamma^* \sum_{\mathrm{f}} |\mathbf{S}^*_{\mathrm{f}}| \Gamma_{\mathrm{f}} \nabla^*_{\mathrm{n}} \boldsymbol{\Psi}_{\mathrm{f}}, \tag{2.44}$$

which is effectively equivalent to (2.24) with zero entries in the time dimension, since γ^* excludes them and does not change the spatial terms. On the other hand, the advection term of (2.42) becomes

$$\int_{S^*} (\boldsymbol{u}^* \boldsymbol{\Psi}) \boldsymbol{n}^* \cdot dS^* = \int_{S^*} (d\boldsymbol{S}^* \cdot \boldsymbol{u}^* \boldsymbol{\Psi}) \rightarrow \sum_{\mathrm{f}} \mathbf{S^*}_{\mathrm{f}} \cdot \boldsymbol{u}^* \boldsymbol{\Psi}_{\mathrm{f}} \rightarrow \sum_{\mathrm{f}} \phi^*_{\mathrm{f}} \boldsymbol{\Psi}_{\mathrm{f}}, \tag{2.45}$$

with $\phi^*_{\mathrm{f}} = \mathbf{S}^*_{\mathrm{f}} \cdot \mathbf{u}^*_{\mathrm{f}}$, the volumetric flux, as in (2.28). And also here, the already described advection schemes have to be applied to obtain the face values $\boldsymbol{\Psi}_{\mathrm{f}}$ in the spatial directions as well as in the time direction.

In other words, the decisive difference to the conventional finite volume method is the fusion of the time derivative and the advection term. The discretisation of the other terms remain the same. The space-time tensor γ^* ensures that spatial phenomena like diffusion or surface tension are not propagated in time.

2.3.4 Boundary conditions

The boundary conditions in the space-time approach represent different physical meanings. Having a computational domain like in Fig. 2.11, the bottom boundary at t_{start} represents the initial condition of the case. For example, if we recall the setup in 2.3.2 and want to assign the 1D beam a starting temperature, the face values at the bottom boundary have to be set accordingly. The boundary *vis-à-vis* at t_{end} is set to *zero gradient* (see Sec. 2.2.5), which actually enforces a steady state at the t_{end} boundary. Hence, if the steady state is not yet reached, an error is induced at this point. A more appropriate boundary condition would be a *zero second derivative* condition, which reflects the physical behaviour more reasonably. However, the *zero gradient* is used in this work, since it is a standard functionality

in OpenFOAM® and the location and the extend of the error are acceptable for most cases. Along the time axis the boundary conditions can be set arbitrarily and have the same meaning as in a conventional simulation. In this context, it is easy to incorporate moving boundary conditions, provided that the motion is known *a priori*, since the boundaries capture the whole space-time domain.

2.3.5 Time slab method

In order to investigate the computational effort of both the conventional and the space-time approach, let us imagine a 1D spatial domain with n_s cells and the request to compute n_t time steps. If we seek a solution for one partial differential equation, *e.g* the heat PDE, the conventional method requires solving n_t times a $n_s \times n_s$ system of linear equations, whereas the space-time method requires solving **once** a $(n_s * n_t) \times (n_s * n_t)$ system of linear equations. In order to decrease the effort, the *time slab method* can be used where the time domain is divided into n_{slabs} *time slabs*, which are computed one after another. This results in solving n_{slabs} times a $(n_s * \frac{n_t}{n_{slabs}}) \times (n_s * \frac{n_t}{n_{slabs}})$ system of linear equations. Thereby, the results received at the faces of the t_{end} boundary of the previous *time slab* serves as the initial condition, *i.e.* the face values of the t_{start} boundary, for the current *time slab*. Herein, the previously discussed error induced by the *zero gradient* boundary condition takes action at each time slab. This issue demands the implementation of the *zero second derivative* boundary condition and further investigation. Nonetheless, this method is showcased in Sec.3.

3 Test cases

This section deals with the validation of the space-time finite volume method in OpenFOAM®. The complexity of the test cases will grow from the first to the last and will cover

- 1D+t - Diffusion
- 1D+t - Advection
- 1D+t - Advection/Diffusion
- 2D+t - Advection/Diffusion
- 2D+t - *Incompressible* Navier-Stokes
- 2D+t - *Incompressible* Navier-Stokes with moving boundary conditions

Each of the following subsections will start with a description of the case, followed by the underlying mathematical formulation, the transformation into space-time, the visualization of the results achieved by the conventional and the space-time OpenFOAM® solver. Additionally, possible benefits or weaknesses of the existing OpenFOAM® architecture will be carried out and also will be mentioned in the subsequent chapters, sections 4 and 5.

3.1 1D+t - test cases

The one-dimensional test cases investigate pure diffusion, pure advection and advection/diffusion each on the same geometry and mesh. For the following, let us assume a one dimensional bar. The meshes for the conventional and the space-time method are shown in Figs. 3.1 and 3.2. Therein, we can see the same discretisation in space and time, *i.e.* the spatial extension of $2m$ into 100 cells with $\Delta x = 0.02$ and the time extension of $2s$ into 100 cells with $\Delta t = 0.02$. For the sake of consistency, the time step of the conventional simulations is always set to $\Delta t = 0.02$ and also 100 steps are proceeded with the OpenFOAM® *scalarTransportFoam*. Keep in mind that a one dimensional bar only has spatial extensions in one direction.

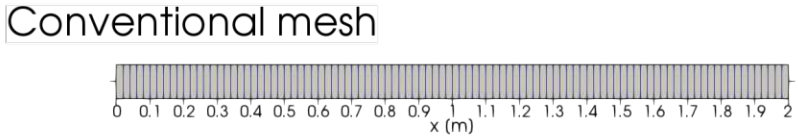


Figure 3.1: Conventional mesh

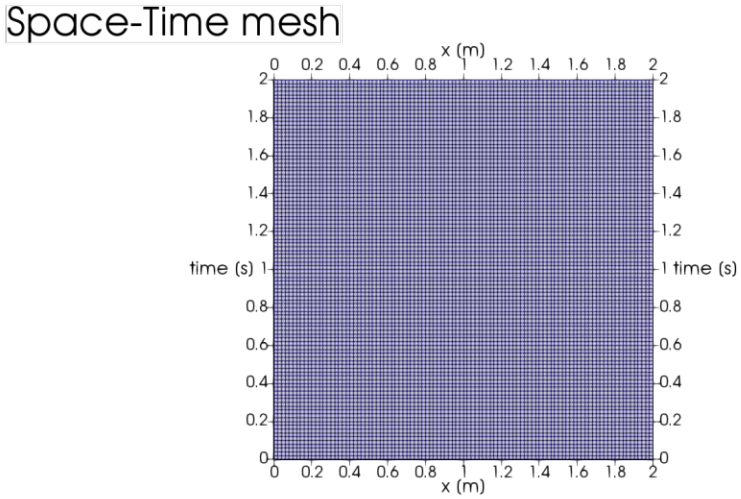


Figure 3.2: Space-Time mesh

3.1.1 1D+t - diffusion

The simplest setup to start is solving the diffusion equation

$$\underbrace{\frac{\partial T}{\partial t}}_{\text{temporal rate of change}} = \underbrace{\kappa \nabla^2 T}_{\text{diffusion}} \tag{3.1}$$

for temperature T and thermal conductivity $\kappa = 0.1\frac{m^2}{s}$. Let us assume

$$\begin{aligned} T(0,t) &= 1 \\ T(2,t) &= 0 \\ T(x,0) &= 0 \end{aligned} \tag{3.2}$$

which is a fixed temperature of $1K$ and $0K$ at the left and right boundary and an initial temperature contribution of $0K$ elsewhere. The according space-time problem reads

$$\nabla^* \cdot (\boldsymbol{u}^* T) = \gamma^* \kappa (\nabla^*)^2 T \tag{3.3}$$

with

$$\boldsymbol{u}^* = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{3.4}$$

$$\nabla^* = \begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial t} \end{pmatrix} \tag{3.5}$$

$$\gamma^* = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \tag{3.6}$$

with fixed value boundary conditions at the left, right and bottom boundaries set to be $1K$, $0K$ and $0K$, respectively and a *zero gradient* boundary condition at the top boundary of Fig. 3.2. The analytical solution to this particular problem is

$$T = 1 - \mathrm{erf}(x/2\sqrt{\kappa t}) \tag{3.7}$$

according to [1]. The result of the space-time finite volume method is revealed in Fig. 3.4, where we can see that the temperature diffuses away from the left boundary.

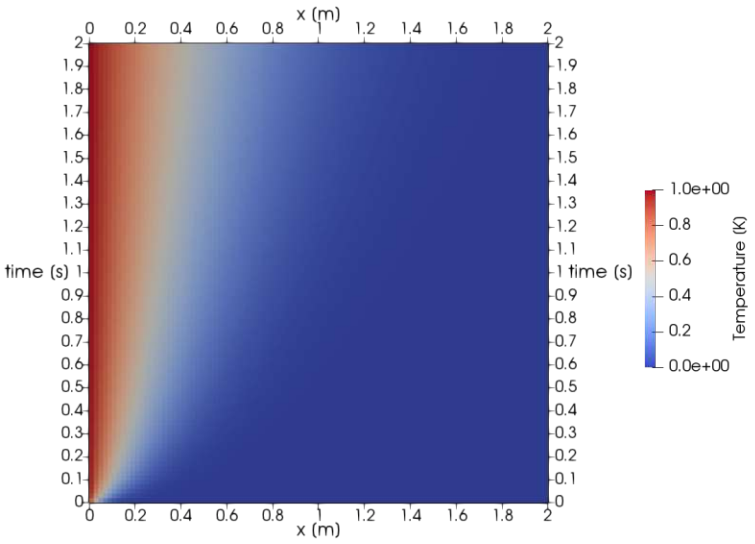


Figure 3.3: Space-Time solution pure diffusion

The *scalarTransportFoam* solver outputs the following temperature distribution recorded at $t = 1.0s$:

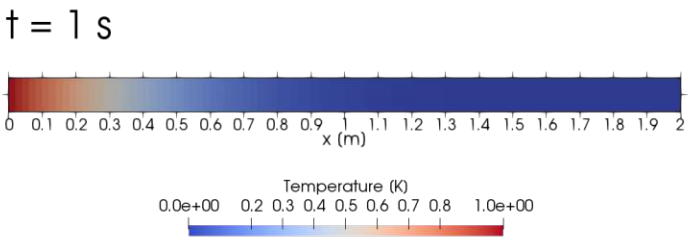
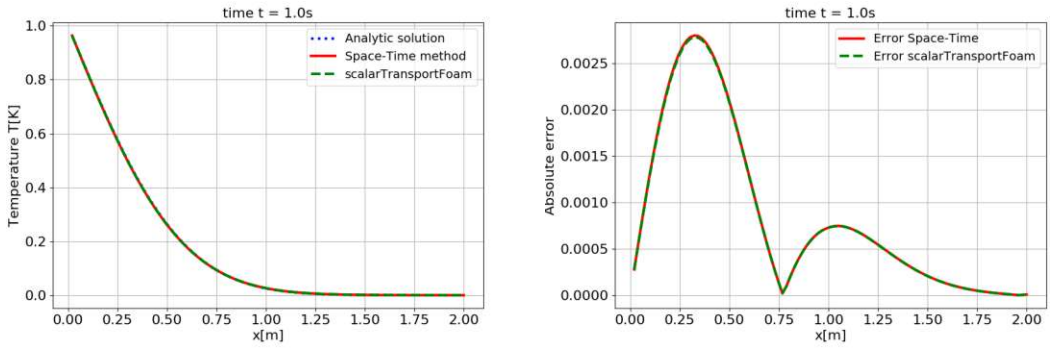


Figure 3.4: *scalarTransportFoam* pure diffusion $t = 1.0s$

A closer look at the results in Fig. 3.5, where the temperature distributions at $t = 1.0s$ of the analytical solution, the space-time simulation and the *scalarTransportFoam* simulation are plotted, outlines a good agreement of the data. The

absolute error is of the order Δt, which is not surprising, since the discretisation schemes in use are of order $O(\Delta t)$ and $O(\Delta x^2)$.



(a) Temperature distribution at $t = 1.0s$ (b) Absolute error at $t = 1.0s$

Figure 3.5: 1D diffusion

3.1.2 1D+t - advection

Next, the advection equation

$$\underbrace{\frac{\partial T}{\partial t}}_{\text{temporal rate of change}} + \underbrace{\boldsymbol{u} \cdot \nabla T}_{\text{convection}} = 0 \tag{3.8}$$

is considered on the same geometry with the only difference that the *fixed value* boundary condition on the right boundary is changed into a *zero gradient* condition. Here, the temperature is transported along the domain purely by the advection velocity, which yields the problem specification:

$$\begin{aligned} T(0,t) &= 1 \\ \frac{\partial T(2,t)}{\partial x} &= 0 \\ T(x,0) &= 0 \\ u_x(x,t) &= 1 \end{aligned} \tag{3.9}$$

The according space-time formulation of the problem is

$$\nabla^* \cdot (\boldsymbol{u}^* T) = 0 \tag{3.10}$$

with

$$\boldsymbol{u}^* = \begin{pmatrix} u_x \\ 1 \end{pmatrix} \tag{3.11}$$

$$\nabla^* = \begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial t} \end{pmatrix} \tag{3.12}$$

Here, the analytical solution is just the step function

$$T(x,t) = T_0(x - u_x t) \tag{3.13}$$

with T_0 being the initial contribution according to 3.12. For starters, first order schemes, *i.s. Gauss upwind* scheme in space and *Euler implicit* scheme in time are used for the *scalarTransportFoam* simulations. The space-time equivalent to the *Euler implicit* scheme is *Gauss upwind* scheme to discretise the time component of ∇^*. The result of the space-time simulation (Fig. 3.4) clearly reveals the numerical diffusion (similar to see Fig. 2.8) increasing in time. Also the *scalarTransportFoam* simulation (Fig. 3.7 snapshot at $t = 1.0s$) indicates the same effect.

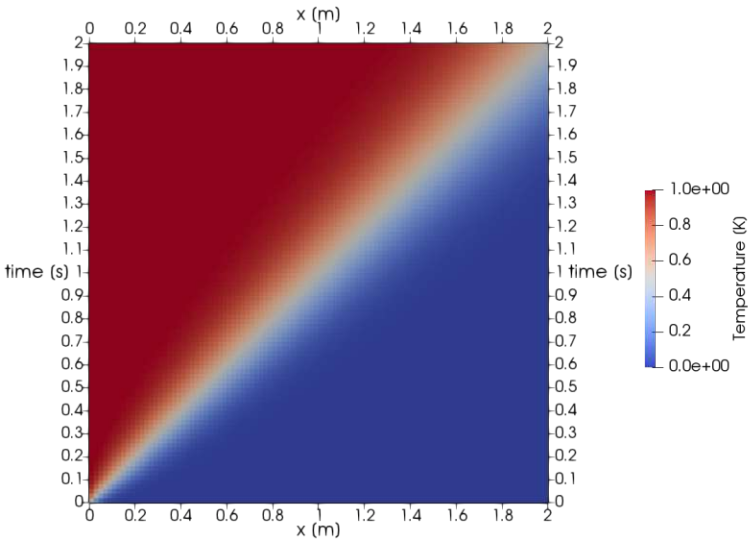


Figure 3.6: Space-Time solution pure advection

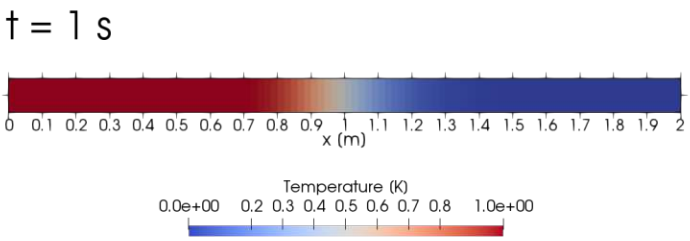


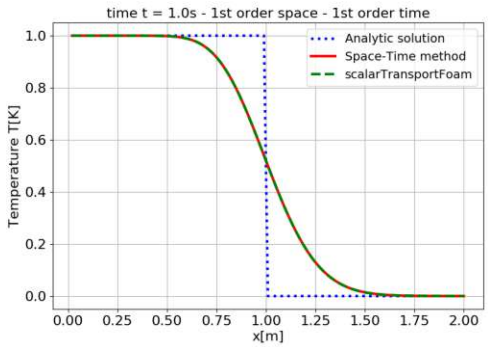
Figure 3.7: *scalarTransportFoam* solution pure advection

Further, having a closer look at the data at $t = 1.0s$ plotted against the analytical solution (see Fig. 3.8a), clarifies the impact of the numerical diffusion. Remedy for this is using higher order schemes. Those higher order schemes are commonly in use in OpenFOAM® performing conventional finite volume simulations, but correct

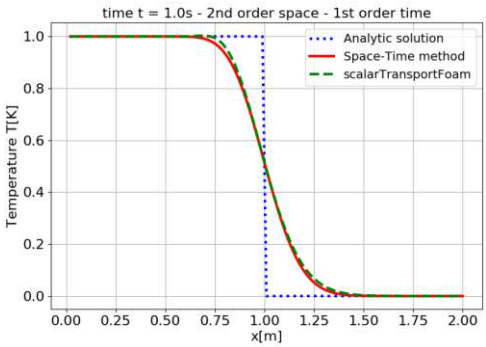
application within the space-time method demands special treatment. Therefore, we reorder 3.24 and introduce the pseudo time derivative $\frac{\partial}{\partial t^*}$

$$\frac{\partial T}{\partial t^*} + \nabla^* \cdot (\boldsymbol{u}^* T) - \gamma^* \kappa (\nabla^*)^2 T = 0 \tag{3.14}$$

and iterate over pseudo time steps with the desired discretisation schemes for ∇^* and $(\nabla^*)^2$ until $\frac{\partial T}{\partial t^*} = 0$, speaking in numeric terms until $\frac{\partial T}{\partial t^*} < \delta_{tolerance}$. The improvement of high order schemes can be seen in Figs. 3.8. Interestingly, using first order schemes in space and time yields the same result for both simulations. Then, increasing the order in space outputs small differences where the space-time simulation performs slightly worse before the advection step of the analytical solution, and slightly better afterwards. Switching both, spatial and temporal discretisation order to 2 indicates an advantage of the space-time method when it comes to overshoots in the surrounding of sharp steps, as we have it in this case. Therein, it has to be mentioned that the conventional simulation is processed with Crank-Nicolson scheme [2], which is, indeed, not the optimal choice in this case. OpenFOAM® offers the possibility to blend between Eulerian and Crank-Nicolson scheme, which would decrease the overshoots, but this is not within the scope of this work.



(a) Temperature distribution first order in space, first order in time



(b) Temperature distribution second order in space, first order in time

(c) Temperature distribution second order in space, second order in time

Figure 3.8: 1D advection snapshot at $t = 1.0s$ with different discretisation schemes

3.1.3 1D+t - advection/diffusion

Having examined pure diffusion and pure advection separately, let us now solve the combined advection/diffusion equation

$$\underbrace{\frac{\partial T}{\partial t}}_{\text{temporal rate of change}} + \underbrace{\boldsymbol{u} \cdot \nabla T}_{\text{convection}} = \underbrace{\kappa \nabla^2 T}_{\text{diffusion}} \tag{3.15}$$

within the same setup, boundary and initial conditions as for the pure advection problem with $\kappa = 0.01$. The space-time problem reads

$$\nabla^* \cdot (\boldsymbol{u}^* T) = \gamma^* \kappa (\nabla^*)^2 T \tag{3.16}$$

with

$$\boldsymbol{u}^* = \begin{pmatrix} u_x \\ 1 \end{pmatrix} \tag{3.17}$$

$$\nabla^* = \begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial t} \end{pmatrix} \tag{3.18}$$

$$\gamma^* = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \tag{3.19}$$

The analytical solution for this particular setup is [7]

$$\bar{T}(x,t) = \frac{1}{2} - \sum_{k=1}^{\infty} \frac{2}{(2k-1)\pi} \sin[(2k-1)\pi X/L] \mathrm{e}^{-\kappa(2k-1)^2\pi^2 t/L^2}. \tag{3.20}$$

The space-time solution and the *scalarTransportFoam* solution (again snapshot at $t = 1.0s$) are revealed in Figs. 3.9 and 3.10.

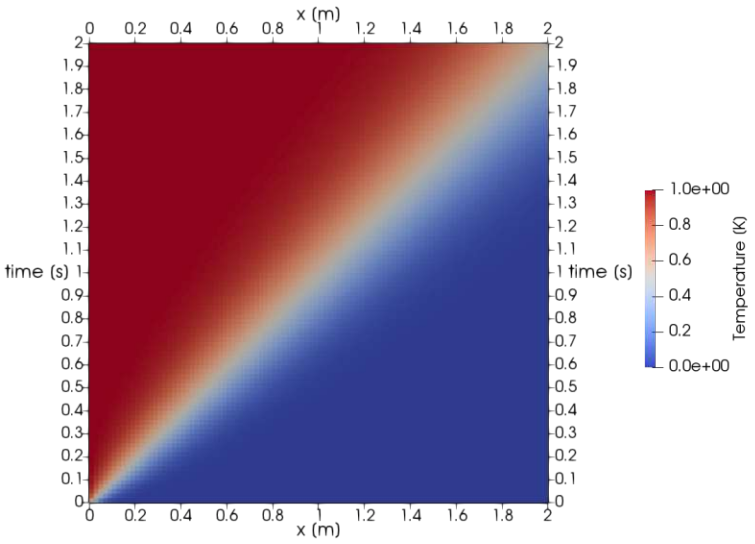


Figure 3.9: Space-Time solution advection/diffusion

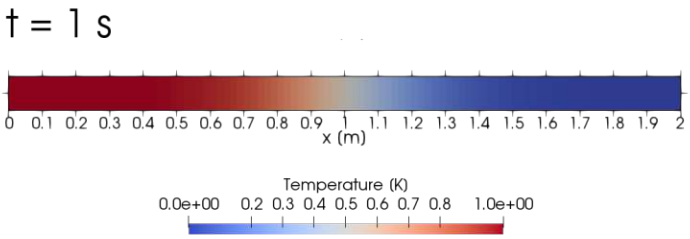
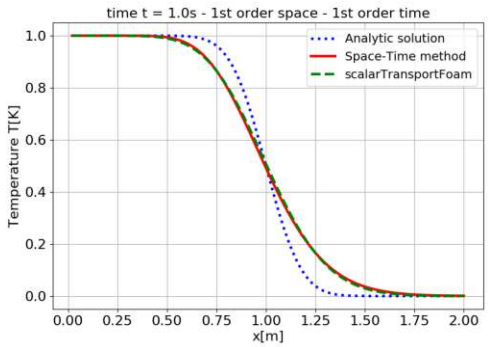
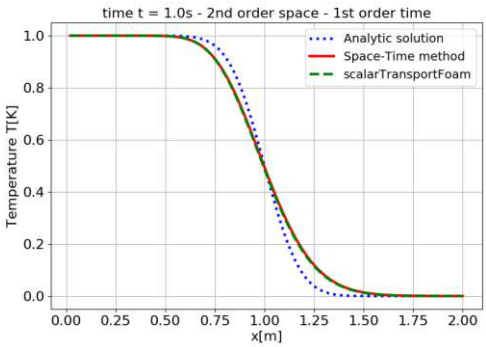


Figure 3.10: *scalarTransportFoam* solution advection/diffusion

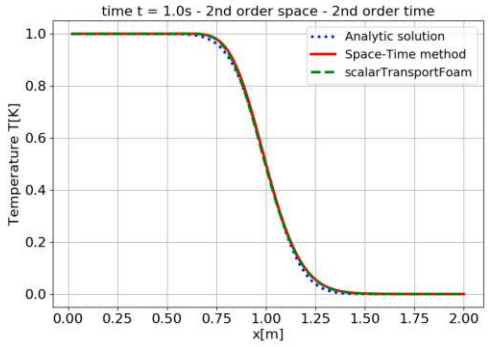
As done above, let us have a closer look at the results and the analytical solution plotted at $t = 1.0s$ in Figs. 3.11, where also different discretisation schemes are used. Compared to the pure advection case, the first order schemes already perform reasonably. Increasing the discretisation order to 2 for both space and time leads to sufficient accuracy in most cases.



(a) Temperature distribution first order in space, first order in time



(b) Temperature distribution second order in space, first order in time



(c) Temperature distribution second order in space, second order in time

Figure 3.11: 1D advection/diffusion snapshot at $t = 1.0s$ with different discretisation schemes

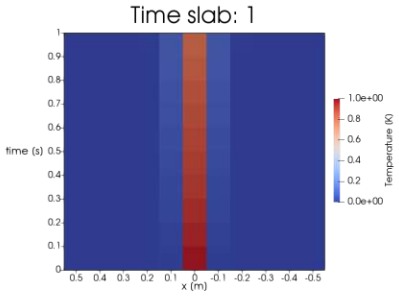
3.1.4 Time slab method

In Sec. 2.3.5 the time slab method has already been discussed. In short, the main idea is to split the whole time domain of the space-time simulation into evenly spaced time slabs in order to save computational effort. Thereby, the result at the end of one time slab serves as the initial condition of the next time slab. The applicability of the method is restricted to non moving boundaries. Fortunately, the OpenFOAM® architecture to process and save time steps can be fully utilised to carry out the method. This can be illustrated by a simple 1D+t diffusion simulation, where we have a 1D bar that is discretised by 11 evenly spaced cells in space direction with $\Delta x = 0.1$. Initially, the bar has a temperature of $1K$ at the center cell and is 0 elsewhere. At the space boundaries the *zero gradient* condition

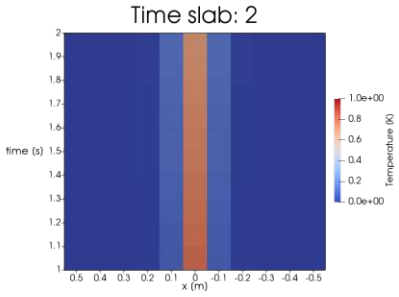
is applied, expecting a diffusion of the temperature in time. Thus, we want to solve the space-time equation

$$\nabla^* \cdot (\boldsymbol{u}^* T) = \gamma^* \kappa (\nabla^*)^2 T \tag{3.21}$$

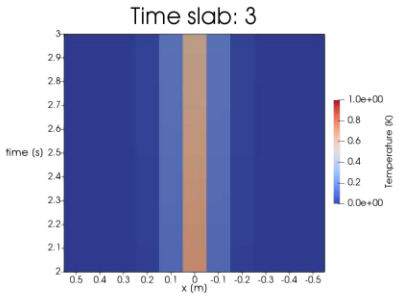
as in Sec.3.1.1, now, with $\kappa = 0.001 \frac{m^2}{s}$. If we seek to know the solution after $10s$ with a $\Delta t = 0.1$ we would yield a $[11 \times 100]$ domain and solve a $(11*100) \times (11*100)$ system of equations. Applying the time slab method and dividing the time domain into 10 time slabs, we now have to solve 10 times a $(11 * 10) \times (11 * 10)$ matrix. Fig.3.12 shows the solutions of the first, second, third and last time slab for the underlying problem.



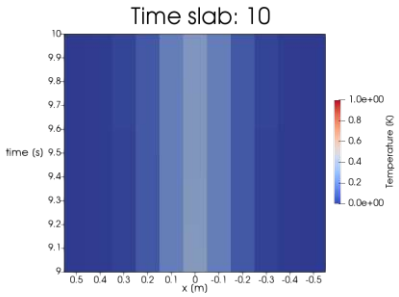
(a) Temperature distribution at first time slab $[0s, 1s]$



(b) Temperature distribution at second time slab $[1s, 2s]$



(c) Temperature distribution at third time slab $[2s, 3s]$

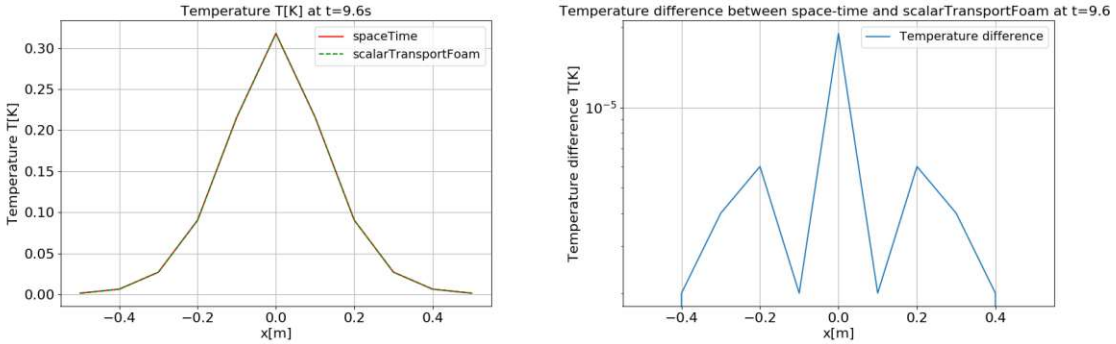


(d) Temperature distribution at last time slab $[9s, 10s]$

Figure 3.12: Illustration of the time slab method applied to the 1D+t diffusion problem described above

As already discussed in Sec. 2.3.5 the *zero gradient* boundary condition is expected to carry along an error, since it forces a steady state at the t_{end} boundary. The plots in Fig. 3.13 show the difference between the conventional simulation with

scalarTransportFoam and the space-time simulation using the time slab method. At the first sight, no difference can be seen in the left left figure. Only the right plot shows the small maximum offset of about 2e-5, which legitimises the use of the *zero gradient* boundary condition for the test cases within this work.



(a) Temperature distribution along x direction at $t = 9.6s$

(b) Temperature distribution difference between space-time and scalarTransportFoam simulation along x direction at $t = 9.6s$

Figure 3.13: Validation of the time slab method

3.2 2D+t - Advection/Diffusion

The 1D+t cases in the previous section validate the concept and implementation of the space-time finite volume method in OpenFOAM®. Now, as the complexity increases, there will no longer be analytical solutions. Thus, the space-time implementations will from now on only be compared to the standard OpenFOAM® solvers. The following test case solves the scalar transport equation [1] of temperature T for a prescribed constant velocity field in a 2D spatial domain. The geometry (see Fig.3.14) and setup is based on the experimental work of Pitz and Daily (1981) and, in terms of numerical investigations, has been much researched within the OpenFOAM® community [10].

[1] equivalent to advection/diffusion equation

Figure 3.14: PitzDaily geometry (dimensions in mm) [10]

Herein, the inlet velocity and outlet pressure condition (see Fig.3.14) induce a steady velocity field that can be evaluated with the OpenFOAM® solver *simpleFOAM*[10].

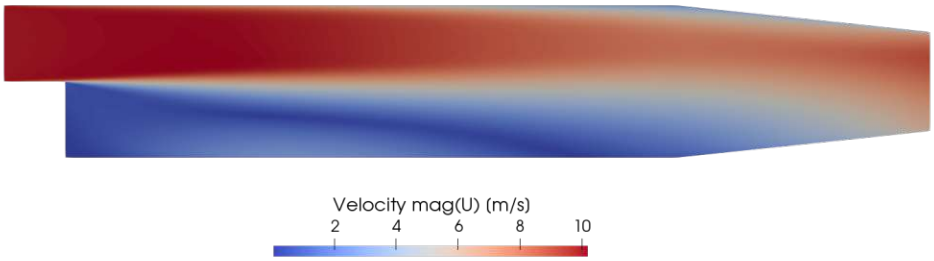


Figure 3.15: Pitz-Daily velocity field based on *simpleFOAM*

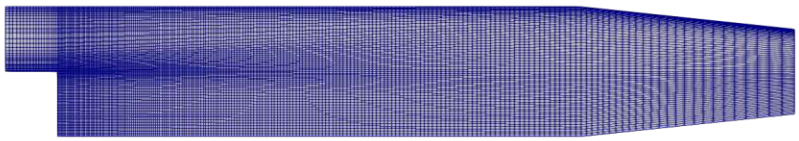


Figure 3.16: Pitz-Daily computational mesh

The steady velocity field (Fig.3.15) and the computational mesh (Fig.3.16) are the basis for our scalar transport test case. Therefore, we apply a fixed temperature of $1K$ at the inlet boundary, a *zero gradient* at the outlet and wall boundary and set an initial temperature field of $0K$ elsewhere according to

$$\begin{aligned} &T(x,y,t) = 1 \text{ at } \partial\Omega_{inlet} \\ &\frac{\partial T(x,y,t)}{\partial x} = 0 \text{ at } \partial\Omega_{outlet,wall} \\ &T(x,y,0) = 0 \text{ at } \Omega \end{aligned} \tag{3.22}$$

The according equation to solve is again

$$\frac{\partial T}{\partial t} + \boldsymbol{u} \cdot \nabla T = \kappa \nabla^2 T \tag{3.23}$$

with $\kappa = 0.01\frac{m^2}{1}$ and $\boldsymbol{u}$ being the prescribed advection velocity field. Conducting the simulation with *scalarTransportFOAM* from $t = [0s, 0.05s]$ with $\Delta t = 0.001$ gives following temperature distributions at $t = 0.01s$, $t = 0.03s$ and $t = 0.05s$ (Fig. 3.17). The numerical discretisation schemes for the time derivative and advection term are chosen to be first order for starters.

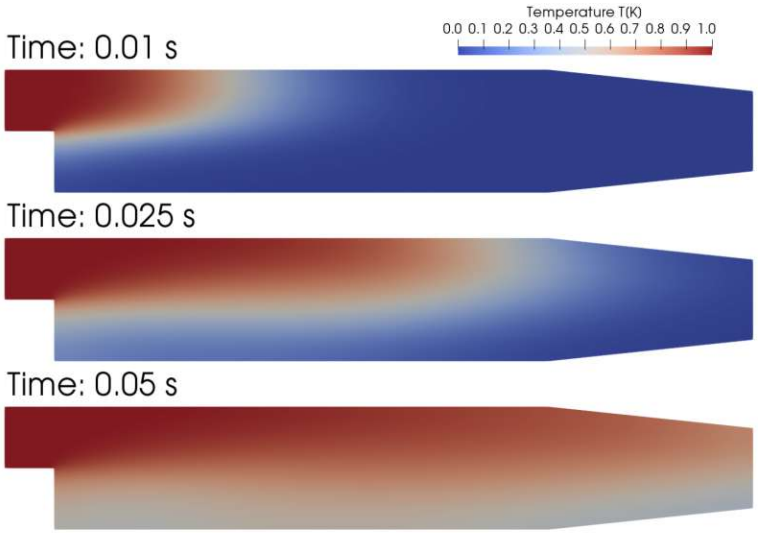


Figure 3.17: Pitz-Daily reference solution with *scalarTransportFOAM*

The according space-time problem again reads

$$\nabla^* \cdot (\boldsymbol{u}^* T) = \gamma^* \kappa (\nabla^*)^2 T \tag{3.24}$$

with

$$\boldsymbol{u}^* = \begin{pmatrix} u_x \\ u_y \\ 1 \end{pmatrix} \tag{3.25}$$

$$\nabla^* = \begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial t} \end{pmatrix} \tag{3.26}$$

$$\gamma^* = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{3.27}$$

Keeping the same discretisation as in the transient simulation in time and space yields following space-time mesh (Fig. 3.18) and solution for the temperature field (Fig. 3.19). Also here, the discretisation schemes for the ∇^*-operator are set to be first order in space and time directions.

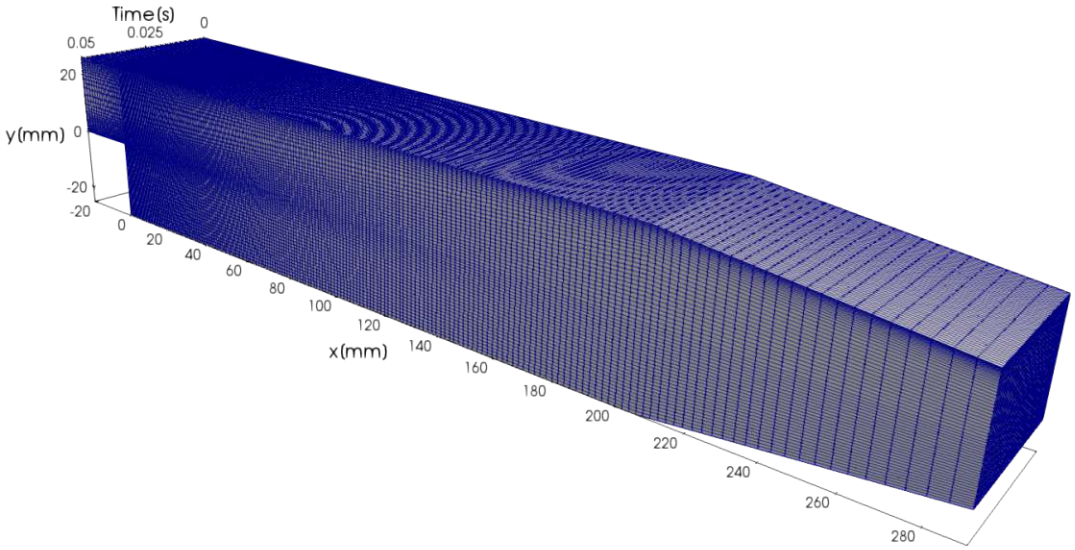


Figure 3.18: Pitz-Daily space-time mesh

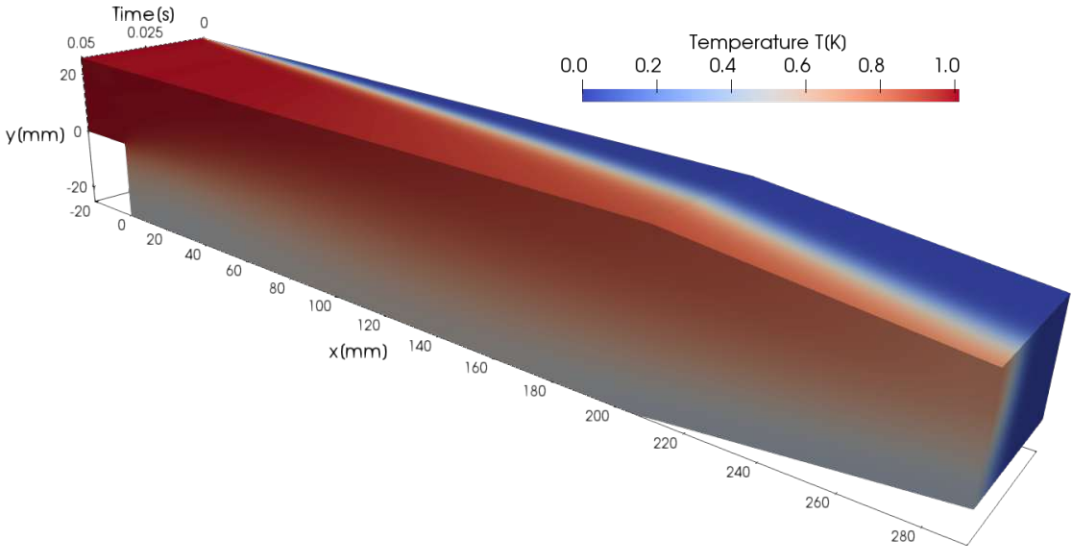
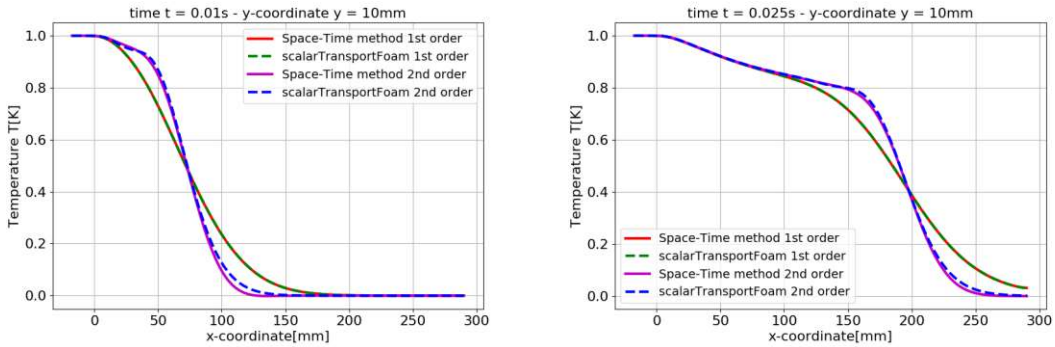


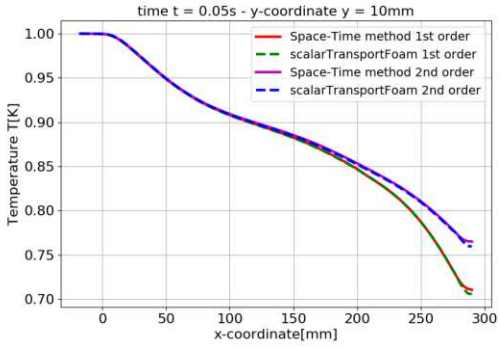
Figure 3.19: Pitz-Daily space-time solution for temperature field

At the first sight, the temperature field achieved by the space-time method resembles the one with *scalarTransportFOAM* qualitatively. In order to compare the results quantitatively, the temperatures are plotted at different time steps along the spacial iso line between $(-20.6mm, 10mm)$ and $(290mm, 10mm)$ in Fig. 3.20. The plots reveal that using first order discretisation schemes yield the same results for both approaches. The graphs, obtained with second order schemes, feature a much sharper step at the transportation front, suggesting that the first order schemes come along with numerical diffusion, as we have seen it in the 1D-t cases. However, within the transient simulations, Crank-Nicolson [2] time discretisation is in use. This explaines the small offset between the space-time and the transient simulations. At any rate, second order schemes also work for 2D+t cases within the underlying implementation fully using OpenFOAM® methods.



(a) Temperature plotted over x-coordinate at $t = 0.01s$

(b) Temperature plotted over x-coordinate at $t = 0.025s$



(c) Temperature plotted over x-coordinate at $t = 0.05s$

Figure 3.20: Comparison of Pitz-Daily results with the space-time method to *scalarTransportFOAM* at different time steps and different discretisation schemes

3.2.1 Local time stepping

The previous case provides a suitable possibility to introduce local time stepping. Let us assume the same setup as before and double the time domain, *i.e.* $t = [0s, 0.1s]$. Having high temperature gradients at the conspicuous transportation front offers the idea to refine the space-time volumes in this region. Figs. 3.21 and 3.22 show the refined mesh and the according solution of the problem. Therein, full usage of the OpenFOAM® methods can be applied, *i.e.* static as well as dynamic refinement. The advantage compared to dynamic refinement in according transient simulations is the local placement of refined cells. Thus, coarse cells can be used in regions of small change and fine local refinement can be complied in critical

regions where higher accuracy is demanded. In conventional simulation, having some velocities in regions where spatial refinement is high, forces us to use a small time step for the whole domain.

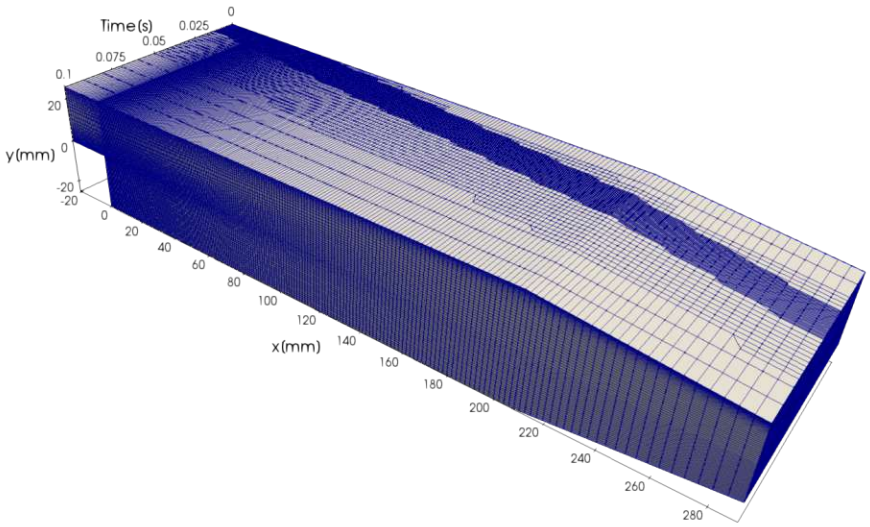


Figure 3.21: Pitz-Daily locally refined space-time mesh

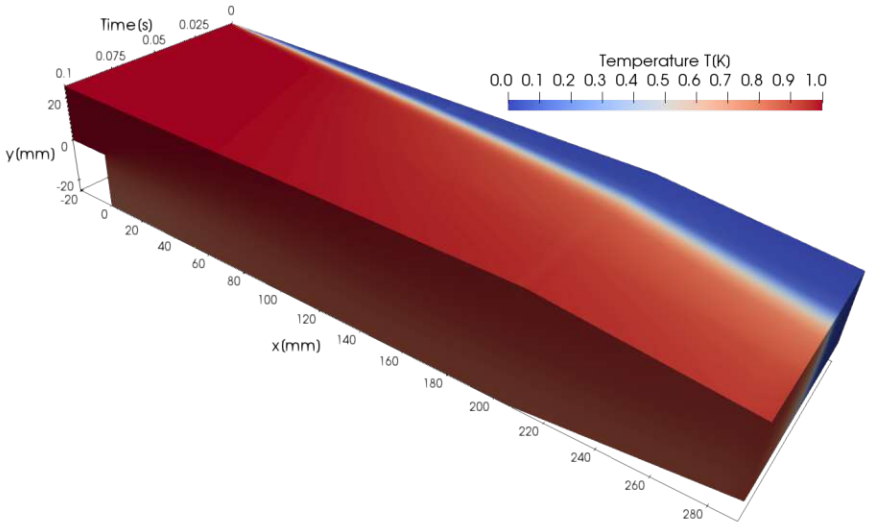


Figure 3.22: Pitz-Daily space-time solution for temperature field on locally refined mesh

3.3 2D+t - Incompressible Navier-Stokes

The second OpenFOAM® solver transferred into space-time is *icoFoam* [5], which computes the Navier-Stokes equations for *incompressible*, Newtonian fluids. *IcoFoam* applies the PISO algorithm to couple the velocity and pressure fields, as described in Sec.2.1.2. Thereby, the space-time equivalent to

$$\text{I:} \quad \frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot (\boldsymbol{u}\boldsymbol{u}) - \nabla \cdot (\nu \nabla \boldsymbol{u}) = -\nabla p \tag{3.28}$$

is

$$\text{I}^*\text{:} \quad \nabla^* \cdot (\boldsymbol{u}^*\boldsymbol{u}^*) - \gamma^* \cdot [\nabla^* \cdot (\nu \nabla^* \boldsymbol{u})] = -\gamma^* \cdot \nabla^* p \tag{3.29}$$

and

$$\text{II:} \quad \nabla^2 p + \nabla \cdot [\nabla \cdot (\boldsymbol{u}\boldsymbol{u})] = 0 \tag{3.30}$$

becomes

$$\text{II}^*\text{:} \quad \gamma^* \cdot [(\nabla^*)^2 p + \nabla^* \cdot [\nabla^* \cdot (\boldsymbol{u}\boldsymbol{u})]] = 0. \tag{3.31}$$

In other words, the time derivative in the momentum predictor (3.33) is pulled into the advection term and pressure equation (3.31) effectively remains unchanged contributing to spatial dimensions only. Similar to the steps described in Sec.2.1.2, we can write the momentum predictor without the pressure gradient as

$$\boldsymbol{A}^* \cdot \boldsymbol{u}^* \equiv A^* \boldsymbol{u}^* - \mathbf{H}^*(\boldsymbol{u}^*) \equiv \nabla^* \cdot (\boldsymbol{u}^*\boldsymbol{u}^*) - \nabla^* \cdot (\nu \nabla^* \boldsymbol{u}^*). \tag{3.32}$$

Including the pressure gradient, the momentum equation becomes

$$\boldsymbol{A}^* \cdot \boldsymbol{u}^* = \gamma^* \cdot [-\nabla^* p], \tag{3.33}$$

the pressure equation

$$\gamma^* \cdot [\nabla^* \cdot \frac{1}{A^*} \nabla^* p] = \gamma^* \cdot [\nabla^* \cdot \left[\frac{\mathbf{H}^*(\boldsymbol{u}^*)}{A^*}\right]], \tag{3.34}$$

and the momentum corrector

$$\boldsymbol{u}^* := \gamma^* \cdot [\frac{\mathbf{H}^*(\boldsymbol{u}^*)}{A^*} - \frac{1}{A^*}\nabla^* p]. \tag{3.35}$$

The procedure of the space-time PISO algorithm is similar to the conventional one, depicted in Fig.2.1. Also now, the momentum equation (3.33) is solved first, followed by solving the pressure equation (3.34) and the momentum correction (3.35), which form the inner PISO loop. After several runs of the inner PISO loop, the momentum equation (3.33) is solved again, denoting the outer loop, as can be seen in Fig.3.23. This procedure is done until the velocity field converges. After that, the flow and pressure fields are determined for the whole space-time domain. In general, both the inner and the outer loop have to be passed through by far more often than in conventional transient simulations.

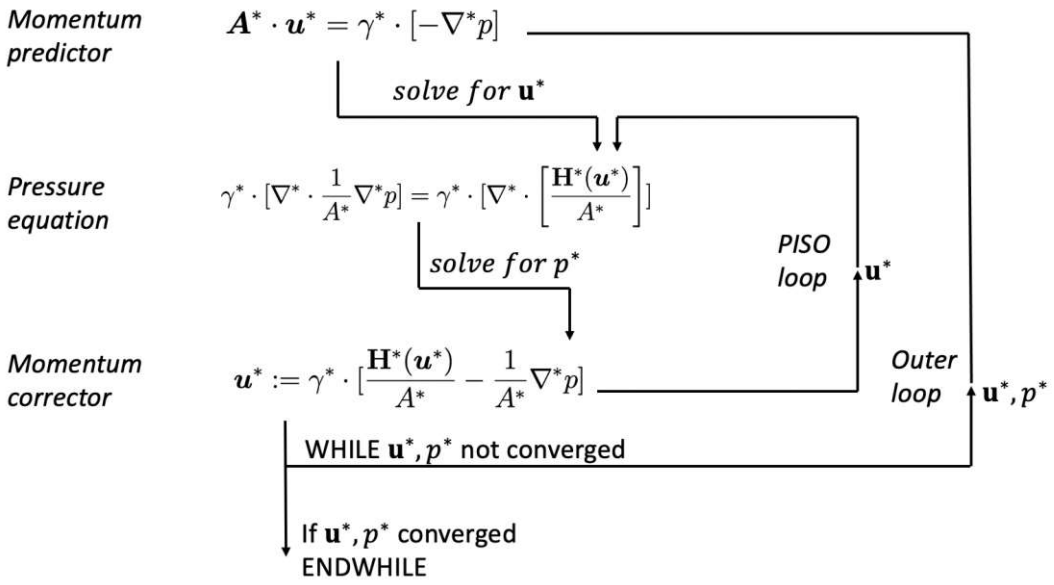


Figure 3.23: Space-time PISO algorithm (derived from [4])

3.3.1 Lid driven cavity

The first case to test the space-time *icoFoam* solver is the calculation of the flow field inside a fixed cavity driven by a moving wall at the top. The geometry and also the 2D mesh for the reference simulation with *icoFoam* can be seen in Fig.3.24.

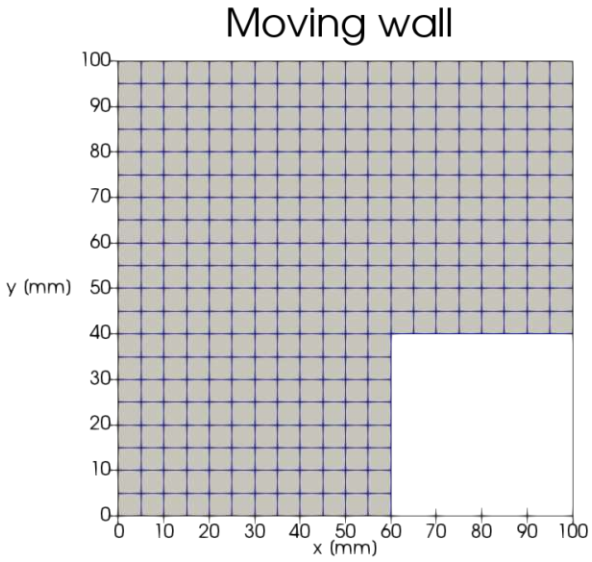


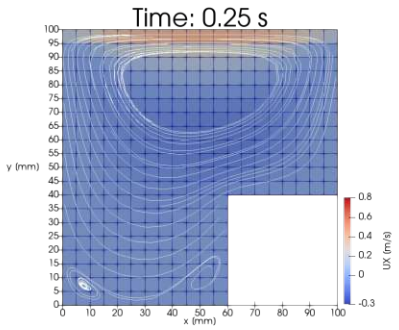
Figure 3.24: Lid driven cavity geometry and mesh

The fluid inside is assumed to be Newtonian and *incompressible*, with a constant viscosity of $\nu = 0.01m^2/s$. As usual, the boundary conditions for the velocity at the walls are set to *no slip* which means zero relative velocity between the fluid and the wall [2]. For the pressure, the *zero gradient* condition is used. The moving wall, that drives the flow is prescribed as

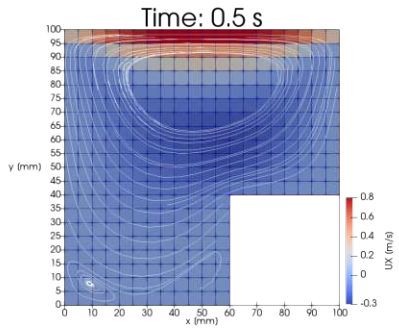
$$u_x = \begin{cases} 2t & \forall t < 0.5 \\ 2 - 2t & \forall t >= 0.5 \end{cases} \tag{3.36}$$

with a total simulation time of $t[0; 1s]$ and a $\Delta t = 0.005s$, yielding 200 time steps for the transient simulation. The reference simulation with *icoFoam* results in the the following evolution of the x-component of the velocity field (Figs.3.25). Therein, the evolution of secondary vortices can be seen.

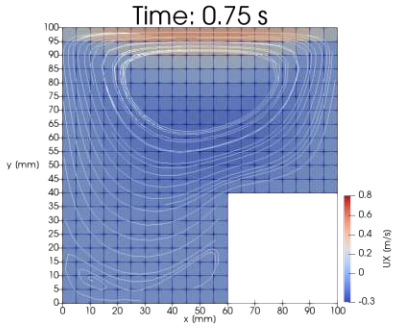
[2] This is equivalent to a *Dirichlet* boundary condition set to zero



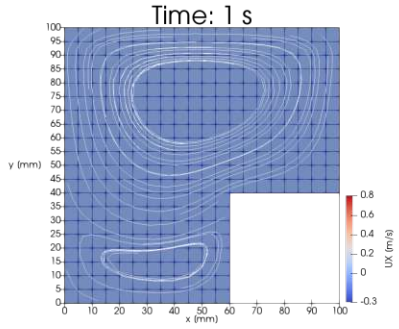
(a) Velocity u_x and streamlines at $t = 0.25s$



(b) Velocity u_x and streamlines at $t = 0.5s$



(c) Velocity u_x and streamlines at $t = 0.75s$



(d) Velocity u_x and streamlines at $t = 1s$

Figure 3.25: Evolution of the u_x and the streamlines over time generated with *icoFoam*

In order to perform the according space-time simulation, the front boundary condition for the velocity at $t = 0$ is set to be zero and the back one to *zero gradient*. The respective pressure boundary conditions are also set to *zero gradient*. Fig.3.26 reveals the space-time geometry of the underlying problem and already the resulting velocity field of the x-component. At the top boundary, the wall movement can be clearly seen. Mind that not the boundary values are shown here, but the cell values adjacent to the boundary. Therefore, the top values do not resemble the velocity of the wall.

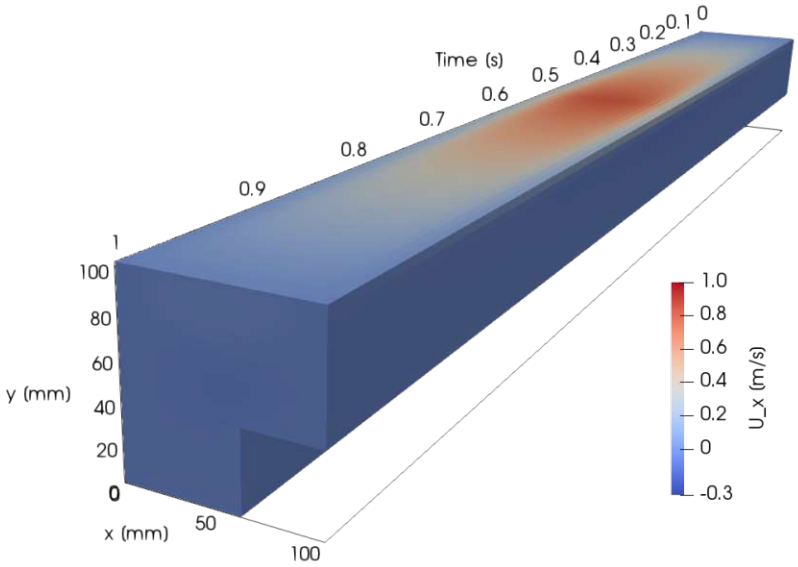


Figure 3.26: Lid driven cavity space-time geometry and result for u_x

This simulation converges after 10 outer loops with each 10 inner PISO loops (according to Fig.3.23). In order to show the result more accurately, Fig.3.27 reveals the velocity components u_x, u_y, the velocity magnitude and the streamlines at the time slice $t = 0.5s$, showing a good agreement with the reference simulation with *icoFoam*.

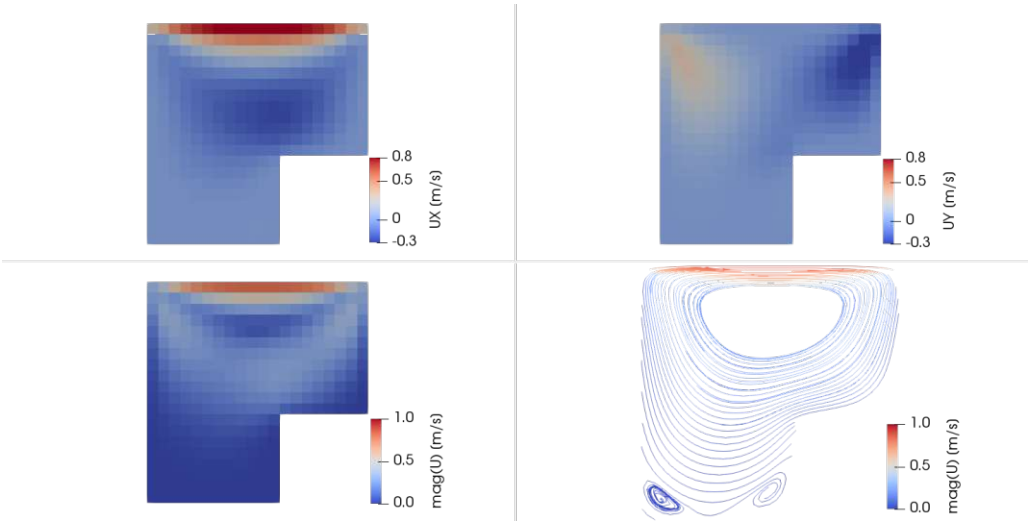


Figure 3.27: Lid driven cavity space-time geometry and result at $t = 0.5s$

Further comparison between the two simulations in Fig.3.28 indicate a small offset between the space-time and the reference simulation. Therein, the velocity magnitude is plotted at probe location $(22.5mm|82.5mm)$ over the entire time domain. There is a small deviation between the two solutions which could stem from slightly different behaviours of the employed discretization schemes and their space-time counterparts, or the different handling of temporal boundary conditions, which is unavoidable in space-time, or from the different convergence behaviour of the PISO loop in space-time, which requires a different number of corrector loops, and small errors could potentially accumulate over course of the solution. For such more complex cases like this one, a small deviation is to be expected.

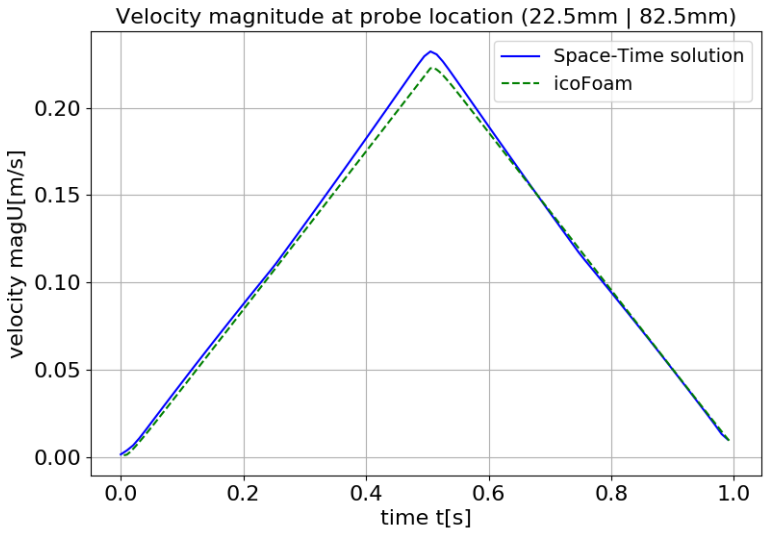


Figure 3.28: Velocity magnitude at probe location $(22.5mm|82.5mm)$ over the whole time domain

3.3.2 Moving boundary condition

The second test case is designed to demonstrate the simple handling of moving boundary conditions. For that, let us imagine an *incompressible*, Newtonian fluid flow in a 2D channel with time dependent indentation of the walls. Fig. 3.29 shows the dimensions of the simulation, the location of the time dependent indentation and the parabolic velocity distribution at the inlet boundary according to [8]. Selecting a maximum velocity of $u_{max} = 0.4\frac{m}{s}$ and a *no slip* condition at the walls, the inlet velocity profile typically becomes

$$u_y(x) = 1000 * (0.04 * x - x^2). \tag{3.37}$$

The indentation of the channel happens symmetrically with

$$h(t) = \begin{cases} 0.04 - 0.2t & \forall\ t < 0.1 \\ 0.02 & \forall\ t \mid 0.1 <= t <= 0.3 \end{cases}. \tag{3.38}$$

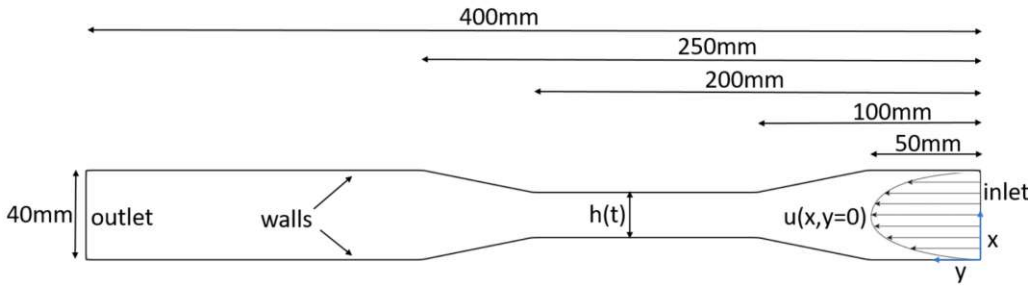


Figure 3.29: 2D channel flow with moving boundary conditions - set up

The according space-time geometry and mesh is depicted in Fig. 3.30. The boundary condition at the outlet is set to zero for the pressure and *zero gradient* for the velocity. At the inlet, we apply *zero gradient* for the pressure and the fixed velocity profile (3.39), suggesting that the indentation does not affect the inlet velocity. The same holds for the t_{start} boundary condition, which implies a steady state velocity profile before the indentation takes place. The t_{end} boundary conditions are set to *zero gradient* for both, the velocity and the pressure, which is legit, assuming a steady state at this point of the simulation.

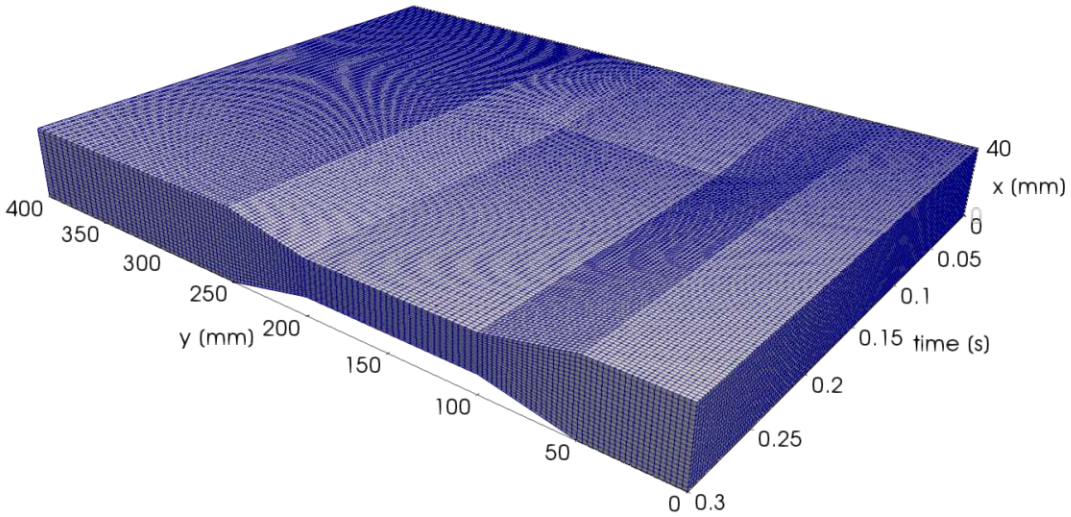


Figure 3.30: 2D channel flow with moving boundary conditions - space-time mesh

In order to remain a laminar flow state, the viscosity is chosen to be $\nu = 0.0005\frac{m^2}{s}$ for which the Reynolds number becomes $Re = \frac{vL}{\nu} = \frac{0.4*0.004}{0.0005} = 32$, with v and L the maximum velocity and the gap height, respectively. After [8], turbulence occurs for $Re > 1000$ in a flow between two plates. The simulation of the underlying problem with the space-time *icoFoam* solver yields the following pressure field (Fig. 3.31), and velocity fields for different time slices at $t = 0s, 0.05s, 0.1s$ and $0.2s$ (Fig. 3.32).

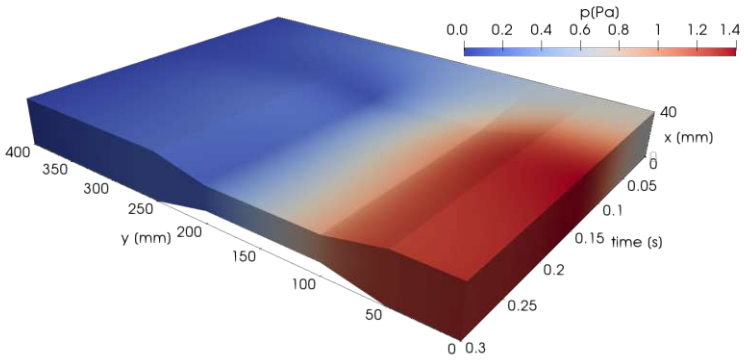


Figure 3.31: Pressure filed of 2D channel flow with moving boundary conditions

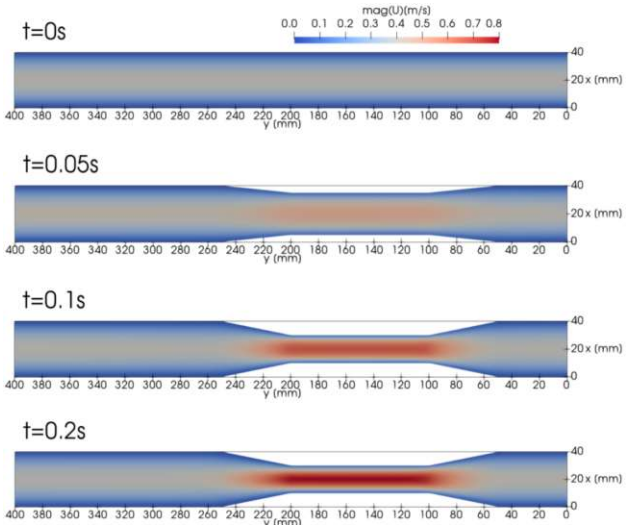


Figure 3.32: 2D channel flow with moving boundary conditions - velocity fields at different time slices

The velocity profile at the first time slice $t = 0$ shows the initial boundary condition at t_{start}. As the walls narrow down within the first $0.1s$, the velocity has to rise within the indentation. We see, that the peak velocity is not yet fully developed at $0.1s$, when the boundary movement is finished, but rather further increases, as we see at time slice $0.2s$. The indentation also induces a pressure peak in the vicinity of the inlet in order to uphold the velocity profile. The pressure peak occurs at about $0.15s$ before the gap and decreases afterwards. A closer look at the velocity development is brought by the subsequent plots, where the velocity magnitudes are plotted along different lines. More precisely, Fig. 3.33 shows the velocity magnitudes at the center ($x = 20mm$) along the y-axis at different times and Fig. 3.34 reveals the velocity magnitudes also at the center ($x = 20mm$) but along the time-axis at different locations y. If we have a close look at the first plot (Fig. 3.33), we see that the indentation presses the fluid towards the outlet, implying an increase in the velocity magnitude in this direction. As the indentation finishes at $t = 0.1s$, two velocity maxima form at both sides of the gap. After that, at $t = 0.15s$, the velocity maximum shifts back in the direction of the inlet until it finally approaches its expected[3] plateau of $0.8\frac{m}{s}$. The latter plot (Fig. 3.34) also reveals that the evolution of the velocity magnitude right in the middle of the indentation (blue curve) does not rise linearly, but rather suggests more complex phenomena taking place. Furthermore, the orange graph at $y = 350mm$ near the outlet indicates an increase of the velocity during the indentation followed by a decrease in order to fulfill the mass conservation. Conservation of mass can be checked by integrating the volume flow at the inlet boundary, which gives $0.0032m^2$ and equals the volume flow at the outlet boundary. This simulation output accords the analytic flow rate, calculated to be

$$\int_0^{x=h} u_y(x)dx * 0.3s = \int_0^{x=h} 1000 * (0.04 * x - x^2)dx * 0.3s = 0.0032m^2. \quad (3.39)$$

[3]The expected maximum velocity within the indentation is double the inlet maximum velocity as the height halves

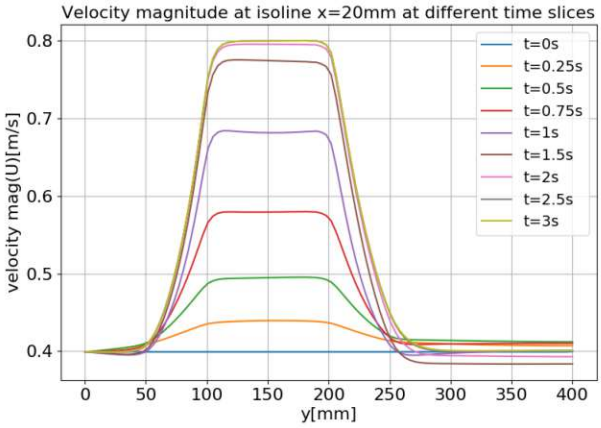


Figure 3.33: Velocity magnitude at center line ($x = 20mm$) along the y-axis at different time slices

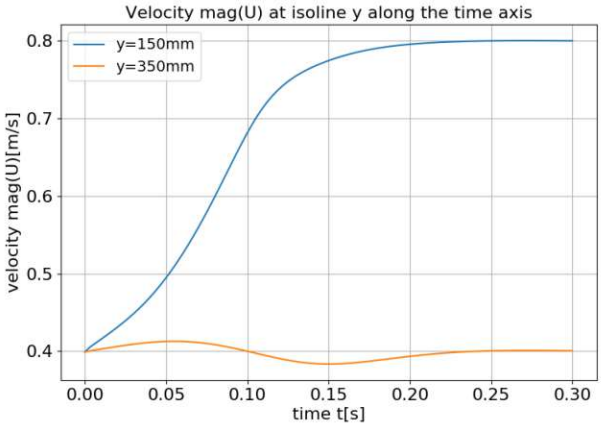


Figure 3.34: Velocity magnitude at center line ($x = 20mm$) along the time-axis at different y-locations

When it comes to moving boundary conditions, the conservation property reveals a huge advantage of the space-time method compared to conventional finite volume methods, where the Leibnitz rule or geometric conservation laws have to be applied [13]. The underlying test case shows that the fusion of the time derivative into the advection term naturally implies a fully conservative description. Hence, such cases can be solved without further modification of the solver.

4 Summary

This thesis has investigated the possibility to calculate time-dependent fluid mechanic problems by means of the space-time finite volume method by full usage of the OpenFOAM® functionality. In particular, the research has carried out following points.

- The governing equations for basic fluid flow problems such as the passive scalar transport equation and the Navier-Stokes equations for *incompressible*, Newtonian fluids are derived.

- The conventional finite volume method is presented including each step from simulation set up all along to solving the governing system of equations. Particular interest is put on the discretisation of the various terms of the partial differential equations.

- Based on the conventional method, the space-time finite volume method is introduced all along with the steps necessary to transform native OpenFOAM® solvers into space-time solvers.

- Having transformed OpenFOAM® solver *scalarTransportFoam* into space-time, the method is tested on basic pure diffusion, pure advection and advection-diffusion cases. Thereby, the 1D+t simulations yield the same result for both the conventional and the space-time solver. As the advection equation brings along numerical diffusion, higher order schemes provide remedy and give the desired order of error. Further, the time slab method is showcased at a pure diffusion test case. Moreover, the space-time method delivers reasonable results in the 2D+t Pitz-Daily [10] test case, where local refinement in space and time is applied.

- Finally, the OpenFOAM® solver *icoFoam* is transferred into space-time and tested on two cases. The lid driven cavity test case yields reasonable agreement between the native OpenFOAM® solver and the space-time solver with a small offset due to the advanced complexity of the problem. The second case, *i.e* the moving boundary case, demonstrates the simple handling dealing with time dependent domains. Therein, the flow evolution in an indenting 2D channel is investigated. This test case shows the implicit conservation property in space and time and reveals the advantage compared to conventional moving boundary techniques.

5 Discussion and outlook

The essence of the underlying thesis is that the transformation of native OpenFOAM® finite volume solvers into space-time finite volume solvers does work. Thereby, the space-time method involves some advantages and some disadvantages. The two main weaknesses of the method are:

- The investigation of the computational costs, examined in Sec. 2.3.5, suggests that a simulation with the space-time method demands far more resources than the conventional method, provided that the simulations feature the same discretisation in space and time.

- As already mentioned in the introduction, there is no algorithm known yet that manages 4D mesh generation. Hence, the space-time method is still settled with up to 2D+t problems or at most, rotationally symmetric 3D+t simulations. Indeed, 4D mesh generators are currently under development. For a matter of fact, the maths remain the same.

Of course, these are severe drawbacks limiting the applicability of the space-time method. Nevertheless, some advantages and potentials arise that justify further development of the approach:

- This thesis shows the huge benefit that it is not necessary to write new space-time finite volume code from scratch but rather manipulate existing finite volume programs in order to transform established methods into space-time. In this regard, for example OpenFOAM® brings full functionality from mesh generation all the way to the visualisation of the simulation. Especially, available adaptive mesh refinement and paralellisation techniques enable enormous performance improvement. For sure, OpenFOAM® is not the only software providing the suitable basis to conduct space-time simulations. Therefore, this work should encourage researchers to make their own experiences with the space-time method based on their finite volume and also finite element codes and enlarge the space-time community. Anyway,

further OpenFOAM® solvers have to be transformed in order to deal with turbulence modelling, multi-phase flows, compressability *etc.*. Actually, the space-time method should be applicable to any time-dependent problem that is suited for a numerical simulation.

- Counteraction regarding the issue of higher computational effort using the space-time method is brought by the possibility of local refinement and parallelisation in space and time. Imagine a conventional transient simulation on a big domain, suggesting the difference between the smallest cells and the domain dimensions is of several orders of magnitude. One according example is the simulation of an additive manufacturing process, where the domain is relatively large and, in some regions, the resolution demand is extremely high in order to reasonably resolve the underlying phenomena. If then, in those regions of small cells, higher velocities occur, the Courant number enforces a very small time step. As a consequence, the whole computational domain has to be processed with this time step, also in regions, where nearly nothing is happening at that time. As a consequence, if we wanted to process the simulation in parallel, on a cluster for instance, communication between the cores as well scales with the enlarged number of time steps. In this scenario, the according space-time simulation with its opportunity to refine locally in space and time, paired with optimal parallelisation and the time slab method, could yield performance benefits compared to the conventional simulation. At this stage of research, this is yet hypothetical, but still, a permissible thought experiment.

- Incorporation of moving boundary conditions is still a big issue in finite volume simulations. As demonstrated here, handling prescribed boundary motion is straight forward due to the conservation property of the space-time method in both space and time. This offers a huge advantage compared to conventional methods. The quite simple test case in this thesis, where a channel flow undergoes boundary indentation, could be extended to far more complex cases as soon as turbulence models and *compressability* are included in the space-time solvers. Another interesting use case could be aerodynamic investigations on airfoils with time-varying inclination angle.

All in all, this study has only scratched the surface of potential research. Subsequent steps will be the implementation of the *zero second gradient* boundary condition and the transformation of *compressible* multi-phase solvers. Furthermore, an analytic study on the possible performance gain regarding large scale simulations has to be done. And last but not least, a proof of concept to expand the space-time method to the fourth dimension is planned.

Bibliography

[1] R. R. Yadav Atul Kumar Dilip Kumar Jaiswal. *Analytical Solutions of One-Dimensional Temporally Dependent Advection-Diffusion Equation along Longitudinal Semi-Infinite Homogeneous Porous Domain for Uniform Flow*. Department of Mathematics Astronomy, Lucknow University, Lucknow-226007, U.P, India, (2012).

[2] *Crank-Nicolson scheme*. `https://www.openfoam.com/documentation/guides/latest/doc/guide-schemes-time-crank-nicolson.html`. Accessed: 2022-11-25.

[3] R.Ma *et al. On the geometric conservation law for unsteady flow simulations on moving mesh*. Computational Aerodynamics Institute, China Aerodynamics Research and Development Center, Mianyang Sichuan, 621000, China, 2015.

[4] Christopher Greenshields and Henry Weller. *Notes on Computational Fluid Dynamics: General Principles*. Reading, UK: CFD Direct Ltd, 2022.

[5] *icoFoam*. `https://www.openfoam.com/documentation/guides/latest/doc/guide-applications-solvers-incompressible-icoFoam.html`. Accessed: 2022-10-13.

[6] J. Kettemann and C. Bonten. *Application of the immersed boundary surface method in OpenFOAM*. IKT, Institut für Kunststofftechnik, University of Stuttgart, Pfaffenwaldring 32, 70569 Stuttgart, Germany, (2020).

[7] Hendrik C. Kuhlmann. *Numerische Methoden der Strömungsmechanik*. Institut für Strömungslehre und Wärmeübertragung, Technische Universität Wien, (2004-2021).

[8] Hendrik C. Kuhlmann. *Strömungslehre für Wirtschaftsingenieure-Maschinenbau*. Institut für Strömungslehre und Wärmeübertragung, Technische Universität Wien, (2004).

[9] *Multiple Reference Frame*. `https://www.learncax.com/knowledge-base/blog/by-author/ganesh-visavale/cfd-modeling-approach-for-turbomachinery-using-mrf-model`. Accessed: 2023-01-17.

[10] *Pitz Daily OpenFOAM*. `https://www.openfoam.com/documentation/tutorial-guide/3-compressible-flow/3.1-steady-turbulent-flow-over-a-backward-facing-step`. Accessed: 2022-11-29.

[11] *scalarTransportFoam*. `https://www.openfoam.com/documentation/guides/latest/doc/guide-applications-solvers-basic-scalarTransportFoam.html`. Accessed: 2022-10-12.

[12] Christian B. Allen Thomas C. S. Rendall and Edward D.C. Power. *Conservative unsteady aerodynamic simulation of arbitrary boundary motion using structured and unstructured meshes in time*. Department of Aerospace Engineering, University of Bristol, Bristol, BS8 1TR, UK, (2011).

[13] Philip J. Zwart. *The Integrated Space-Time Finite Volume Method*. University of Waterloo, (1999).