Quasi One-Dimensional Modelling of Turbulence and Interaction of Combustion Chambers in a Shockless Explosion Combustor

Dissertation
zur Erlangung des akademischen Grades einer
Doktorin der Naturwissenschaften (Dr. rer. nat.)

am Fachbereich Mathematik und Informatik
der Freien Universität Berlin

vorgelegt von
Giordana Tornow

Berlin 2021

Erstgutachter und Betreuer:
Prof. Dr.-Ing. Rupert Klein

Zweitgutachter:
Prof. Dr.-Ing. Christian Oliver Paschereit

Tag der Disputation:
10.11.2021

Selbstständigkeitserklärung

Abstract

This thesis deals with the modelling of two-dimensional coupling of quasi one-dimensional domains and turbulence within a quasi one-dimensional combustion chamber. Also an interpolation-free finite volume moving mesh method is described.

First, the basic framework of a gas turbine is introduced including an uncommon approach for constant volume combustion: the shockless explosion combustion (SEC). In a preceding work a simulation code for this combustion process solving quasi one-dimensional reactive Euler equations with a finite volume (FV) Riemann solver has been developed and was extended for the thesis at hand.

A network model is presented, allowing for the investigation of interaction of multiple pulsating combustion chambers of an SEC gas turbine with the plenums and each other. It couples the quasi one-dimensional domains using boundary conditions and flux corrections such that interactions of slanted combustion chambers with the plenums are possible. A series of simulations utilising this model is carried out to show possible fields of research for this tool.

As the simulation of combustion processes are especially sensitive to spacial resolution but complex chemistry also imposes restrictions on the number of grid cells a feature for adaptive remeshing is described. It uses the moving mesh idea within the FV solver. As interpolation introduces too much numerical diffusion a flux correction is given which evolves governing equations and mesh simultaneously without changing the Euler equations themselves. The performance of this feature is demonstrated with simulations of a detonation and a cyclic SEC.

Finally, the prerequisites for the research of the starting process of an SEC gas turbine are created by including molecular transport and turbulence in the SEC-code. Towards this aim, the one-dimensional turbulence (ODT) model is adjusted for this application. The ODT-line on which the stochastic eddy events, representing the turbulence, occur is aligned with the streamwise direction of the long-stretched combustion chamber. Also ODT is used as a stand-alone and subgrid-scale model. The main features of turbulence and ODT are compared to the new variant ODT-FHD. This study reveals that the ODT-FHD is able to generally reproduce the correct dependency of turbulence on mean flow velocity along with a plausible distribution of eddy sizes and kinetic energies. While lacking the possibility to generate new extrema of flow properties along the ODT-line it incorporates turbulent diffusion very well. The influence of the three model parameter is shown in addition to the simulation of a turbulent flame and a turbulent single-tube SEC.

Contents

Those who know nothing must believe everything.
- *Marie von Ebner-Eschenbach*

Chapter 1

Introduction

Energy is the secret star in most of our lives. We use it to travel far, stay up late, entertain or educate ourselves and even to wash our hands. The infrastructure around us breathes energy and although we are starting to realise it should get less, we are consuming more and more. Since going back to a new "candle light age" as a whole global society is unthinkable, it is vital to rethink our energy production processes. Surely, clean, renewable energies need to be our final goal but until this point is reached, it still helps to improve established production techniques by saving fuel and emissions as well as by gaining knowledge. One of these classical energy producing machines is a gas turbine. Although the sketch in Figure 1.1 depicts a jet engine which is not used in a power plant but in air planes, the general architecture and working principle is the same: First air is sucked in from the environment and densified by several stages of a compressor. This air is blown through a connecting volume called plenum into typically 5 to 6 combustion chambers. There it gets mixed with a fuel and burned continuously in a turbulent, subsonic flame. The hot gas expands and thereby drives a turbine which lies behind the combustion chambers and a second plenum. Its shaft is usually coupled with the compressor shaft, to keep driving it once the turbine started to run, and a generator. The latter converts the kinetic energy from the rotating turbine shaft to electricity. Over the past decades a lot of efforts have been made to increase the efficiency of such stationary gas turbines and eventually the curve reached its saturation point. There is not much left to enhance today if we stick to the basic principles of the workflow. Therefore, we change it.

In the last century new concepts for burning fuels have arisen and been studied. What they have in common is the idea of replacing the constant pressure combustion (CPC) of nowadays deflagration with a constant volume combustion (CVC). From theoretical thermodynamic cycle analysis (see e.g. [31]) we know that a perfect CVC is much more efficient than a perfect CPC thanks to the substitution of the Brayton cycle (also known as Joule cycle) by the Humphrey cycle. Actually, this ansatz has already been realised in combustion engines because cars and motorcycles

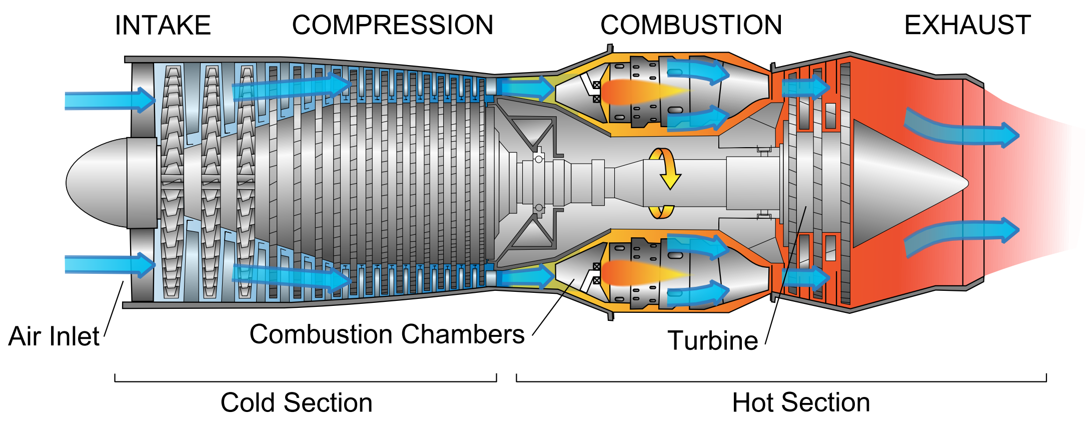

Figure 1.1: Scheme of a jet engine showing the gas flow from left (upstream) to right (downstream) through the main machine parts [17].

have worked like that since the beginning of the last century. The one but critical difference is that the combustion chambers in such a piston engine are closed and the ones in gas turbines are not. This means that the hot gas in a car engine has no other chance but to keep its volume and instead raise the pressure while in a gas turbine it can freely expand. It is of course possible to close the combustion chambers in gas turbines but that would overthrow the complete working and design principle of it. The other option is to burn the fuel as fast as - or even faster than - the gas can expand, thus with the speed of sound within the combustion chamber. To achieve this the pulse detonation combustion (PDC) has been investigated since the 1940s. The main idea here is to begin with a deflagration and accelerate the flame such that a detonation develops, sending a strong shock wave downstream through the fuelled gas which ignites it. Afterwards the combustion chamber is purged and refuelled starting the cycle again. The major drawbacks of this concept are the loss of efficiency due to the deflagration-detonation-transition and the material stress emerging from the high pressure peak of the shock wave.

Several approaches like the rotating detonation combustion (a PDC running in circles) tried to overcome these issues with varying success but there is another possibility to approximate CVC: homogeneous auto-ignition. This can be attained by filling the combustion chamber with a fuel-air mixture that is stratified such that it ignites simultaneously everywhere. This leads to a much weaker and broader pressure wave travelling downstream and being reflected as a suction wave at the open end due to acoustic reasons. This suction wave than refills the combustion chamber with a fresh air buffer to purge some of the hot exhaust gas and than starting the cycle anew as stratified fuel is taken in. This concept, as depicted in Figure 1.2, is called shockless explosion combustion (SEC) and is original to the Collaborative Research Centre (CRC) 1029, funded by the German Research Foundation (DFG), which this thesis' author was part of. For two four-year phases the project A03 of this CRC was dedicated to investigating the SEC through simulations and real-life experiments. A brief overview of the efforts and achievements of the first phase will be given in Section 1.1 while Section 1.2 will introduce the reader to the simulation-related questions of the second phase which this thesis will be about.

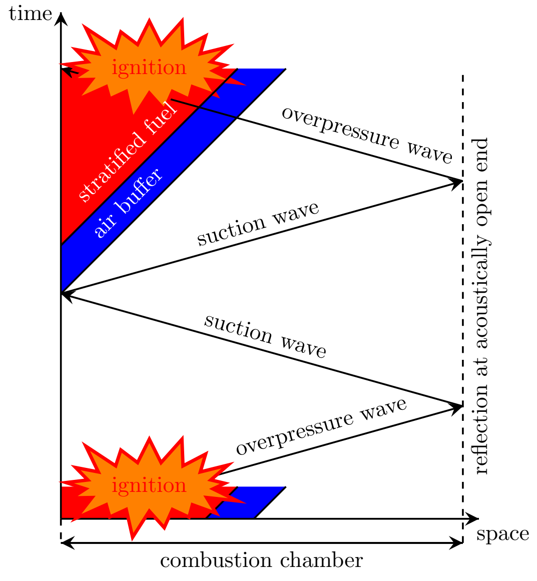

Figure 1.2: Schematic representation of an SEC cycle in a diagram over space and time. A pressure waves travels down the combustion chamber reflecting at the open end, refuels the chamber and the mixture auto-ignites again.

1.1 First Phase

Beginning in 2012 the first phase of project A03 of CRC 1029 established an initial and general understanding of the SEC process, its requirements, challenges and benefits (see [10], [11] and [7]). Alongside a real-life experimental set-up, that was able to produce a single-shot quasi-homogeneous auto-ignition by the end of the phase, a simulation software was developed by Berndt to prove the concept of the SEC and guide the way for the experiments. A short overview of its possibilities and methods is given here, for more information the reader is referred to [7].

The SEC-code's design is chosen carefully to meet the special requirements of the task. It solves the one-dimensional reacting Euler equations

$$\frac{\partial}{\partial t}\begin{pmatrix}\varrho\\ \varrho u\\ \varrho E\\ \varrho Y\end{pmatrix} = -\frac{\partial}{\partial x}\begin{pmatrix}\varrho u\\ \varrho u^2+p\\ (\varrho E+p)u\\ \varrho Y u\end{pmatrix} + \begin{pmatrix}0\\ 0\\ \varrho\dot{E}_{chem}\\ \varrho\dot{Y}_{chem}\end{pmatrix}, \tag{1.1}$$

where the following notation is used throughout this work: t denotes time, x first (axial) spacial direction, ϱ density, u axial flow velocity, E total specific energy, p pressure, Y the vector of species mass fractions, $\dot{E}_{chem}$ and $\dot{Y}_{chem}$ are the energy and species mass fraction source terms from chemical reaction. The ideal gas law is provided to close (1.1)

$$\begin{aligned} p &= \varrho R_s T,\\ \varrho E &= \frac{1}{2}\varrho u^2 + \varrho e,\\ e &= \int_{T_0}^{T} c_V(\theta, Y)\, d\theta + e_0(Y) \end{aligned} \tag{1.2}$$

with R_s being the specific gas constant, T temperature, e internal energy, c_V specific heat-capacity at constant volume, e_0 energy stored in chemical bonds at some reference temperature T_0. In [7] and the SEC-code e_0 is treated as zero. Within the scope of the thesis at hand this fact will have no impact on the considered concepts and will hence be omitted from discussions. Please consult the original thesis for details on this idea.

The Euler equations (1.1) with the equations of state (1.2) model an ideal, reactive, compressible but inviscid fluid flow in one spacial dimension, which is a good approximation of the vital processes taking place in the combustion chambers once the SEC runs. The one-dimensionality is justified because these chambers, also called SEC-tubes, are typically shaped like cylinders with small diameter of 1 to 4 cm compared to their length of about 1 m. Of course all flow properties must be understood as cross-sectional averages. This cross-sectional area A may also vary over x thanks to a feature of the code which enables the quasi one-dimensional solution of (1.1) as will be described in Subsection 3.2.2.

To solve (1.1) the SEC-code uses a finite volume method (FVM), which is per se conservative, in conjunction with common flux, limiter and reconstruction methods (see [44] or [60] for a very detailed insight to FVM). Let

$$q(x,t) := (\varrho, \varrho u, \varrho E, \varrho Y)^T$$

be the vector of conserved flow quantities as a continuous function of space x and time t and $f(q(x,t))$ be the flux function of the Euler equation

$$f(q(x,t)) := (\varrho u, \varrho u^2 + p, (\varrho E + p)u, \varrho Y u)^T.$$

We also define x_i, $x_{i\pm1/2}$ and Δx for all $i \in \{1, ..., N\}$ as the cell midpoints, interfaces and width, respectively, of the spacial grid over the simulation domain and t_k and Δt^k for all $k \in \mathbb{N}$ as the discrete time levels and current adaptive step width. Henceforth, we assume $i \in \{1, ..., N\}$ and $k \in \mathbb{N}$, unless otherwise indicated, and omit "$\forall i \in \{1, ..., N\}$" and "$\forall k \in \mathbb{N}$" in equations. Introducing the vector of cell-averaged integral conserved flow quantities

$$Q_i^k := \frac{1}{\Delta x} \int\limits_{x_{i-1/2}}^{x_{i+1/2}} q(\zeta, t_k)\, d\zeta,$$

we formulate the main idea of FVMs by

$$\frac{d}{dt} \Delta x Q_i^k = f(q(x_{i-1/2}, t_k)) - f(q(x_{i+1/2}, t_k)). \tag{1.3}$$

This means that the conserved quantity within $[x_{i-1/2}, x_{i+1/2}]$ can only change due to its flux over the grid cell boundaries. The time update of the solution to (1.1) implemented in the SEC-code is realised as

$$Q_i^{k+1} = Q_i^k - \frac{\Delta t^k}{\Delta x}(F_{i+1/2}^{k+1/2} - F_{i-1/2}^{k+1/2}), \tag{1.4}$$

where $F_{i-1/2}^{k+1/2}$ is the numerical form of $f(q(x_{i-1/2}, t_{k+1/2}))$. Usually, much effort is taken to find a flux function meeting special requirements. As one constant value Q is assigned to each grid cell, a Riemann problem arises at every cell interface. Accordingly, a Riemann solver was to be implemented. For the distinct applications of the SEC-code, the well-established HLL flux [29] with Einfeldt's correction [23] was a good choice. It only had to be adjusted to multi-species flows [7]. To achieve second order accuracy, this flux function is evaluated using an estimation of the flow states at $t_k + \Delta t^k/2$ and $x_{i\pm1/2}$ referred to as reconstruction. The spacial and temporal approximation of q is here piecewise linear, yielding the so-called MUSCL-Hancock scheme (see [60], section 13.4). As recommended for MUSCL-type schemes limiter functions are provided, enhancing the stability of the approximation at the interfaces in the presence of shock waves.

Chemistry and fluid dynamics are handled separately via the second-order operator splitting method Strang splitting [53]. This means, the fluxes or source terms, respectively, from the different parts of the process are computed sequentially: e.g. first the chemistry time update is calculated for half a time step width, then a full gasdynamics time update is done and finally, another half chemistry step, yielding a scheme like $Q_i^{k+1} = \mathsf{Chemistry}_{\Delta t^k/2}(\mathsf{Gasdynamics}_{\Delta t^k}(\mathsf{Chemistry}_{\Delta t^k/2}(Q_i^k)))$. This idea of splitting the solution process into independent steps makes modularised implementation possible as well as an acceleration of the computations. For convenience, update steps like (1.4) are formulated only with respect to the current operator, omitting fractional time stepping due to the splitting throughout the following chapters.

Beside the inclusion of complex, realistic chemistry, a simplified perfect gas kinetic model was developed in [8] to enable fast studies of the general processes governing the SEC. In this thesis it will be referred to as "3-species ignition delay kinetics" because it works with only three generic species representing a fuel, a radical, and a product species. The reaction from fuel to radical serves as an ignition delay clock, whereas energy is released during the transition from radical to product. This is also the only reaction path possible within this model. The reaction rate constants r have the general form of Arrhenius equations $r \sim \exp(1 - E_A/T)$, E_A being the activation energy and T the temperature (see [35], section 10.2 for a theoretical insight to this assumption). Ignition delay time and heat release rate were calibrated to imitate the behaviour of a realistic fuel igniting in one stage. This scheme was also extended to implement a two-stage ignition using five species. The models introduce reference values coming from hot gas. Whenever these kinetics are used in the following, the flow properties are understood as non-dimensional with reference values: $p_{ref} = 10^5$ Pa, $T_{ref} = 1000$ K, $x_{ref} = 0.8$ m and $t_{ref} = 1^{-3}$ s.

As common for FVM boundary conditions are imposed via ghost cells: additional grid cells adjacent to the computational domain which are computed according to the condition that is to be fulfilled. Currently, they allow for reflecting walls, zero-gradient, expansion into an infinitely large plenum chamber with fixed pressure and periodic boundaries as well as simply presetting any (time-dependent) state.

In the first phase of the CRC 1029 one was able to show that, once it is started, the SEC is a promising concept - not only in simulations but also in reality. Suggestions for possible fuel mixtures, axial tube radius variations and operation temperatures have been concluded from the SEC-code. The latter has been validated via well-known model problems and proven to meet the demand for SEC-specific accuracy, stability and efficiency. This left us with further questions pointing towards the realisation of the SEC within a gas turbine and a handy software foundation to tackle them in the second phase.

1.2 Second Phase, Outline and Beyond

After the elementary work of the first phase, the next step was to investigate the benefits, challenges, and overall possibilities of an implementation of the SEC in a gas turbine. This aim includes studying the starting process of such a machine as well as the interaction of multiple SEC-tubes with adjacent plenum chambers and each other when it is operating.

A first approach towards simulating the influence of the periodic fluctuations from the SEC-tube on a turbine plenum was made in the first phase using the feature of axial variation of the cross-sectional area. Albeit, such a set-up is unemployable when researching the interaction of multiple SEC-tubes. Therefore, a new feature was needed combining multiple quasi one-dimensional domains and coupling them in a network model. The preliminary work of [59] has been extended and improved for this thesis. The current implementation is described in Chapter 2 in addition to a simulation series of different placements of combustions chambers along the plenums.

Since the SEC-code was also used by partner projects some changes stem from the requirements of other groups' research. The largest of these adaptations is the implementation of an adaptive remeshing algorithm called "moving mesh method" as explained in Chapter 3. This algorithm is based on the idea of keeping the number of grid cells constant while changing their width and thereby resolving regions of high gradients better than others. Inherently, it is best used with travelling waves which we find plenty in the applications of the SEC-code. It is also very suitable to keep the computational cost from complex chemistry at bay which was one of the most relevant demands from our partner project A08.

A conventional gas turbine is started by accelerating the shaft of turbine and compressor with an auxiliary device. Then one ignites the deflagration flames in the combustion chambers and slowly raises the continuous fuel supply until the turbine reaches the desired speed. For an SEC gas turbine the proposed starting process begins just the same. However, when a predefined goal pressure is reached in the combustion chamber the fuelling valve operation is changed to first insert an air buffer and then inject a stratified fuel profile to create a homogeneously auto-igniting fuel package which than starts the SEC cycle. Hence, the investigation of this process requires the simulation of a deflagration flame and consequentially molecular transport and turbulence. These features have been realised and described in Chapter 4.

To simulate a simple deflagration flame, the kinetic model described in Section 1.1 has been stripped off its ignition delay by cancelling the radical species and letting the fuel species release energy right away when reacting to product species. Thereby, we get a perfect gas model we will call "2-species Arrhenius kinetics". As this model is made for starting the SEC from a deflagration flame its point of reference is gas at room temperature, generating a slightly different set of reference values than the one of the 3-species ignition delay kinetics, which is designed for evaluation of an already working SEC: $p_{ref} = 10^5$ Pa, $T_{ref} = 300$ K, $x_{ref} = 1$ m, $t_{ref} \approx 3.4 \times 10^{-3}$ s. Both kinetic models were also augmented by constant molecular transport coefficients: dynamic viscosity $\mu = 5 \times 10^{-5}$ Pa·s, thermal conductivity $\kappa = 0.2$ W/(m·K) and mass diffusion coefficient $D = 4 \times 10^{-5}$ m^2/s. All values have been chosen to lie within a range reasonable for gases like hydrogen or methane at high temperatures (cf. [24, 19]) and are non-dimensionalised with the respective reference values. There is also one version of the “2-species Arrhenius kinetics” with different transport coefficients obeying a thickened flame ansatz yielding $\mu = \kappa = D = 0.009$. This approach, dating back to [12], broadens the reaction front of a flame while keeping the flame speed to be able to resolve the usually very thin flame front within a simulation of bigger framework with a comfortable number of grid cells.

Since the CRC 1029 was terminated after the second phase but a lot of questions still remain the SEC will be studied further in a different scientific context. The next step will be the full simulation of an SEC gas turbine starting and operating at different stages of work load. The foundation for this goal is lain with the precedent work of phase one, research from partner projects as well as project partners and this thesis.

Chapter 2

One-Dimensional Network Model

The basic SEC-code by Berndt (described in detail in [7]) was designed to simulate a one-dimensional approximation to a single SEC-tube for the study of operating points, chemical models and the fuelling process in general. As already stated in Chapter 1, a real-life gas turbine works with more than only one combustion chamber. Usually 5 to 6 are bundled in an annular array in the combustor section. The combustion chambers suck air from an upstream compressor and fire into a plenum connected to the turbine. In order to simulate and examine such a one-dimensional network configuration it was necessary to extend the basic SEC-code. A first step towards this goal has already been taken in [59]. The code was improved since then, enhancing the computational efficiency as well as the accuracy and flexibility. Section 2.1 will be about the details of the current implementation. Differences from the first version will be highlighted wherever relevant. Simulation results, expanding the study from [59] researching the placement of combustion chambers along the plenum, are shown in Section 2.2. Conclusions and a short outlook on possible future work are given in Section 2.3.

2.1 Implementation

The original SEC-code solves the one-dimensional reactive Euler equations (1.1) in one computational domain. Thus, it can simulate chemistry and inviscid fluid dynamics in a single SEC-tube. To be able to look at the phenomena of interaction between multiple combustion chambers with and via one or two plenum chambers an extension was added such that every component can be model by an own computational domain. Interactions are realised through suitable boundary conditions coupling domains in a two-dimensional framework. In [59] it has already been shown, that such an approach is possible and desirable, yielding an efficient tool for configuring a multi-tube test rig as well as developing control algorithms for a working SEC gas turbine. Most of the modelling effort actually lies in the coupling method requiring pseudo two-dimensional treatment of boundary conditions and a posteriori flux correction to ensure conservation. These are also the main fields of improvement

```
for $j$ = 1 to $n_{domains}$ do
    set up domain $j$ (see Appendix B.4 for instructions);
end
while $t < t_{end}$ do
    for $j$ = 1 to $n_{domains}$ do
        calculate the maximal time step size $\Delta t_j^k$;
    end
    set global time step size $\Delta t^k = \min_j(\Delta t_j^k)$;
    for $j$ = 1 to $n_{domains}$ do
        current domain number = $j$;
        store edge cells;
        solve governing equations through Strang splitting;
        if interacting domain number > $j$ then
            restore edge cells;
        end
        if two-dimensional interaction then
            begin second dimension boundary handling:
                add lateral momentum of 0 to $\mathbb{Q}$;
                compute upper and lower boundaries;
                reconstruct states at upper and lower cell interfaces;
                compute flux over upper and lower cell interfaces;
                update state vectors using FV scheme (1.4);
                rotate flux for flux correction;
                store flux;
                remove lateral momentum;
            end
        end
    end
    begin flux correction:
        for $j$ = 1 to $n_{domains}$ do
            if interacting domain number > $j$ then
                advance interactive cell with stored flux;
            end
        end
    end
    $t \leftarrow t + \Delta t^k$;
end
```

Figure 2.1: SEC-code workflow with multiple domains

in comparison to the preceding work.

Figure 2.1 shows the full workflow of a network simulation. Roughly, the procedure is as follows: After setting up all domains separately, fixing an order, the simulation is started. The time step size Δt^k equals the minimum of all maximal time step sizes for all domains. This is an improvement, for in the first version a global time step size was defined manually for the complete simulation time. Hence, the new implementation is more robust and efficient. Using this step size Δt^k each domain's solution gets evolved in time exploiting Strang splitting for chemistry, molecular transport and gasdynamics. If the current domain is coupled to another one in a two-dimensional fashion, the second dimension is evolved in time afterwards as in common dimensional splitting schemes. Finally, the fluxes between the domains are corrected to ensure conservation of mass, momentum, energy density and species. The details of domain coupling and flux correction are described in Subsection 2.1.1. It will be explained using the example of multiple combustion chambers meeting a plenum as it was designed for this case but of course the configuration could be anything. Subsection 2.1.2 provides information about the turbine and compressor models implemented for the usage in this network configuration.

2.1.1 Coupling of Domains

Simulating multiple quasi one-dimensional domains is easily done as long as they are separated from each other: simply loop over the simulation process with different configurations representing the current domain. Conversely, this means that the most interesting part is the region where two domains meet. To get into the subject, let us consider the simplest possible arrangement first: Suppose, we have two domains being aligned like they were just one big domain cut in halves. We start the simulation by advancing the solution of the first domain adding cells from the second one as boundary condition ghost cells to one end. Now, store the flux over this boundary interface and update the second domain using this flux for the corresponding interface. This sounds straightforward which it actually is for this case.

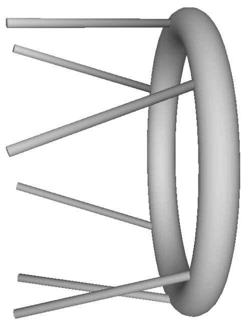

Figure 2.2: A three-dimensional model of the torus-shaped turbine plenum with six slanted combustion chambers.

The complexity of the current implementation results from the aspiration to connect domains not only on a one-dimensional line but in a two-dimensional plane. Thereby, multiple combustion chambers can meet a plenum even being slanted. A first approach towards this issue was described in [59] and the general idea remained the same, changing only the minutiae of execution. To connect two domains in a two-dimensional way, we obviously need a second spacial dimension, i.e., boundary conditions and a solution update step in the lateral direction y. Naturally, this could be achieved by extending the code to two dimensions in general, necessitating two-dimensional grid management, evolution of all grid cells in two dimensions, lateral momentum and more. To avoid this computational overhead, keeping the simple one-dimensional structure for efficiency, a trick is needed enabling two-dimensional interaction of essentially one-dimensional domains. Consider the case of three SEC combustion chambers being connected to a turbine plenum. As depicted in Figure 2.2, the plenum is modelled as a torus employing periodic boundary conditions, while the combustion chambers largely remain straight cylinders. For the sake of simplicity let us assume the junctions are orthogonal for now. An SEC-tube might be connected to more than one plenum cell. We require integers here as we do not involve complex cut-cell algorithms or the like. Therefore, we need to average the flow properties over these interacting cells to gain a boundary condition ghost cell. Also, the momentum stored in the state vector of the plenum cells will be in the lateral direction seen from the combustion chamber. The lateral momentum of the plenum cells which we need for the combustion chamber boundary is supposed to be zero. Consequentially, the ghost cells will have zero momentum. Here is where slanting domains adds complexity but we will postpone this issue a little further. At this point, we already see a problem, when coping with the plenum boundaries as we have averaged plenum cells to compute one flux over one interface but for the plenum cells we need the flux over multiple interfaces. The finer resolution of the interaction interface calls for the favouring of the flux calculated from the plenum domain instead. So the possible workflow described in the case of aligned domains changes. We now restore the edge cell of the combustion chamber which interacts with the plenum and advance it again later when the flux from the plenum is available. This approach is common in the handling of multi-dimensional coarse-fine grid interfaces (see e.g. [6]).

Now, let us turn to the plenum. After the solution was updated in the axial direction (following the perimeter of the torus in this case) we rearrange our view on the plenum as a two dimensional domain with only one cell in the lateral direction. Figure 2.3 illustrates this picture with already added ghost cells for the lateral direction. As the plenum is a closed torus, most cells represent a reflecting wall boundary condition. The only exceptions are the ones adjacent to the plenum cells which interact with the SEC-tubes. These are copies of the downstream state of the corresponding combustion chamber domains. Reinterpreting the combustion chambers' axial momentum as lateral for the plenum, we find ourselves in a situation similar to advancing the solution in a one-dimensional domain. We merely need to

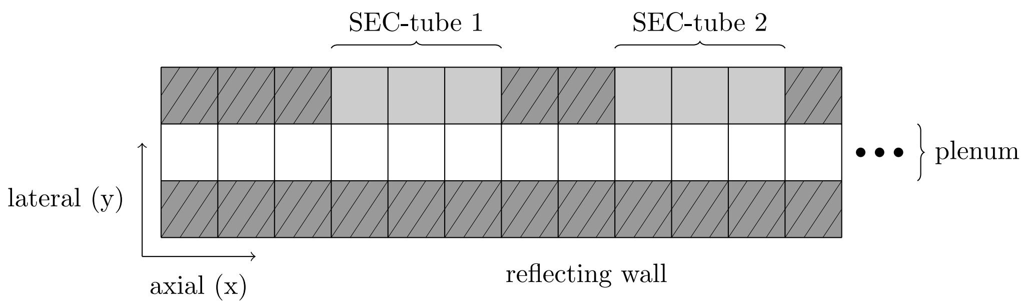

Figure 2.3: Interaction of SEC-tubes and a plenum. White cells show the pseudo one-dimensional domain of the plenum, dark grey cells are solid wall boundary cells and light grey cells are copies of SEC-tubes' states.

repeat the flux computation and finite volume scheme (1.4) for every plenum cell in the second direction. As we stay with our one-dimensional view the lateral momentum produced in the plenum must dissipate and, therefore, is converted implicitly to internal energy by keeping the energy density but setting the lateral momentum to zero.

If we now consider slanted combustion chambers we can transform their momentum vector to the plenum's coordinate system, keep the axial part and dissipate the lateral. Generally speaking, this was the approach applied in [59] based on the groundwork of Berndt.

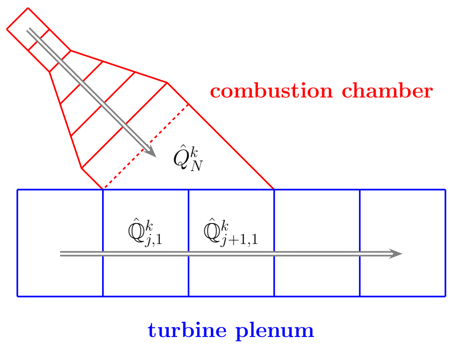

Figure 2.4: Example of a slanted combustion chamber joining the turbine plenum involving two cells. Gray arrows indicate the direction of quasi one-dimensional domains. $\hat{Q}$ and $\hat{\mathbb{Q}}$ are two-dimensional states of the combustion chamber or plenum, respectively.

For the thesis at hand the process was refined to include a more realistic view on the slanted configuration since e.g. the interface area actually changes with the angle. Please keep in mind, that we still require the junction to be resolved by an integer number of cells from the plenum's side. We will call it s. Figure 2.4 depicts a quasi one-dimensional SEC-tube meeting the plenum in a non-orthogonal

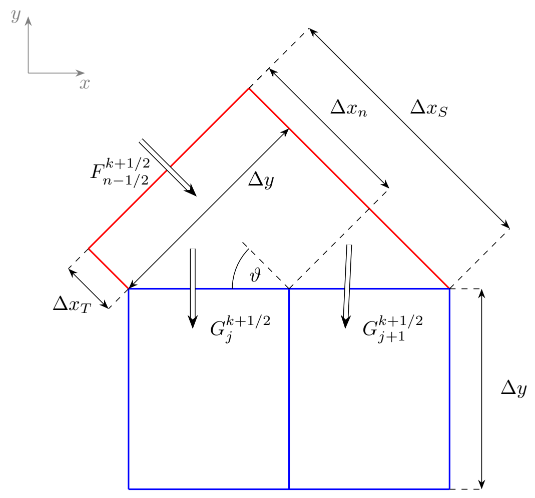

Figure 2.5: Zoom into the junction of Figure 2.4 showing the three coupled grid cells. Fluxes over cell interfaces $F^{k+1/2}_{N-1/2}, G^{k+1/2}_{j}, G^{k+1/2}_{j+1}$, angle ϑ, combustion chamber's cell width Δx_T, last cell's average width Δx_N, long side length Δx_S and plenum diameter Δy are marked.

fashion. To avoid a confusing amount of indices we will write combustion chamber states as Q^k_i and plenum states as $\mathbb{Q}^k_{j,i}$ using two subscripts for the latter, denoting axial and lateral direction, respectively. We also indicate a two-dimensional state vector by giving it a hat: $\hat{Q}^k_i$ or $\hat{\mathbb{Q}}^k_i$, respectively. Moreover, we stick to the one combustion chamber as an instance for all possible junctions. Let us work through the second dimension boundary handling once again looking at all the details of the current implementation. Figure 2.1 shows us the path we have to follow. At first, all plenum cells get a lateral momentum component of zero, making them fully two-dimensional. Afterwards the ghost cells for the y-direction are computed. As before the ones representing the combustion chamber are s copies of the corresponding domain's state. Since we consider the case of a combustion chamber interacting with a turbine plenum this will be the downstream cell's state. Suppressing an index to indicate that the number of cells N belongs to the combustion chamber to avoid double subscripts we write Q^k_N. Remember that we restored this value, so k is the correct time index. To fit in with the current coordinate system of the plenum, these state vectors need to be transformed by rotation. Hence, they gain a lateral momentum component, too

$$Q_N^k = \varrho_N \begin{pmatrix} 1 \\ u_N \\ E_N \\ Y_N \end{pmatrix} \longrightarrow \hat{Q}_N^k := \varrho_N \begin{pmatrix} 1 \\ -u_N \cos(\vartheta) \\ -u_N \sin(\vartheta) \\ E_N \\ Y_N \end{pmatrix}. \tag{2.1}$$

Here ϑ is the angle between the x-axis of the combustion chamber and the one of the plenum as can be seen in Figure 2.5 which is the close up sketch of the interacting cells. Regarding such a slanted combustion chamber cell we find that it is no longer a square but a trapezoid. Now, the states are reconstructed at the interfaces. These parts are new for the current version of the domain coupling. For the plenum cells this is a straightforward adaptation from the axial direction, simply exchanging Δx for Δy, the plenum diameter

$$\hat{\mathbb{Q}}_{i,1/2}^{k+1/2} = \hat{\mathbb{Q}}_{i,1}^k - \frac{\Delta y}{2}\frac{\partial \hat{\mathbb{Q}}}{\partial y} + \frac{\Delta t^k}{2}\frac{\partial \hat{\mathbb{Q}}}{\partial t}, \quad \forall i \in \{j, ..., j+s-1\}.$$

For the combustion chamber cell, we need to redefine Δx as the downstream interface is not parallel to the upstream interface. When computing the volume of this cell through $V_N = A_N \cdot \Delta x_N$ it is clear from Figure 2.5, that Δx_N has to be the average of the short and the long side of the trapezoid, i.e., $\Delta x_N := \frac{\Delta x_T + \Delta x_S}{2}$. Hence, we can reconstruct the state at the cell's edge using $\frac{\Delta x_N}{2}$

$$\hat{Q}_{N+1/2}^{k+1/2} = \hat{Q}_N^k + \frac{\Delta x_N}{2}\frac{\partial \hat{Q}}{\partial x} + \frac{\Delta t^k}{2}\frac{\partial \hat{Q}}{\partial t}.$$

Another option that is implemented is the reconstruction of the combustion chamber's state at each partial face that coincides with a face of the turbine plenum cell. In this case, we would need to define a Δx_{N_i} for each of the s faces which should satisfy $\Delta x_{N_i} = \chi_i - (x_{N-1/2} + \frac{\Delta x_N}{2})$, with χ_i being the midpoint of face i, and then compute

$$\hat{Q}_{N+1/2,i}^{k+1/2} = \hat{Q}_N^k + \Delta x_{N_i}\frac{\partial \hat{Q}}{\partial x} + \frac{\Delta t^k}{2}\frac{\partial \hat{Q}}{\partial t}, \quad \forall i \in \{j, ..., j+s-1\}.$$

Now, to advance the plenum cells which interact with the combustion chamber in time, we have to use the lateral flux g from the two-dimensional reactive Euler equations

$$g(q) = \begin{pmatrix} \varrho v \\ \varrho uv \\ \varrho v^2 + p \\ (\varrho E + p)v \\ \varrho Y v \end{pmatrix}.$$

Here, v denotes the lateral velocity. Its numerical counterparts $G_i^{k+1/2}$, $\forall i \in \{j, ..., j+s-1\}$ can readily be calculated from reconstructed values by an already available HLLE-flux function written for perfect gases in three dimensions. As there is no flux opposite to the interaction face due to a reflecting wall boundary, the update step is computed by

$$\hat{\mathbb{Q}}_{i,1}^{k+1} = \hat{\mathbb{Q}}_{i,1}^{k} + \frac{\Delta t^k}{\Delta y} G_i^{k+1/2}, \quad \forall i \in \{j, ..., j+s-1\}.$$

Employing the technique described in Subsection 3.2.2, area variation is also included for the lateral time stepping. As before, the lateral moment is set to zero afterwards to meet the dissipation assumption.

What remains is to transform the s two-dimensional fluxes from the plenum coordinate system into one one-dimensional flux for the combustion chamber.To this aim, we reconsider the rotation applied to the combustion chamber state Q_N^k in (2.1) and insert this general state into the lateral flux function g yielding

$$g(\hat{q}) = \begin{pmatrix} \varrho u \sin(\vartheta) \\ \varrho u^2 \cos(\vartheta)\sin(\vartheta) \\ \varrho(u\sin(\vartheta))^2 + p \\ (\varrho E + p)u\sin(\vartheta) \\ \varrho Y u \sin(\vartheta) \end{pmatrix}.$$

To regain the form of the axial two-dimensional flux f we compute

$$\hat{g}(q) = \begin{pmatrix} g(q)_1/\sin(\vartheta) \\ g(q)_2/(\cos(\vartheta)\sin(\vartheta)+p) \\ (g(q)_3 - p)/(\sin(\vartheta))^2 \\ g(q)_4/\sin(\vartheta) \\ g(q)_5/\sin(\vartheta) \end{pmatrix} .$$

Actually, we drop the lateral momentum flux as the dissipation via conversion to internal energy is already done by keeping the energy density flux. Now, using the transformed fluxes $\hat{G}_i^{k+1/2}$, the combustion chamber cell at the junction to the plenum can be advanced in time through

$$\hat{Q}_N^{k+1} = \hat{Q}_N^k - \frac{\Delta t^k}{\Delta x_N} \left(\frac{1}{s} \sum_{i=j}^{j+s-1} \hat{G}_i^{k+1/2} - F_{N-1/2}^{k+1/2} \right) .$$

Although we have always spoken about combustion chambers and plenums, this has only been an illustrating example. The described method for coupling domains is valid in general and can be used with any configuration of domains.

2.1.2 Turbine and Compressor Model

In the considerations so far, the turbine plenum has been closed. For a simulation of multiple SEC-tubes firing into that plenum for a long time we would run into a problem as of course a real-world gas turbine does not keep the flow parcels from the combustion chambers but pass them on to a turbine driving its blades thereby. Hence, we need a model of this process. In [59] a first version has already been implemented which was now extended to simulate not only the mass loss through a turbine but also the gain through a compressor.

A compressor or turbine uses the energy of rotating blades to enhance the pressure of a fluid or the energy of an expanding fluid to drive a rotor, respectively. Since the focus of this work is the interaction of combustion chambers and plenums, it suffices to reduce the turbomachine to its net effects on the flow, yielding a zero-dimensional model, instead of fully simulating it. For now, we are only interested in the mass flux coming from or vanishing into the compressor or turbine, respectively. To express the working characteristics of a specified turbomachine, it is common practice to use non-dimensional quantities for mass flux $\dot{m}$ (dot notation emerges from classical engineering notation), rotational speed, temperature and pressure (see

[15], section 4.5). For temperature and pressure, one uses the ratios of outgoing T_{out}, p_{out} to incoming T_{in}, p_{in} flow property. For mass flux

$$\frac{\dot{m}\sqrt{R_s T}_{in}}{A_T p_{in}} \tag{2.2}$$

has proven useful. R_s denotes the specific gas constant, A_T the cross-sectional area of the turbomachine. As both are mostly constant over a certain configuration, one even reduces the expression to the so-called corrected (dimensional) mass flux

$$\dot{m}_{corr} := \frac{\dot{m}\sqrt{T_{in}}}{p_{in}}. \tag{2.3}$$

For a our simple turbomachine model, we assume the point of operation does not greatly change. Hence, we fix $\dot{m}_{corr}$ over one configuration, essentially depending on A_T. This gives us an equation for the mass flux by rearranging (2.3).

For the calculation of the mass flux to the turbine, we insert p_s and T_s, the stagnation pressure and temperature, respectively. We compute these from flow states in the turbine plenum using the following equations

$$p_s = p\left(1 + \frac{\gamma - 1}{2}\mathcal{M}^2\right)^{\frac{\gamma-1}{\gamma}},$$

$$T_s = T\left(1 + \frac{\gamma - 1}{2}\mathcal{M}^2\right)$$

with isentropic exponent γ. We assume that the fluid reaches the speed of sound, thus Mach number $\mathcal{M} = 1$. This is reasonable for turbomachines operating at full power only. Since we know $m = \varrho V$, V being the volume, and $\dot{m} = \frac{\Delta m}{\Delta t}$ the difference in density in a plenum $\Delta\varrho$ that is produced by the turbine within a time step Δt^k can be expressed by

$$\Delta\varrho = \dot{m}\frac{\Delta t^k}{A_P \Delta x}. \tag{2.4}$$

A_P denotes the cross-sectional area of the plenum. An update of the turbine plenum state thus looks as follows

$$Q_i^{k+1} = \frac{\varrho - \Delta\varrho}{\varrho} Q_i^k,$$

The minus sign emerges from our knowledge that gas expands into the turbine to drive the rotor.

Since we need the ingoing pressure and temperature for (2.3), the mass flux from the compressor uses a predefined pressure p_c and temperature T_c. (2.4) is still valid for the compressor but the new density is calculated by $\varrho_n = \varrho + \Delta\varrho$. We calculate the new pressure p_n and energy density $\varrho_n E_n$ of the plenum state by assuming an isentropic expansion, for which we know pV^γ is constant throughout the process. We call this constant c_e and compute

$$p = c_e \varrho^\gamma. \tag{2.5}$$

Inserting the ideal gas law $\varrho = \frac{p}{R_s T}$ into (2.5) for the compressor state we get

$$c_e = \frac{(R_s T_c)^\gamma}{p_c^{\gamma-1}}.$$

Now we use (2.5) again for p_n yielding

$$p_n = \frac{(R_s T_c)^\gamma}{p_c^{\gamma-1}} \varrho_n^\gamma$$

With this new pressure we compute the updated energy density by $\varrho_n E_n = \frac{p_n}{(\gamma-1)} + \frac{u^2}{2\varrho_n}$. The remaining compressor plenum quantities mass, momentum and species are updated through multiplication by $\frac{\varrho_n}{\varrho}$ as in the turbine plenum. Please note, that R_s is the specific gas constant of the gas mixture. According to [35], equation (16.3), R_s is calculated from

$$R_s = R_u \sum_{i=1}^{N_{spec}} \frac{Y_i}{M_i}$$

with R_u being the universal gas constant, N_{spec} the number of species and M_i the molar mass of species i. For non-dimensional quantities R_s cancels out due to division by reference values.

With these equations the mass flow from the compressor remains constant over time scaling only with the time step size. Therefore, it is designed to stop filling at a certain threshold. Although the turbine changes the mass flow according to the state within the plenum, a threshold is needed, too. This is due to the fact that we fixed the point of operation, so that the turbine would not stop working, even when the ambient pressure is reached in the plenum. The threshold values hence represent the maximum and the minimum output pressure, respectively.

2.2 Simulations

governing equations	quasi one-dimensional Euler
chemistry	3-species ignition delay kinetics
<u>domains</u>	
combustion chambers	number: 3, length: 1, radius: 0.025, widening towards turbine plenum
connection pieces	number 3, length: 0.05, radius: 0.075, tightening towards combustion chamber

compressor plenum	number: 1, length: 5, radius: 0.075, no variation
turbine plenum	number: 1, length: 5, radius: 0.1, no variation
initial values	
combustion chambers	stabilised cyclic SEC at ignition
connection pieces	$T = 1$, $p = 1, u = 0$, product species everywhere
compressor plenum	$T = 1$, $p = 1, u = 0$, product species everywhere
turbine plenum	$T = 2.6$, $p = 1, u = 0$, product species everywhere
boundaries	
combustion chambers	left: connection piece with pressure valve, right: 45° connection to turbine plenum
connection pieces	left: 45° connection to compressor plenum, right: combustion chamber with pressure valve
compressor plenum	left and right: periodic, top: reflecting wall, bottom: 45° connections to connection pieces at $(0.8, 2.5, 4.2)$, $(1.5, 2.5, 3.5)$, $(2, 2.5, 3)$ and $(1.5, 2.5, 4.5)$, resp.
turbine plenum	left and right: periodic, top: 45° connections to combustion chambers at $(0.8, 2.5, 4.2)$, $(1.5, 2.5, 3.5)$, $(2, 2.5, 3)$ and $(1.5, 2.5, 4.5)$, resp., bottom: reflecting wall
grid cells	combustion chambers: 600, connection pieces: 30, compressor plenum: 354, turbine plenum: 266
time steps	step size chosen automatically, snapshots stored at multiples of 4×10^{-3}

Table 2.1: Settings for network model simulations.

Of course the configuration introduced in this chapter has been simulated with the network feature. To outline the setting, Table 2.1 holds the important information to replicate the simulations. It includes three combustion chambers and two plenums - one for the upstream compressor, one for the downstream turbine. For stability reasons three connection pieces between compressor plenum and combustion chambers were added. They ensure the proper operation of the pressure valve which is used to fuel the combustion chambers. As this value is unity and equal to the compressor input pressure as well as the minimum turbine output pressure, the compressor plenum works as a transition chamber in this case. Due to this fact, we do not expect high amplitudes for pressure in the turbine plenum.

The plenums are set up as toruses with periodic boundary conditions and without cross-sectional area variation. The diameter for the turbine plenum was extracted from the studies in [59]. The stability of the SEC is less sensitive to the compressor plenum diameter which is hence set to be just sensibly smaller than the turbine plenum's. Both domains have the same length of 5 which corresponds to the perimeter of the toruses. The connection pieces and combustion chambers, respectively, are coupled to these domains via second dimension boundaries as described in Subsection 2.1.1. All non-interactive boundary cells represent a reflecting wall. Both plenums are started with $p = 1, u = 0$ and product species everywhere. For a first trial, also both plenums were initialised with $T = 1$ but for the presented simulation, the turbine plenum was preheated to $T = 2.6$, which was the mean temperature at $t = 20$ in the preliminary study. The resolution is set to a fifth of each radius.

The connection pieces interact with the compressor plenum at the upstream side and the combustion chambers at the downstream side. They are used to perform the tightening of the cross-sectional area from compressor plenum value to combustion chamber value. They are only as short as 0.05 but this intermediate step is crucial for the operation of the pressure valve as a suction wave travelling upstream would be weakened to much by the area variation if it was implemented in the combustion chambers before the pressure valve preventing a proper flushing and refilling of the SEC-tube. Hence, the valve is included in the interaction interface between combustion chamber and connection piece. The connection pieces are initialised just as the compressor plenum. Since they are aligned with the combustion chambers they are slanted by 45° and resolved by the same grid width.

The combustion chambers surely form the heart of the network and feature the most pronounced gas dynamical processes. Therefore, they are resolved quite high with 600 grid cells at a domain length of 1. The radius variation towards the turbine plenum is not separated for this direction as there is no sensitive valve to pay attention to. The upstream boundary is formed by the pressure valve of the corresponding connection piece while the downstream boundary interacts with the turbine plenum. The valve, will open when the pressure in the leftmost grid cell of the combustion chamber drops below the pressure of the adjacent grid cell of the corresponding connection piece. When opened the left boundary switches from reflecting wall to simple domain interaction and the leftmost combustion chamber cell is directly supplied with an air buffer first, then a stratified fuel profile which is configured to auto-ignite homogeneously in a single SEC-tube as simulated in the first phase of the CRC 1029. Since they are slanted by 45°, each combustion chamber domain ends with a trapezoidal grid cell of a much bigger width than the other cells equal to 0.1. Consequentially, SEC-tubes are actually a bit longer than 1. This rather big trapezoidal grid cell is also the main reason for differences between the simulation at hand and the ones carried out in [59] with a similar setting though without compressor plenum. Since the full starting process of the machine would go beyond the scope of this thesis, the combustion chambers where initialised right before ignition (when the radical mass fraction is at its peak value) of a stabilised SEC-cycle taken from the before mentioned preliminary study for this network simulation.

Beside the proof of concept for the network model described within this chapter, this series of simulations aims to study the influence of the combustion chambers' placement on the plenums' perimeters. One simulation was carried out with equidistantly distributed combustion chambers, two were set up with bundled tubes with minimal distance of 1 and 0.5. The forth is a more asymmetric distribution with two bundled combustion chambers at distance 1 and the third being separated as far as possible from both.

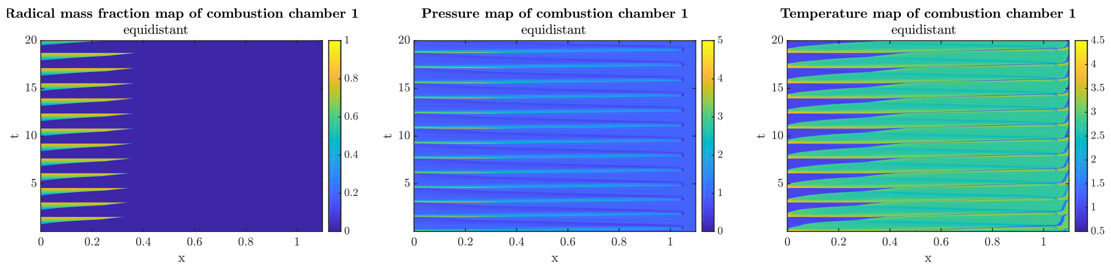

Figure 2.6: Equidistant case as representative example of radical mass fraction (left), pressure (middle) and temperature (right) over space and time for a combustion chamber in network simulation.

First of all, it is important to notice that the SEC works stably in all cases. The differences within the combustion chambers between the four placement cases are only minor for our setting. An exemplary insight is given in Figure 2.6 for the equidistant case and combustion chamber one - leftmost as seen from the turbine plenum. A slight difference in the cycle length is the most interesting point here because for a real-life gas turbine which runs very much longer than simulated and features more combustion chambers a shorter cycle might be advantageous in terms of efficiency. In this view, the asymmetric fourth positioning option would be best, followed by the equidistant one.

Figure 2.7 shows velocity, pressure and temperature over space and time for the compressor plenum. Due to its transitional character for this simulation series the amplitudes are very low. Nonetheless, we can detect tendencies towards some characteristic patterns for the distinct cases. Especially, the fully bundled positioning

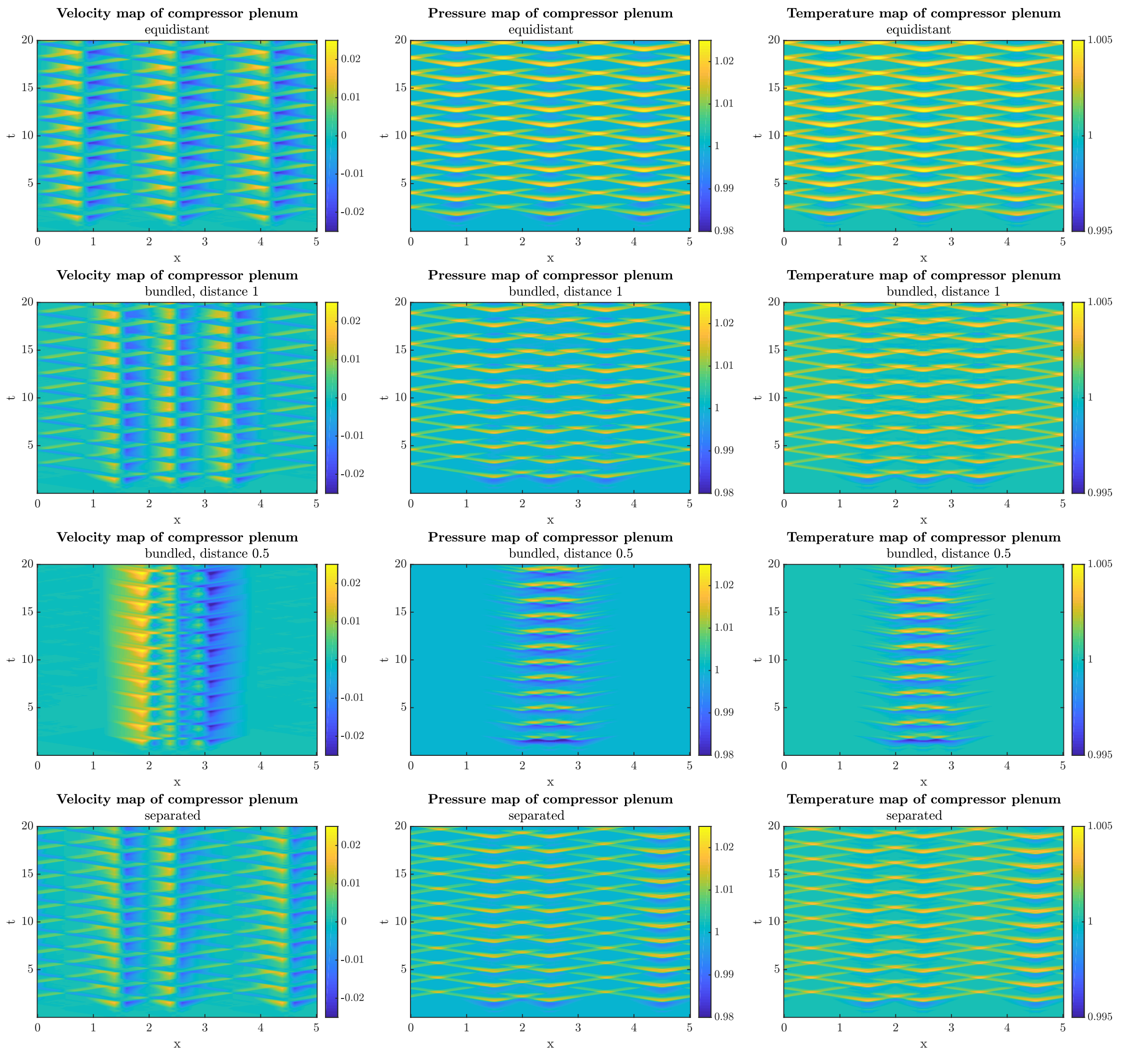

Figure 2.7: Comparison of velocity (left), pressure (middle) and temperature (right) map over space and time for compressor plenum in network simulation with different placement: equidistant (first row), bundled with minimal distance 1 (second row), bundled with minimal distance 0.5 (third row) and two bundled and one isolated combustion chamber (last row).

options two and three show weaker and more locally bounded perturbation waves. Strong locality of peaks can be disadvantageous in terms of material stress and must be considered seriously, when deciding for a design layout. Case four gives the impression of a mixture of bundled and equidistant case as the waves are weaker than the equidistantly distributed case but also wider spread over the full plenum length than in the fully bundled cases.

The most interesting part is the influence of the combustion chamber placement on the turbine plenum. As has been expected, the pressure peaks are not yet very high but the specific patters can be studied, nonetheless. From Figure 2.8 we can

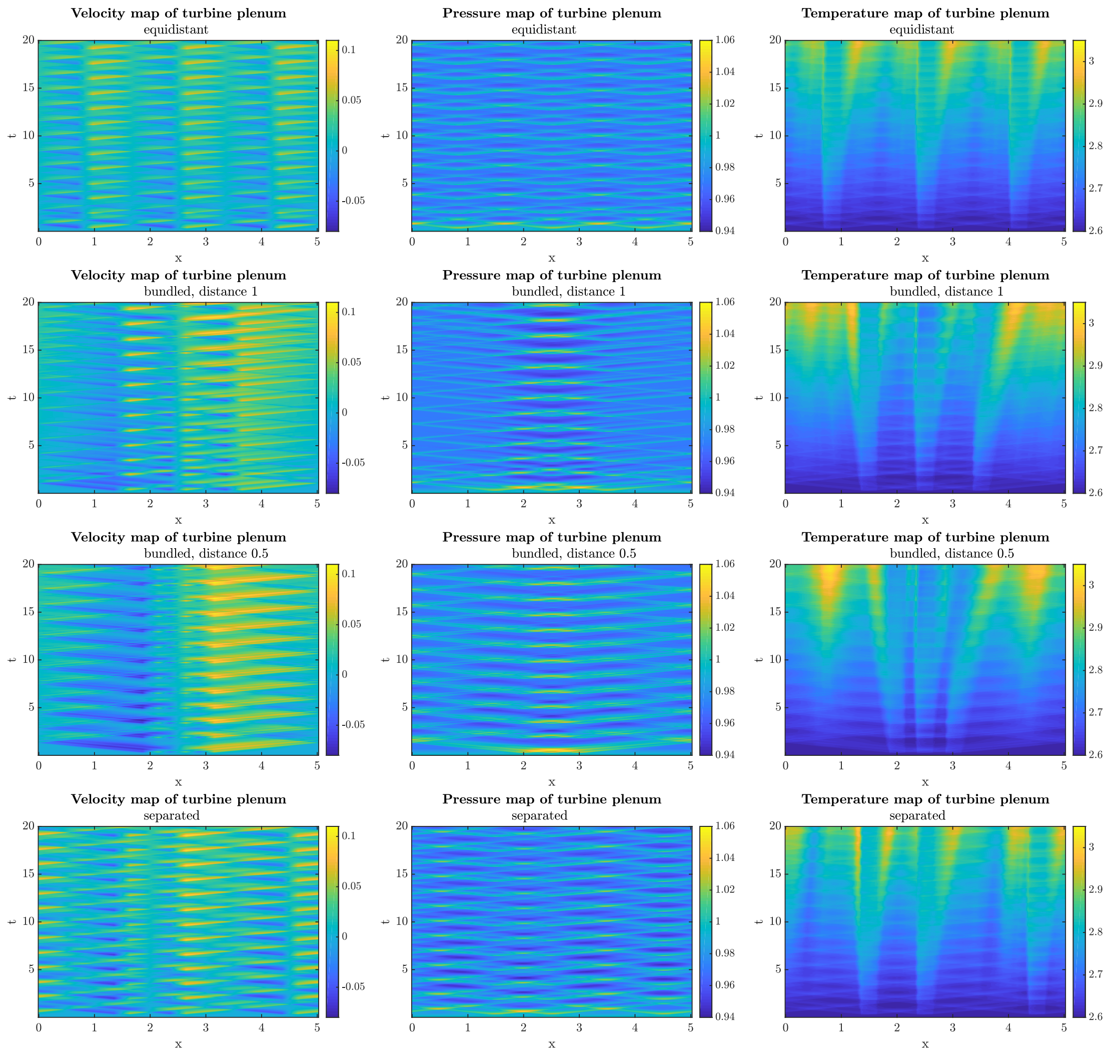

Figure 2.8: Comparison of velocity (left), pressure (middle) and temperature (right) for turbine plenum in network simulation with different placement: equidistant (first row), bundled with minimal distance 1 (second row), bundled with minimal distance 0.5 (third row) and two bundled and one isolated combustion chamber (last row).

deduce that the fully bundled cases feature higher pressure amplitudes. Although the more distant case two takes longer, both configurations bound the highest and lowest pressures in the plenum centre at $x = 2.5$, hence, establishing the same locality found in the compressor plenum. This shows best in Figure 2.10 where the pressure over space for the last time point $t = 20$ is depicted in the middle panel. For the other two cases the pressure peaks are lower, although case four again looks like a mixture between bundled and equally distributed placement as can be seen in the pressure function over time at the plenum centre in Figure 2.9.

The velocity shows a clear uptrend over time in Figure 2.9, which implies that the slanting of combustion chambers leads to a corresponding mean flow in the

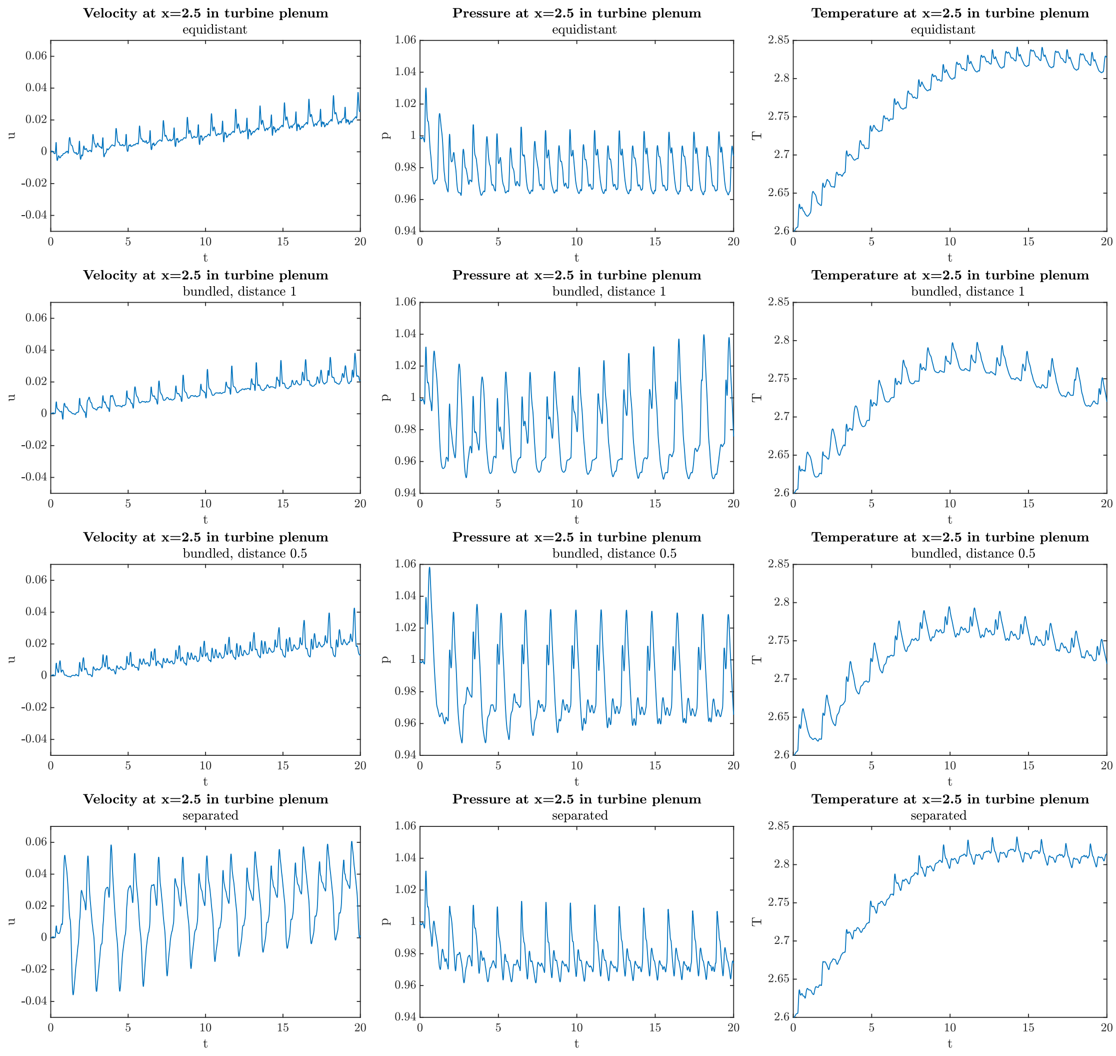

Figure 2.9: Comparison of velocity (left), pressure (middle) and temperature (right) over time at $x = 2.5$ for turbine plenum in network simulation with different placement: equidistant (first row), bundled with minimal distance 1 (second row), bundled with minimal distance 0.5 (third row) and two bundled and one isolated combustion chamber (last row).

turbine plenum as would be expected in a real-life experiment. Which might be most confusing is the comparably high amplitude in velocity for case four. Figure 2.8 and Figure 2.10 give us a clearer view here. The bundled cases have actually higher velocity amplitudes but only to the left and right of the centre $x = 2.5$. For case four the influence of the tube to the right is much smaller than the one of the left tube and of course the middle one at $x = 2.5$ resulting in stronger amplitudes at this location. This gives us an impression of how the tube placement and angle between combustion chamber and turbine plenum might play a role for the velocity distribution in the turbine plenum.

The described differences in the pressure and velocity map influence the temper-

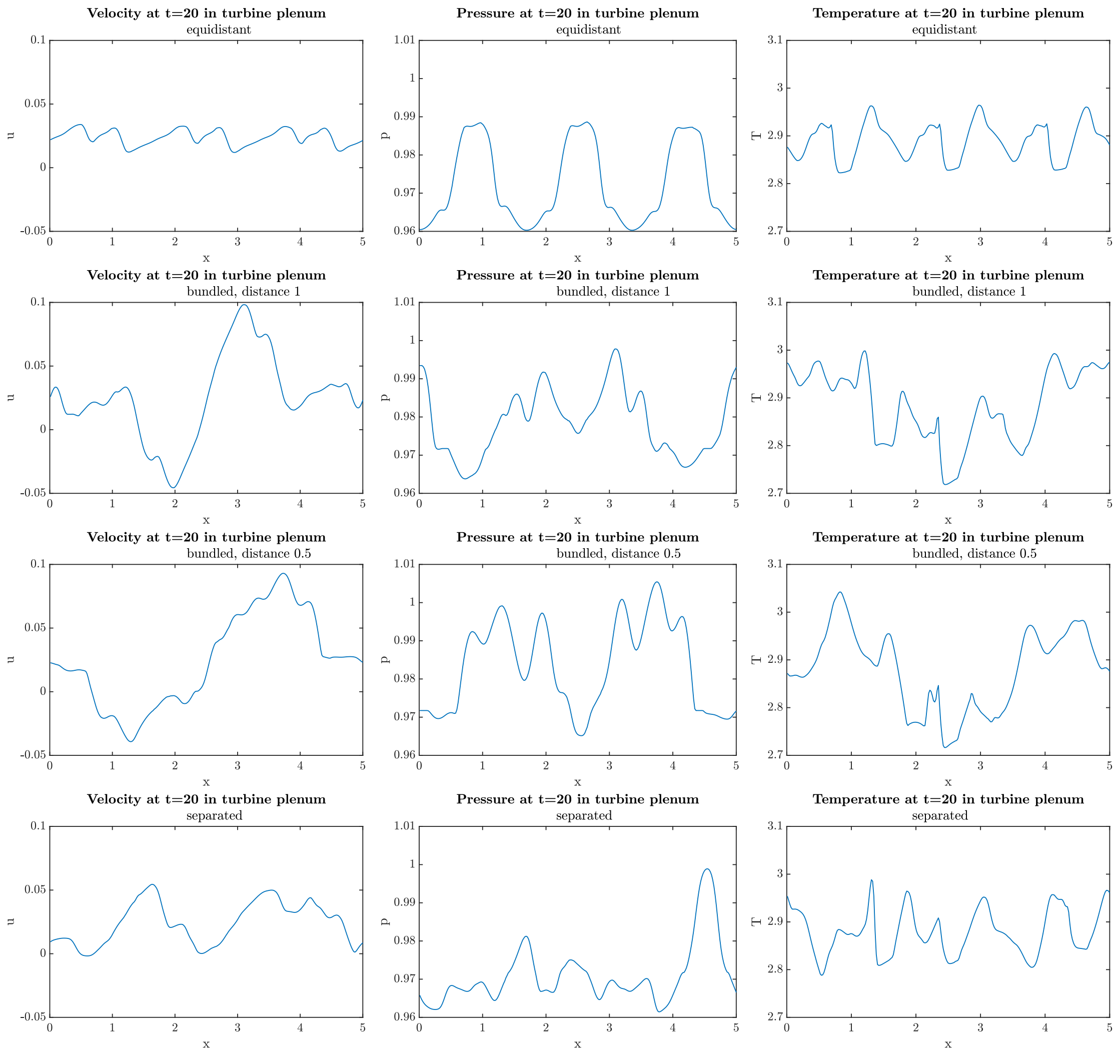

Figure 2.10: Comparison of velocity (left), pressure (middle) and temperature (right) over space at $t = 20$ for turbine plenum in network simulation with different placement: equidistant (first row), bundled with minimal distance 1 (second row), bundled with minimal distance 0.5 (third row) and two bundled and one isolated combustion chamber (last row).

ature distribution within the turbine plenum. The locally strong velocity amplitudes in the bundled cases carry away the temperature peaks coming from the combustion chambers very fast also giving rise to hotter and colder spots than in case one and four (see Figure 2.10). As the pressure is not only highest but also lowest in the centre for the bundled cases, it is also cooler compared to the rest of the plenum perimeter although a combustion chamber directly fires into that region. Figure 2.9 suggests that while in all cases the mean temperature rises in the turbine plenum, the bundled cases might settle with a lower mean temperature in the end, which could be desirable to enhance the efficiency of turbine blade cooling. Case one and four show quite similar behaviour for the temperature so far.

2.3 Conclusions

The simulations of Section 2.2 have proven that the one-dimensional network model is a useful tool for the investigation of interaction behaviour in different gas turbine configurations. Its computational efficiency allows for in-depth parameter studies as well as the development and tuning of optimal control schemes. The simulations have also given hints about how the distribution of combustion chambers along the turbine plenum perimeter can change the flow property patterns therein so it will be worth a more intensive study. For now, we conclude that the bundled cases can generate higher velocities but also locally higher material stress than the equidistant one. Hence, for a gas turbine with about six combustion chambers, it might be beneficial to combine both options by equally spacing bundles of combustion chambers out as has been done in case four.

For future studies, a more detailed model of the turbomachinery is already on the anvil. It will couple compressor and turbine as should be in a real-life gas turbine and thereby enable the simulation of the starting process and optimisation of plenum parameters. It might also be interesting to have a look at firing patterns as has been done for the PDC in [62] instead of running the SEC-tubes simultaneously. A thorough investigation of misfiring resistance of the SEC as well as the restarting of a choked off combustion chamber as has been teased in [59] should also help to gain some more insight of the possible advantageous and challenges of a full SEC gas turbine.

Chapter 3

Moving Mesh Method

When hot spots in the SEC setting are to be examined, a high spacial resolution of at least 5×10^{-5} m is crucial since they can lead to detonations. Usually, such high resolutions would violate the assumption of cross-sectional averaging but hot spot studies have a slightly different mindset. In this case the legitimation of one-dimensional simulation follows from spherical symmetry, interpreting the spacial dimension as the radius. Such an investigation was to be conducted by a partner project within the CRC using the SEC-code (see [63]). Therefore the code needed to be extended by the possibility to handle fine resolved grids in a computationally efficient way. A classical ansatz for such a challenge is adaptive mesh refinement but in this chapter an alternative called "moving mesh method" (MMM) is shown. At first, a short introduction to adaptive remeshing is given in Section 3.1, afterwards the specific implementation into the SEC-code is described in detail in Section 3.2 and last a hot spot and an SEC simulation using the MMM are presented in Section 3.3, followed by the concluding remarks of Section 3.4.

3.1 Introduction to the MMM

As high spacial grid resolution is essential for accuracy of solutions but also one of the biggest consumers of computation time in most applications of numerical simulation, adaptive mesh refinement strategies have been around for decades now. The general idea is as simple as can be: equidistant grids can only have a high resolution everywhere but one often just needs specific areas to have such a costly resolution (e.g. due to edgy geometry of the domain or steep gradients of the solution). So only these critical regions should have many grid points while elsewhere the mesh can be coarser. As the optimal resolution might be unknown a priori or the critical regions change over time, one needs a procedure which automatically decides where the numerical grid has to be refined or coarsened. The best known approach for adaptive meshing methods is to identify critical and uncritical areas and add or subtract grid points. This is usually referred to as h-adaptivity and leads to a varying number of grid points over time. For problems where the solution develops waves of steep gradients travelling rapidly through the domain, like the SEC, this approach is rather impractical. Furthermore, computational costs emerging from chemical reactions scale with the number of grid cells. This is because simulating complex chemistry requires solving a mostly stiff system of ordinary differential equations with as many equations as there are designated reactions per grid cell. Consequentially, an alternative to h-adaptive mesh refinement is used, which shifts the grid cell interfaces along with the critical regions, maintaining their number. This class of methods is called r-adaptive or moving mesh and especially suited for our specific application. For an extensive overview of adaptive moving mesh methods see [58], [30], [56] and [32].

Most of the MMM are formulated for finite differences and finite elements but for the SEC-code a proposal from van Dam and Zegeling for a moving mesh finite volume algorithm for hyperbolic partial differential equation (PDE) systems published in [61] was adapted and will be outlined in the following. The original algorithm is based on the approach introduced by Tang and Tang in [55] combined with a monitor function, which is the driving force behind the remeshing process, developed by Beckett et al. in [5]. Roughly, the algorithm works like this: Compute monitor function values for each grid cell based on the current solution and move grid interfaces accordingly, then rematch the solution with the new grid via interpolation and, finally, advance the solution to the new time level using any numerical scheme suitable for the problem at hand.

As simple as this sounds, the brainpower is in the details as usual, beginning with the definition of the adaptive mesh. As a scope of reference $\Omega_c := [0, 1]$ with cell interfaces $\xi_{i-1/2}$, $i \in \{1, ..., N+1\}$ is introduced. This will be the computational domain which is subdivided equidistantly into N cells. To switch between Ω_c and the physical domain $\Omega_p := [x_L, x_R]$ with cell interfaces $x_{i-1/2}$, a transformation is used

$$x = x(\xi), \quad \xi \in \Omega_c \quad \text{or} \quad \xi = \xi(x), \quad x \in \Omega_p, \tag{3.1}$$

respectively. The continuous solution with s quantities is again denoted by $q(x,t) \in (\Omega_p \times \mathbb{R}_{\geq 0} \to \mathbb{R}^s)$. Over many numerical studies in the past years it has proven advantageous to impose an equidistribution condition on the mesh (see [58] for details)

$$\int\limits_{x_{i-1/2}}^{x_{i+1/2}} \omega(q(\zeta, t))\, d\zeta = \text{constant}, \quad \forall t \in \mathbb{R}_{\geq 0},\ i \in \{1, ..., N\}. \tag{3.2}$$

This

$$\omega : (\Omega_p \times \mathbb{R}_{\geq 0} \to \mathbb{R}^s) \to \mathbb{R}_{>0}$$

will be called monitor function. The application of (3.2) results in small grid cells where ω is large and big grid cells where ω is small. This makes sense if ω is some kind of error-related function. In the case at hand, it will be a function of solution gradients (see (3.7) below). In a discrete form, (3.2) reads

$$\Delta x_i \omega_i = \text{constant}, \quad \forall i \in \{1, ..., N\} \tag{3.3}$$

with $\Delta x_i := x_{i+1/2} - x_{i-1/2}$ and $\omega_i := \omega(q(x_i, t))$ as shorthands. In a more generalised notation one finds

$$\Delta x_i = \left.\frac{\partial x}{\partial \xi}\right|_i \Delta \xi = \left.\frac{\partial x}{\partial \xi}\right|_i N^{-1}, \quad \forall i \in \{1, ..., N\}. \tag{3.4}$$

The last equation uses the fact that the mesh on Ω_c is equidistant and thus $\Delta \xi = N^{-1}$. Inserting (3.4) into (3.3) yields an implicit definition of the mesh transformation function (3.1)

$$\omega_i \left.\frac{\partial x}{\partial \xi}\right|_i = \text{constant} \Leftrightarrow \frac{\partial}{\partial \xi}\left(\omega_i \left.\frac{\partial x}{\partial \xi}\right|_i\right) = 0, \quad \forall i \in \{1, ..., N\}. \tag{3.5}$$

From (3.5) a linear equation system in $(x_{3/2}, ..., x_{N-1/2})$ can be calculated by inserting the monitor values of the current solution and central differences for $\left.\frac{\partial x}{\partial \xi}\right|_i$. We now proceed to using the discrete Q_i^k, which is the integral solution average of the i-th grid cell at the current time level t_k. We will suppress the time step index k here and use the iteration instead, yielding $Q_i^{[\nu]}$. Hereafter, we imply $\nu \in \mathbb{N}$ and omit "$\forall \nu \in \mathbb{N}$" from equations. The ν-th step in the Gauss-Seidel (GS) iteration for the linear equation system of grid cell interfaces then looks as follows

$$x_{i-1/2}^{[\nu+1]} = \frac{\omega(Q_{i-1}^{[\nu]}) x_{i-3/2}^{[\nu+1]} + \omega(Q_i^{[\nu]}) x_{i+1/2}^{[\nu]}}{\omega(Q_{i-1}^{[\nu]}) + \omega(Q_i^{[\nu]})}, \quad \forall i \in \{2, ..., N\}. \tag{3.6}$$

It has been proven in [55] that the mesh interfaces maintain their order when computed with (3.6) which is crucial for any sensible mesh redistribution. To keep the additional computational costs low a rather generous tolerance value or small number of maximum iteration steps is recommended for the iterative GS solver. Finding the new mesh with high accuracy is not the main subject here. For every iteration of the GS solver the solution must be interpolated on the new mesh (see Subsection 3.2.1).

As the monitor function ω is the core of any MMM and crucial for its performance in terms of computational efficiency and accuracy much care must be taken when deciding for a distinct function. Van Dam and Zegeling adapted suggestions from [5] for systems of PDEs with s quantities q_j. Their approach is as follows

$$\omega(q) = \sum_{j=1}^{s} \left[(1 - \beta_M)\alpha_j(q) + \beta_M \left|\frac{\partial q_j}{\partial \xi}\right|^{1/2}\right] \tag{3.7}$$

with

$$\alpha_j(q) = \int\limits_{\Omega_c} \left| \frac{\partial q_j}{\partial \xi} \right|^{1/2} d\xi, \quad \forall j \in \{1, ..., s\}.$$

The value $\alpha_j(q)$ is a floor value which depends on the solution and prevents the mesh from collapsing at very steep gradients. $\beta_M \in (0, 1)$ is a constant model parameter and represents the ratio of points in critical regions.

Since very fast mesh movements imply an issue for the solution interpolation after this movement, it is recommended in [61] to apply at least one step of smoothening to the monitor function with a low-pass filter

$$\omega_i \leftarrow \frac{1}{4} \left(\omega_{i-1} + 2\omega_i + \omega_{i+1} \right), \quad \forall i \in \{2, ..., N-1\}. \tag{3.8}$$

After having determined a sufficiently good approximation to the new mesh and the solution's interpolation on it, the next step is to advance the solution forward in time. Van Dam and Zegeling use a MUSCL-type method with local Lax–Friedrichs flux and Runge-Kutta time-stepping scheme while any other method will also do because this step is completely independent from the remeshing.

3.2 Implementation

Van Dam and Zegeling set great value on universal applicability of their algorithm which makes it a good basis. To adapt it to the SEC-code and the challenges of its applications, some adjustments have been implemented. As a minor change, the direction from which the GS iterative solver begins, i.e., left or right side of domain, is selectable including the option to solve with both directions and use the averaged grid which is the most unbiased but also computationally expensive variant. The solution derivatives for the monitor function can now be evaluated on the non-equidistant physical domain. Moreover, usage of the MM feature is enabled not only within the context of spherical symmetry but also for our cross-sectional averaged SEC simulations through introduction of a lower bound for Δx_i^k in the calculation of the new mesh, stopping the GS iterations when a grid cell gets too small otherwise.

The most interesting question that has come up during the implementation of the MM feature pertained to the matter of interpolation. As such, the interpolation smears the solution, aggravating numerical diffusion - a property which is undesirable within the scope of the SEC-code. Thus a different approach was realised, that solves the PDEs directly jumping from one grid to the next. To the literature it is known as an interpolation-free MMM albeit realised different from the common approach: A mere correction of the numerical flux had to be implemented to achieve this. More detailed descriptions are given in Subsection 3.2.1. Furthermore, the quasi one-dimensionality was restored as explained in Subsection 3.2.2. The full workflow considering the discretised formulations is shown in Subsection 3.2.3.

3.2.1 Interpolated versus Simultaneous Solving

Clearly, the discrete solution changes, if the grid is renewed. Thus, in [61] an interpolation step is suggested after every step ν of the GS solver. First we define $\mathcal{V}(x_{i-1/2}^{[\nu+1]}) := x_{i-1/2}^{[\nu]} - x_{i-1/2}^{[\nu+1]}$, $\forall i \in \{1, ..., N+1\}$, the difference between old and new mesh which is assumed to be small. This has been quantified by [55] with $|\mathcal{V}(x) << 1|, \forall x \in \Omega_p$. Then, in resemblance to (1.4), we write a conservative interpolation scheme

$$Q_i^{[\nu+1]} = \frac{\left(x_{i+1/2}^{[\nu]} - x_{i-1/2}^{[\nu]}\right) Q_i^{[\nu]} - \left((\mathcal{V}Q)_{i+1/2}^{[\nu+1]} - (\mathcal{V}Q)_{i-1/2}^{[\nu+1]}\right)}{x_{i+1/2}^{[\nu+1]} - x_{i-1/2}^{[\nu+1]}}, \tag{3.9}$$

where we keep suppressing the time step index k for the grid cell values Q. $(\mathcal{V}Q)_{i-1/2}$ can be interpreted as a numerical flux and computed employing e.g. the Van Leer flux

$$(\mathcal{V}Q)_{i-1/2}^{[\nu+1]} = \frac{\mathcal{V}_{i-1/2}^{[\nu+1]}}{2}(Q_{i-1/2}^{+} + Q_{i-1/2}^{-}) - \frac{\left|\mathcal{V}_{i-1/2}^{[\nu+1]}\right|}{2}(Q_{i-1/2}^{+} - Q_{i-1/2}^{-}), \tag{3.10}$$

with $\mathcal{V}_{i-1/2}^{[\nu+1]} := \mathcal{V}(x_{i-1/2}^{[\nu+1]})$ and $Q_{i-1/2}^{+}$ and $Q_{i-1/2}^{-}$, $\forall i \in \{1, ..., N+1\}$, being estimations of the solution's value at $x_{i-1/2}^{[\nu]}$ from the right and from the left, respectively. Classical reconstruction methods like linear approximation by spacial derivatives can be used to determine these values.

As a first approach to the implementation of MM into the SEC-code, interpolation was realised in a similar manner but found to diffusive. Of course more educated numerical fluxes - just like the one used to solve the Euler equations - could have been used instead of (3.10) and a few still simple ones have been tested. Nonetheless, the result was unsatisfactory considering accuracy. Because MM is supposed to save computation time and very sophisticated numerical fluxes can produce too much computational overhead, it has proven prohibitive to stick to that course. Therefore, a different ansatz was chosen which is part of the current implementation of the SEC-code: simultaneous updating of mesh and flow properties. In the literature this is most often realised by including the mesh equation into the PDE system which is to be solved. According to [56] this approach helps with large gradients, reduces time variation and saves time due to skipping the interpolation steps. The drawbacks are increased stiffness of the differential equations and an additional variable which is hard to determine optimally. Since the MMM was already implemented and one of the driving ideas of the code's design is operator splitting, following [25] a solution was found which lies in-between these two strategies. The calculation of the new mesh is still carried out as a stand-alone function. Afterwards the governing equations are solved on a linearly moving mesh. The advantages of this method are mixed: there is no extra stiffness and variable, while it is still easier to resolve steep gradients without needing interpolations.

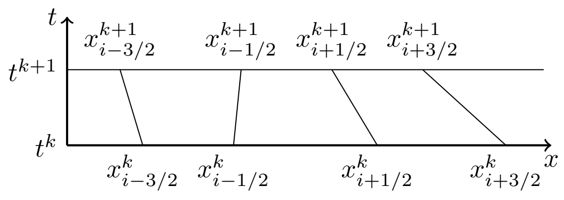

Figure 3.1: Diagram showing linearised movement of grid cell interfaces between two time levels t_k and t_{k+1}.

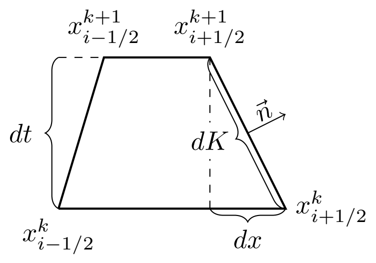

Figure 3.2: Close-up of one trapezoidal element from time-space diagram

To achieve this new goal we need to consider trapezoidal space-time elements rather than rectangular ones as can be seen in Figure 3.2. Denoting the normal vector of the domain boundary as $\vec{n}$ we rethink the conservation law (1.3) in integral form as

$$\frac{d}{dt}\int\limits_{\Omega_p} q(x,t)\; dx = -\int\limits_{\partial\Omega_p} f(q)\vec{n}ds.$$

We will understand these equations as componentwise for q and $f(q)$. With the trapezoidal elements the flux over one cell interface K can be written as

$$\int\limits_K (f(q), q)\cdot\vec{n}\; dK. \tag{3.11}$$

The need of a vector $(f(q), q)$ might not seem evident at a first glance. So just imagine that no stream would cause a flux of quantity q over the boundary over time but the boundary itself moves. This movement causes more or less q to be in that distinct cell. This implication is depicted by the t-entry of the normal vector and must thus be multiplied by q whereas the x-entry is multiplied by $f(q)$. This means a correction term must be found to include the mesh movement into the computation of the flux. First, we want to rewrite (3.11) to find the flux of the Euler equations on a simultaneously moving mesh. For this purpose, we use the following geometrical arguments as can be extracted from Figure 3.2

$$\begin{aligned}
\mathcal{V} =& \frac{dx}{dt}, \\
\vec{n} =& \frac{1}{\|\vec{n}\|}\begin{pmatrix} 1 \\ \\ -\mathcal{V} \end{pmatrix} = \frac{1}{\sqrt{1+\mathcal{V}^2}}\begin{pmatrix} 1 \\ \\ -\mathcal{V} \end{pmatrix}, \\
dK =& \sqrt{dt^2 + dx^2} = \sqrt{dt^2\left(1 + \frac{dx^2}{dt^2}\right)} = \sqrt{(1+\mathcal{V}^2)}\ dt.
\end{aligned}$$

With these identities the integral (3.11) reads

$$\int\limits_{t_k}^{t_{k+1}} (f(q), q) \cdot \frac{1}{\sqrt{1+\mathcal{V}^2}} \begin{pmatrix} 1 \\ \\ -\mathcal{V} \end{pmatrix} \sqrt{1+\mathcal{V}^2}\ dt = \int\limits_{t_k}^{t_{k+1}} f(q) - \mathcal{V}q\ dt.$$

So, $f(q) - \mathcal{V}q$ is our moving mesh flux. It is desirable to reuse the original numerical flux function to exploit its advantages. To achieve this, we introduce the relative flow velocity $u - \mathcal{V}$ to the static-grid flux $f(q)$ as

$$f^{rel}(q) := \begin{pmatrix} \varrho(u-\mathcal{V}) \\ \\ \varrho(u-\mathcal{V})^2 + p \\ \left(\varrho e + \varrho\frac{(u-\mathcal{V})^2}{2} + p\right)(u-\mathcal{V}) \\ \\ \varrho Y(u-\mathcal{V}) \end{pmatrix}.$$

$f(q) - \mathcal{V}q$ can be recast to equal $f^{rel}(q)$ plus some correction term which will be derived in the following equations. For this purpose, we rearrange the moving mesh flux

$$f(q) - \mathcal{V}q = \begin{pmatrix} \varrho u - \mathcal{V}\varrho \\ \\ \varrho u^2 + p - \mathcal{V}\varrho u \\ \\ (\varrho E + p)u - \mathcal{V}\varrho E \\ \\ \varrho Y u - \mathcal{V}\varrho Y \end{pmatrix}$$

$$= \begin{pmatrix} \varrho(u-\mathcal{V}) \\ \varrho(u-\mathcal{V})u + p - \varrho(u-\mathcal{V})\mathcal{V} + \varrho(u-\mathcal{V})\mathcal{V} \\ \varrho E(u-\mathcal{V}) + pu - p\mathcal{V} + p\mathcal{V} \\ \varrho Y(u-\mathcal{V}) \end{pmatrix}$$

$$= \begin{pmatrix} \varrho(u-\mathcal{V}) \\ \varrho(u-\mathcal{V})^2 + p + \varrho(u-\mathcal{V})\mathcal{V} \\ (\varrho E + p)(u-\mathcal{V}) + p\mathcal{V} \\ \varrho Y(u-\mathcal{V}) \end{pmatrix}.$$

So only the third component needs some more effort. We write

$$E = e + \frac{u^2}{2} \rightarrow E^{rel} := e + \frac{(u-\mathcal{V})^2}{2}.$$

Using these equations the third component of $f(q) - \mathcal{V}q$ can be recast

$$\begin{aligned}
(f(q) - \mathcal{V}q)_3 =& \varrho\left(e + \frac{u^2}{2}\right)(u-\mathcal{V}) + p(u-\mathcal{V}) + p\mathcal{V} \\
=& \varrho\left(e + \frac{1}{2}(u^2 - 2u\mathcal{V} + \mathcal{V}^2) + \frac{1}{2}(2u\mathcal{V} - \mathcal{V}^2)\right)(u-\mathcal{V}) \\
& + p(u-\mathcal{V}) + p\mathcal{V} \\
=& \left(\varrho E^{rel} + p\right)(u-\mathcal{V}) + \varrho(u-\mathcal{V})\left(u - \frac{\mathcal{V}}{2}\right)\mathcal{V} + p\mathcal{V} \\
=& f^{rel}(q)_3 + \varrho\left((u-\mathcal{V})^2 + \frac{1}{2}(u\mathcal{V} - \mathcal{V}^2)\right)\mathcal{V} + p\mathcal{V} \\
=& f^{rel}(q)_3 + \mathcal{V}\left(\varrho(u-\mathcal{V})^2 + p\right) + \frac{\mathcal{V}^2}{2}\varrho(u-\mathcal{V}) \\
=& f^{rel}(q)_3 + \mathcal{V}f^{rel}(q)_2 + \frac{\mathcal{V}^2}{2}f^{rel}(q)_1.
\end{aligned}$$

Now we can write the moving mesh flux $f(q) - \mathcal{V}q$ in terms of the relative velocity flux $f^{rel}(q)$

$$f(q) - \mathcal{V}q = f^{rel}(q) + f^{rel}(q)_1 \begin{pmatrix} 0 \\ \mathcal{V} \\ \dfrac{\mathcal{V}^2}{2} \\ 0 \end{pmatrix} + f^{rel}(q)_2 \begin{pmatrix} 0 \\ 0 \\ \mathcal{V} \\ 0 \end{pmatrix}. \tag{3.12}$$

This flux will be applied to reconstructed values of the discrete solution Q. They are approximated with the same MUSCL-Hancock scheme as in the basic SEC-code, only taking the non-equidistant mesh and its movement during the time step into account. Figure 3.3 shows a space-time trapezoid element in close up. Approximating the mesh movement linearly, we get

$$Q_{i-1/2}^{k+1/2} = Q_i^k + \left(\frac{x_{i-1/2}^k + x_{i-1/2}^{k+1}}{2} - x_i^k \right) \frac{\partial Q}{\partial x} + \frac{\Delta t^k}{2} \frac{\partial Q}{\partial t}, \quad \forall i \in \{1, ..., N+1\}. \tag{3.13}$$

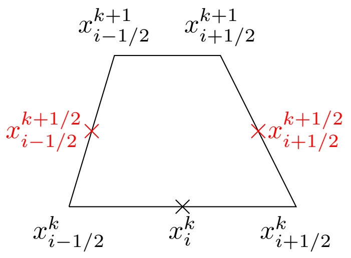

Figure 3.3: Points of reconstruction for half-time grid cell interface values of solution within a moving mesh simulation.

To calculate the relative velocity flux $f^{rel}(q)$ we need the solution q to be transformed to the relative system in a fashion similar to the flux correction

$$q^{rel} := \begin{pmatrix} \varrho \\ \varrho(u - \mathcal{V}) \\ \varrho e + \varrho \dfrac{(u-\mathcal{V})^2}{2} \\ \varrho Y \end{pmatrix} = \begin{pmatrix} \varrho \\ \varrho u - \varrho \mathcal{V} \\ \varrho e + \varrho \dfrac{u^2}{2} - \varrho \dfrac{u^2}{2} + \varrho \dfrac{(u-\mathcal{V})^2}{2} \\ \varrho Y \end{pmatrix} \tag{3.14}$$

$$=q+\begin{pmatrix} 0 \\ -\varrho\mathcal{V} \\ \frac{\varrho}{2}\left((u-\mathcal{V})^2-u^2\right) \\ 0 \end{pmatrix}. \tag{3.15}$$

Now everything we need to advance the solution on a simultaneously moving mesh is gathered. Subsection 3.2.3 will walk the reader through the workflow in detail. In the preliminary work, it has been seen that this approach is a better fit to the requirements of the SEC-code, which is why the method was kept as the only possible treatment.

3.2.2 Quasi One-Dimensionality and Its Well-Balancing

One of the interesting features of the SEC-code originating from [7] is the support of axial variation of the cross-sectional area A of the simulated domain. Surely, it is desirable to keep this feature within the MM scope. Only slight changes had to be made to incorporate the MM idea in the existing code. The major difference is the fact that this area is now necessarily given by a function since the cell interfaces can and should move a lot during one simulation, thus discrete values would yield a very bad approximation. Hence, A must be kept up to date for every new mesh.

When implementing the MM feature for the partner project another hurdle appeared in its proximity. That is why it will be discussed here although it is not directly related to the MM topic. To understand the challenge, we must know how the original feature for quasi one-dimensional problems was realised. It was implemented following [52] yielding a conservative scheme. Solving the quasi one-dimensional Euler equations

$$\frac{\partial}{\partial t}\begin{pmatrix} \varrho A \\ \varrho A u \\ \varrho A E \\ \varrho A Y \end{pmatrix}+\frac{\partial}{\partial x}\begin{pmatrix} \varrho u A \\ \varrho u^2 A + pA \\ uA(\varrho E + p) \\ \varrho u A Y \end{pmatrix}=p\frac{\partial}{\partial x}\begin{pmatrix} 0 \\ A \\ 0 \\ 0 \end{pmatrix} \tag{3.16}$$

simply involved multiplying all conservative states with A, updating the momentum with a second order term

$$\varrho u \leftarrow \varrho u + \frac{p}{A}\frac{\partial A}{\partial x}\frac{\Delta t^k}{2}+\frac{1}{A}\left(\frac{\partial p}{\partial t}\frac{\partial A}{\partial x}\frac{(\Delta t^k)^2}{8}\right) \tag{3.17}$$

and solving the Euler equations (1.1) like before.

As convenient as this method is, there is one major drawback: Due to the splitted solution of momentum source term (3.17) and Euler equations, it is not well-balanced. This means, that steady-state solutions with zero flow velocity cannot be maintained. Since sometimes simulations with flowless initial condition are needed, e.g. because the chemistry is to be observed, a different approach was developed in the scope of this thesis. Starting with (3.16) and rearranging with the help of the chain rule of differentiation as well as the obvious fact that $\frac{\partial A}{\partial t} = 0$ we arrive at the non-conservative formulation

$$\frac{\partial}{\partial t}\begin{pmatrix} \varrho \\ \varrho u \\ \varrho E \\ \varrho Y \end{pmatrix} + \frac{\partial}{\partial x}\begin{pmatrix} \varrho u \\ \varrho u^2 + p \\ u(\varrho E + p) \\ \varrho u Y \end{pmatrix} = -\frac{u}{A}\frac{\partial A}{\partial x}\begin{pmatrix} \varrho \\ \varrho u \\ \varrho E + p \\ \varrho Y \end{pmatrix}. \tag{3.18}$$

To solve (3.18) we can again follow the operator splitting idea. To find an easier expression for the energy equation, we execute the following steps which result in an equation for pressure. Unfortunately, this involves the restricting assumption that the specific heat-capacity at constant volume c_V is a constant. Nevertheless, this approximation is most often justified for there is no chemistry changing c_V rapidly in this step. For the sake of readability we will use a common shorthand for partial derivatives, namely $(.)_t$ and $(.)_x$ for the derivation with respect to time or space, respectively. The set of equations to begin with then looks as follows

$$\varrho_t = = -\varrho u \frac{A_x}{A} \qquad \text{by (3.18)}, \tag{3.19}$$

$$(\varrho E)_t = -\varrho u \frac{A_x}{A}\left(E + \frac{p}{\varrho}\right) \qquad \text{by (3.18)}, \tag{3.20}$$

$$E = e + \frac{u^2}{2} \qquad \text{by (1.2)}, \tag{3.21}$$

$$u_t = 0 \qquad \text{by (3.18)}, \tag{3.22}$$

$$e = c_v T = \frac{p}{(\gamma - 1)\varrho} \qquad \text{if } c_V \text{ is a constant.} \tag{3.23}$$

Now inserting (3.21) into the energy equation (3.20) yields

$$(\varrho e)_t + \left(\varrho \frac{u^2}{2}\right)_t = -\varrho u \frac{A_x}{A}\left(e + \frac{u^2}{2} + \frac{p}{\varrho}\right).$$

Substituting the factor in front of the bracket on the right-hand side via (3.19) and employing the chain rule yields

$$(\varrho e)_t + \varrho_t \frac{u^2}{2} + \varrho \left(\frac{u^2}{2} \right)_t = \varrho_t e + \varrho_t \frac{u^2}{2} + \varrho_t \frac{p}{\varrho}.$$

With the help of (3.22) all terms involving u cancel out and we replace e using (3.23)

$$\frac{p_t}{\gamma - 1} = \frac{\varrho_t}{\varrho} \left(\frac{p}{\gamma - 1} + p \right) = \frac{1}{\gamma - 1} \frac{\varrho_t}{\varrho} \gamma p.$$

Multiplying by $\gamma - 1$ finally gets us

$$p_t = \frac{\varrho_t}{\varrho} \gamma p. \tag{3.24}$$

Replacing the energy equation from (3.18) by (3.24) the system can be solved analytically through

$$\begin{aligned}
\varrho(t) =& \varrho(t_0) \exp \left(-u \frac{A_x}{A} (t - t_0) \right), \\
u(t) =& u(t_0), \\
p(t) =& p(t_0) \exp \left(-u \frac{A_x}{A} (t - t_0) \right), \\
Y(t) =& Y(t_0).
\end{aligned}$$

Naturally, t_0 is chosen to be the old time level, such that $t - t_0 = \frac{\Delta t^k}{2}$. Please keep in mind that due to the Strang splitting two half time steps are calculated. It is easy to see, that for $u = 0$ everywhere, the solution will not experience unphysical changes due to variation of A any longer. For the Euler equations the original solver can be used since only the source term differs from (1.1).

As the implementation of the well-balanced method described above has proven to be a huge improvement in some applications, a scheme was derived which combines conservation with well-balancing and removes the restriction of c_V being a constant. In Chapter 1 we defined the discrete solution Q as a cell-averaged integral value. Now, for the quasi one-dimensional formulation, we need the weighting with $A(x)$, redefining

$$Q_i^k := \frac{1}{A_i^k \Delta x_i^k} \int\limits_{x_{i-1/2}}^{x_{i+1/2}} q(\zeta, t) A(\zeta) \, d\zeta. \tag{3.25}$$

Employing this notion, it turns out one only needs a smart discretisation of (3.16). Using again $F_{i-1/2}^{k+1/2}$ and $A_{i-1/2}^k$ as the numerical flux and cross-sectional area value at $x_{i-1/2}$, respectively, we can write

$$\left(Q_i^{k+1}\right)_{1,3,4} = \left(Q_i^k\right)_{1,3,4} - \frac{\Delta t^k}{\Delta x_i^k}\left(\frac{A_{i+1/2}^k F_{i+1/2}^{k+1/2} - A_{i-1/2}^k F_{i-1/2}^{k+1/2}}{\frac{1}{2}\left(A_{i+1/2}^k + A_{i-1/2}^k\right)}\right)_{1,3,4} \tag{3.26}$$

for ϱ,ϱE and ϱY. The momentum ϱu needs the additional pressure source term

$$\left(Q_i^{k+1}\right)_2 = \left(Q_i^k\right)_2 - \frac{\Delta t^k}{\Delta x_i^k}\left(\frac{A_{i+1/2}^k\left(F_{i+1/2}^{k+1/2} - p_i^*\right) - A_{i-1/2}^k\left(F_{i-1/2}^{k+1/2} - p_i^*\right)}{\frac{1}{2}\left(A_{i+1/2}^k + A_{i-1/2}^k\right)}\right)_2 . \tag{3.27}$$

Since (3.16) and (1.4) are basis for this scheme it clearly is conservative. To make sure it is also well-balanced, the pressure p_i^* must be chosen correctly. For a steady state solution, the only remaining contribution to the flux f is p in the momentum component. Considering $p_x = 0$, p_i^* must be chosen as the average of the pressures $p_{i+1/2}^*$ and $p_{i-1/2}^*$ that the numerical flux function calculates to achieve a well-balanced formulation. This is the way quasi one-dimensionality is carried out in the current SEC-code.

3.2.3 Discretised Workflow

The main parts of the MM algorithm have been described above. In this subsection a complete overview of the workflow of the SEC-code in MM-mode as depicted in Figure 3.4 is given supplying further insight to the discrete structures.

After the simulation domain is set up with fitting MM configurations the algorithm starts the time stepping. It begins with the calculation of the new mesh. To be able to follow specific features of the solution and e.g. let the mesh react to temperature and pressure only, the user can simply select the quantities which influence the monitor function. For this purpose, Q is reduced to $\mathfrak{Q}$ which contains only the chosen quantities.

In order to evaluate the monitor function $\mathfrak{Q}$ is numerically differentiated on either the computational (uniform) grid or the physical (non-equidistant) grid. Although the latter makes the algorithm less stable, it is sometimes beneficial for resolving really sharp gradients which can occur at detonation fronts. For the differentiation on a non-equidistant grid a new method had to be implemented. According to common practice, let the approximation scheme be

$$\frac{\partial^j h}{\partial x^j}(\zeta) \approx \sum_{i=n_L}^{n_R} w_i^j h(x_i)$$

for any function h, order of derivation j, weights w_i^j, grid points x_i and evaluation point ζ. The stencil width $n_R - n_L$ depends on the order of accuracy chosen. The weights are calculated by the most commonly used Fornberg algorithm (see [27]).

```
set up domain with MM configuration (see Appendix B.2 for instructions);
while $t < t_{end}$ do
    begin mesh movement:
        begin calculate monitor values:
            extract chosen guiding quantities from state vectors;
            calculate numerical derivatives on uniform computational or
              non-equidistant physical domain;
            if dimensional then
                divide by reference value;
            end
            evaluate monitor function;
            for number of smoothing steps do
                apply low-pass filter (3.8);
            end
        end
        for GS iterations do
            compute new mesh according to (3.6);
        end
        if CFL condition (3.29) not met then
            rewind mesh as far as necessary;
        end
    end
    calculate $\Delta t^k$ using the new mesh;
    if switched on then
        apply chemistry and/or molecular transport;
    end
    begin solve Euler equations with MM:
        reconstruct solution on grid cell interface midtime (3.13);
        transform reconstructed values of solution to relative scope (3.14);
        calculate numerical flux as usual;
        apply flux correction (3.12);
        update mesh;
        if $A$ varies then
            update $A$;
        end
        apply flux in state update for changing mesh (3.30);
    end
    if switched on then
        apply chemistry and/or molecular transport;
    end
    $t \leftarrow t + \Delta t^k$;
end
```

Figure 3.4: Workflow of MM set-up

The basic idea is to differentiate the Lagrange interpolation polynomial of h and compute the weights w_i^j recursively from lower orders. It is formulated generally for any grid, evaluation point, order of derivation and accuracy. The implementation in the SEC-code follows [33] and [26] making extensive usage of Matlab's matrix notation for speed-up. Since it is not necessary to find $\frac{\partial \mathfrak{Q}}{\partial x}$ with high accuracy, the stencil size is chosen to be at the lower bound to save computation time.

If the solution is given in dimensional form, the derivatives are divided by a time-dependent reference value

$$\mathfrak{Q}_{ref}^k := \left| \max_{i \in \{1,\ldots,N\}} (\mathfrak{Q}_i^k) - \min_{i \in \{1,\ldots,N\}} (\mathfrak{Q}_i^k) \right| . \tag{3.28}$$

This procedure compensates for the possibly great differences in value scales between the flow quantities. If (3.28) yields 0 then the value is set to 1.

The discretisation of the floor value vector α slightly depends on the chosen domain of differentiation

$$\alpha = \begin{cases} \mathcal{L}^{-1} \displaystyle\sum_{i=1}^{N} \Delta x_i^k \left(\left| \frac{\partial \mathfrak{Q}}{\partial x} \right|_i \right)^{1/2} & \text{for physical domain,} \\ N^{-1} \displaystyle\sum_{i=1}^{N} \left(\left| \frac{\partial \mathfrak{Q}}{\partial \xi} \right|_i \right)^{1/2} & \text{for computational domain.} \end{cases}$$

with $\mathcal{L}$ being the physical domain's length. For the computational domain $|\Omega_c| = 1$ by definition and $\Delta\xi = N^{-1}$. Afterwards the monitor function values are computed according to (3.7) and the smoothening (3.8) is applied.

With the monitor function values the new mesh is calculated as suggested in (3.6). A user defined maximum number of GS iterations will be executed to find the new mesh. A threshold for the minimal mesh movement is not implemented because even van Dam and Zegeling say, a sensible tolerance threshold will unlikely be reached. The monitor function is only evaluated once before the GS algorithm since it was found more suitable to save the computation time than to more accurately weight the grid points. Moreover, we would need the interpolated solution which we substituted by the simultaneous evolution.

If the mesh is to shift too fast, there will be a different control mechanism anyway: a CFL-like condition. The time step is chosen, such that the solution waves do not intersect. Hence, to ensure that the numerical domain of dependence still includes the true domain of dependence, a grid point must not move more than half a grid cell to either side

$$-\frac{\Delta x_{i-1}^k}{2} \leq \mathcal{V}_{i-1/2}^{k+1} \leq \frac{\Delta x_i^k}{2}, \quad \forall i \in \{2, \ldots, N\}. \tag{3.29}$$

If this condition is not met, the mesh is calculated such that it maximally moves linearly towards the last calculated position without violating (3.29).

With this new mesh at hand but the old one still applied, the maximum time step Δt^k is calculated using the new grid and, following the Strang splitting idea, chemistry and molecular transport are computed if switched on. Next, the one-dimensional Euler equations must be solved. For that, the interface values of the solution are reconstructed via (3.13) and transformed into the relative system by (3.14) to compute the numerical flux F as usual before correcting it using (3.12). Now the new mesh and cross-sectional area, if applicable, are set to be the current ones. The last step is to incorporate the flux estimation $\mathcal{F}$ into the time stepping scheme to advance the solution to the next time level t_{k+1}. Since the SEC-code works with a FVM it is crucial to account for the possible change of cell width. This is done similar to the interpolation step (3.9)

$$Q_i^{k+1} = \frac{\Delta x_i^k \, Q_i^k - \Delta t^k \left(\mathcal{F}_{i+1/2}^{k+1/2} - \mathcal{F}_{i-1/2}^{k+1/2} \right)}{\Delta x_i^{k+1}}. \tag{3.30}$$

For a simulation with cross-sectional area variation we simply need to take one more aspect into account: the recalculation of the area A^k to A^{k+1}. Denoting the cell area averaged over the edges as $A_i^k := \frac{1}{2}\left(A_{i-1/2}^k + A_{i+1/2}^k\right)$ and the midtime area as $A_{i-1/2}^{k+1/2} := \frac{1}{2}\left(A_{i-1/2}^k + A_{i-1/2}^{k+1}\right)$, this yields the following discretised update step for all components but momentum

$$\left(Q_i^{k+1}\right)_{1,3,4} = \left(\frac{\Delta x_i^k A_i^k Q_i^k - \Delta t^k \left(A_{i+1/2}^{k+1/2} \cdot \mathcal{F}_{i+1/2}^{k+1/2} - A_{i-1/2}^{k+1/2} \cdot \mathcal{F}_{i-1/2}^{k+1/2} \right)}{\Delta x_i^{k+1} A_i^{k+1}} \right)_{1,3,4}.$$

The momentum component needs the additional pressure source term

$$\left(Q_i^{k+1}\right)_2 = \left(\frac{\Delta x_i^k A_i^k Q_i^k - \Delta t^k \left(A_{i+1/2}^{k+1/2} \cdot \left(\mathcal{F}_{i+1/2}^{k+1/2} - p_i^* \right) - A_{i-1/2}^{k+1/2} \cdot \left(\mathcal{F}_{i-1/2}^{k+1/2} - p_i^* \right) \right)}{\Delta x_i^{k+1} A_i^{k+1}} \right)_2.$$

Finally, the time update is completed applying half a time step for chemistry and molecular transport if activated.

3.3 Simulations

To demonstrate the moving mesh feature, two examples will be described and shown in the following section. Recommendations for default settings and the overall usage of MM within the SEC-code are given in Appendix B.2. All simulations with MM are started with a preprocessed grid using the included automatic grid initialisation function with 1000 iterations and 10 smoothing steps, while static grid simulations use an equidistant mesh.

governing equations	one-dimensional Euler (with MM)
chemistry	3-species ignition delay kinetics
domain	length: 0.5, no cross-sectional area variation
initial values	$T(x) = 1 - 0.2(x - 0.01)/0.01$ for $0 < x < 0.01$ and 1 else, $p = 1$, $u = 0$, fuel species for $0 < x < 0.2$ and product species else
boundaries	left: reflecting wall, right: continuous
MM setting	$\beta_M = 0.9$, monitor function: on computational domain, GS: from left with 5 iterations, smoothing steps: 1, followed quantities: ϱ, ϱu, ϱE
grid cells	200, 1000 and 3000 (static); 200 (MM)
time steps	step size chosen automatically, snapshots stored at multiples of 10^{-3}

Table 3.1: Settings for hot spot simulations.

3.3.1 Hot Spot

As the impulsion to implement a method for adaptive meshing came from the objective to investigate hot spots, this is the first example which will be given. In this simulation, the domain of length 0.5 is set up such that it triggers a premature ignition. On that account, we chose $p = 1$ and $u = 0$ everywhere, the leftmost two fifth of the domain are filled with fuel whereas the rest contains product species. The simulation name stems from the temperature profile which features a strong peak of maximum 1.2 at $x = 0$ rapidly degrading to 1 at $x = 0.01$. Since this is a simulation with spherical symmetry the left boundary must be a reflecting wall modelling the centre of the sphere while the right boundary is continuous.

In Figure 3.5 the temperature and pressure within the simulation domain are shown over space and time. We can see immediately that the static grid solutions differ a lot from each other as the resolution is refined and it is also obvious that the 200-grid-cells-solution with moving mesh matches the highly resolved static grid solution with 3000 grid cells quite well. It is notable that not only the temperature curve is properly met but the pressure wave patterns also look alike. This comes at the cost of the moved mesh temperature being a little blurry at the right edge, but

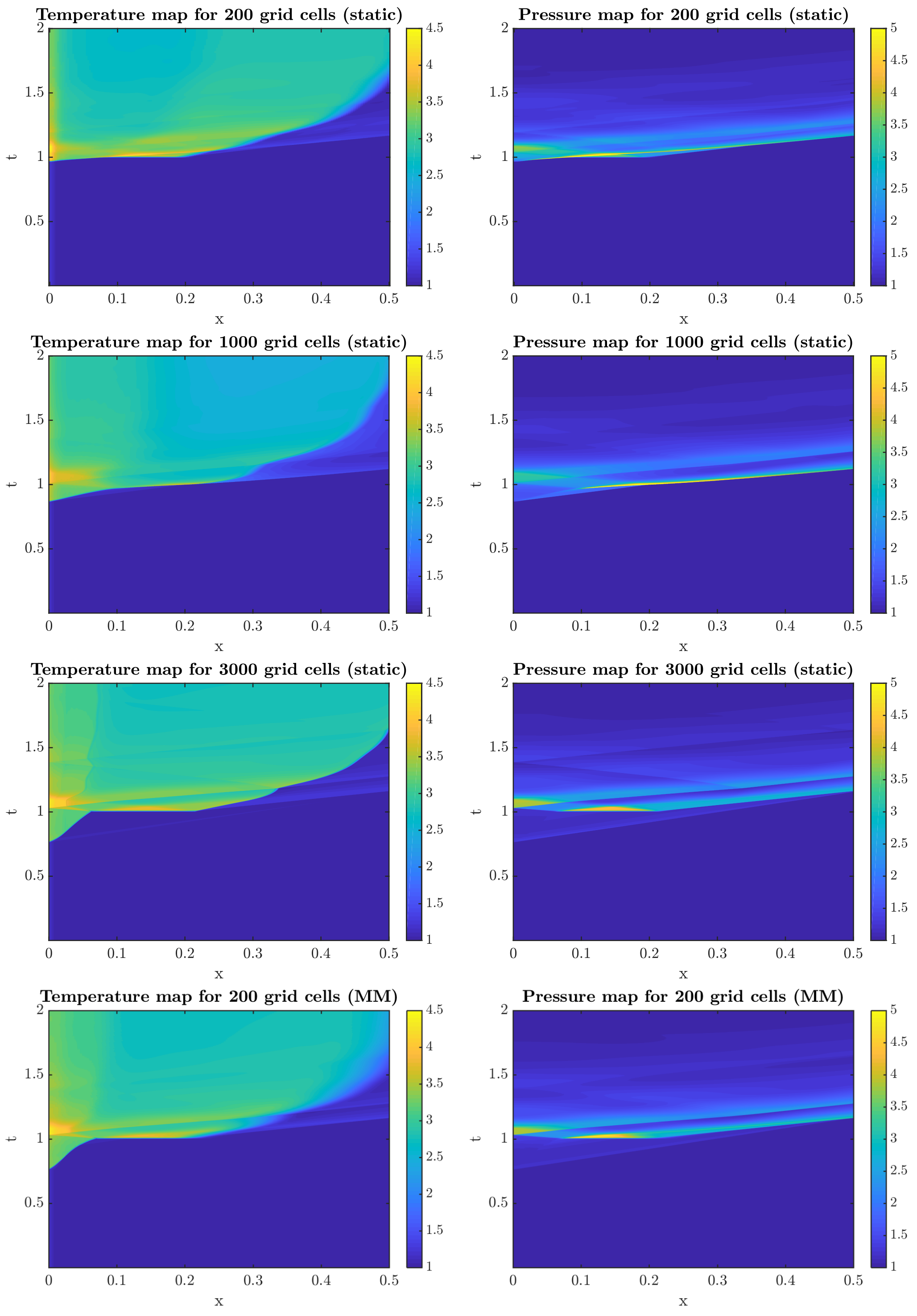

Figure 3.5: Temperature (left) and pressure (right) of a hot spot simulation over space and time with static grid with 200, 1000 and 3000 grid cells and with MM with 200 grid cells (top to bottom).

that is the behaviour one expects from this method as the pressure gradients are steeper.

It is save to say that MM is a very helpful tool when exploring new settings where the correct resolution might be unknown, because it is able to catch important features of the solution nonetheless. In terms of computation time there is also a big profit as the moving mesh simulation only needed about 30 seconds to initialise the grid and 244 seconds to compute the solution whereas the highly resolved static grid simulation took about 641 seconds on the same machine. This comparison of course assumes that one knows the number of grid cells needed otherwise some more simulations with an under-resolved grid would add to the static grid time budget. Such a comparison would be even more impressive if a complex chemistry mechanism was used since the number of cells has a much bigger impact on computation time in that case.

3.3.2 SEC

governing equations	quasi one-dimensional Euler (with MM)
chemistry	3-species ignition delay kinetics
domain	length: 1, area variation: Laval nozzle
initial values	$p = 1$, T: 1 for $x < 0.5$ and 2.5 else, $u = 0$, radical species for $x < 0.4$ and product species else
boundaries	left: fuelling pressure valve, right: isentropic expansion to $p = 1$
MM setting	$\beta_M = 0.9$, monitor function: on computational domain, GS: average from both sides with 5 iterations, smoothing steps: 1, followed quantities: ϱ, ϱu, ϱE, $\varrho Y_{radical}$
grid cells	100, 300 (static); 100, $\min(\Delta x) \geq 10^{-3}$ (MM)
time steps	step size chosen automatically, snapshots stored at multiples of 10^{-3}

Table 3.2: Settings for moving mesh SEC simulations.

Since the MM was equipped with a lower bound for Δx_i^k we are able to apply the adaptive remeshing to an SEC-setting. Hence, this simulation is set up to

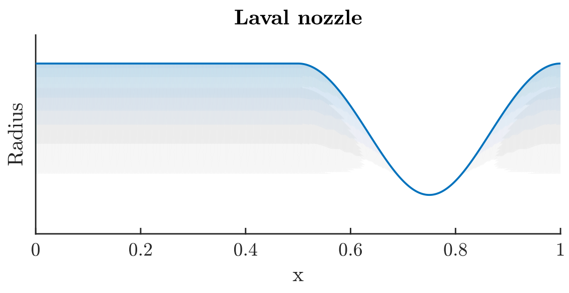

Figure 3.6: Radius of combustion chamber with Laval nozzle configuration.

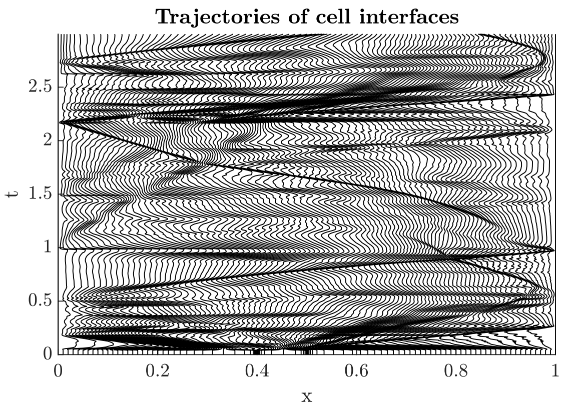

Figure 3.7: Trajectories of the grid cell interfaces for an SEC simulation with MM.

model a working SEC. The simulation domain of length 1 features a Laval nozzle as depicted in Figure 3.6 demonstrating quasi one-dimensionality in the MM frame. The initial values are chosen, such that a parcel of radical species, filling $x = [0, 0.4]$ at $T = 1$ is followed by an "air" buffer with product species at the same temperature separating the reactive flow from the hot exhaust gas modelled by product species at $T = 2.5$ in the downstream half of the combustion chamber. Pressure and velocity are constantly $p = 1$ and $u = 0$ everywhere at first. The right boundary represents an infinitely large plenum at constant pressure $p = 1$ while the left boundary is configured as a pressure valve, opening when the pressure in the leftmost grid cell of the combustion chamber drops below $p = 1$, injecting an "air" buffer first, then a stratified fuel profile which is tuned to auto-ignite homogeneously when generally undisturbed.

Figure 3.8 shows the comparison of temperature and pressure in the combustion chamber over space and time of two static grid and the MM simulation of this SEC example. The adaptive mesh is able to produce results as good as with thrice the grid cells in a static grid simulation. The trajectories of the grid cell interfaces in Figure 3.7 show how well the highly resolved regions follow the pressure wave emerging from the ignition area and still manage to catch lesser gradients as temperature and radical mass fraction.

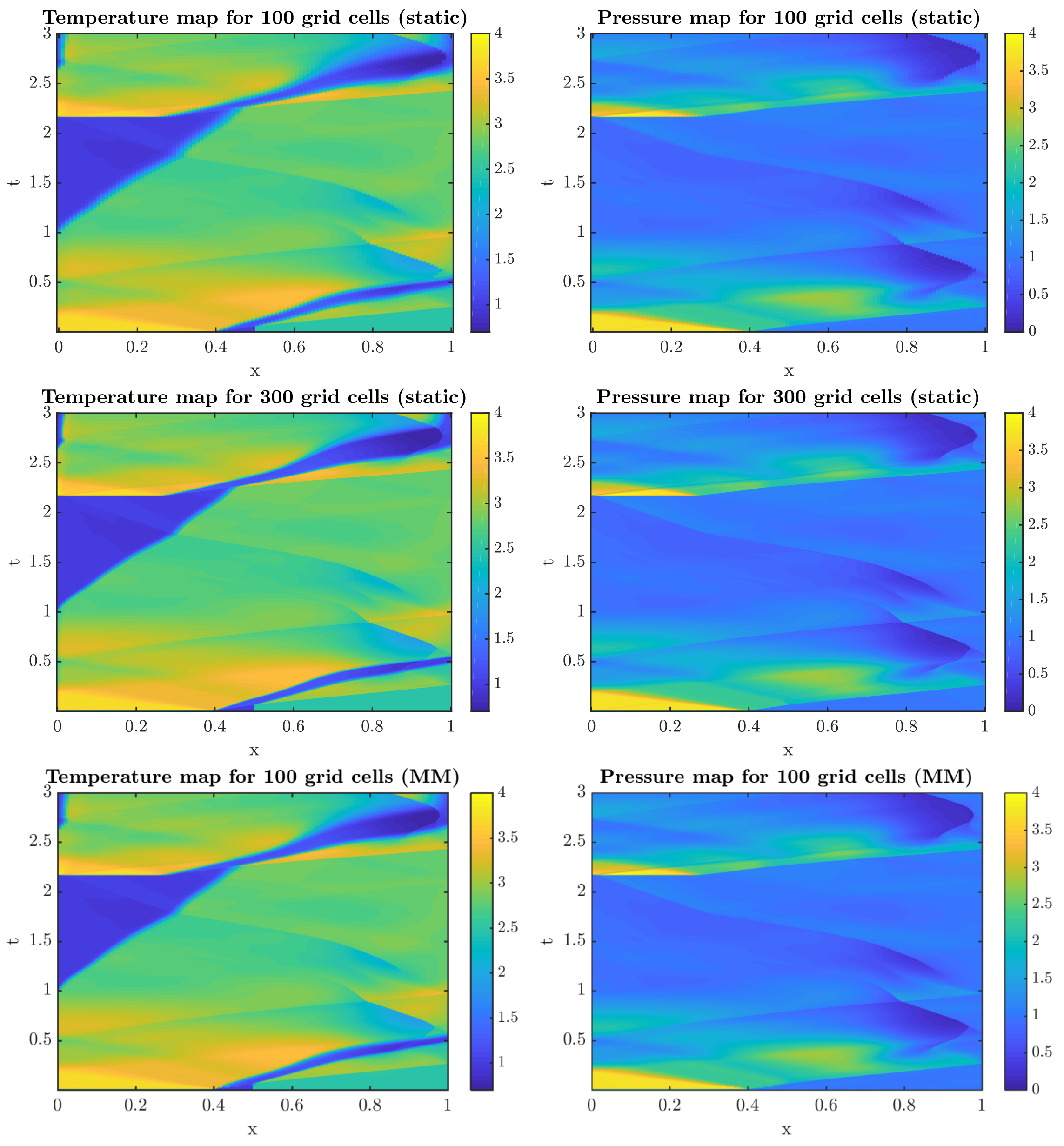

Figure 3.8: Temperature (left) and pressure (right) of an SEC simulation over space and time with static grid with 200 and 300 grid cells and MM with 100 grid cells (top to bottom).

3.4 Conclusions

The simulations from Section 3.3 have made it obvious that an adaptive mesh is highly beneficial for the SEC-code. The MMM fulfils the requirements of being computationally lightweight, accurate and easy to handle even without having any a priori knowledge of the solution which will be computed. It does a very good job resolving the relevant patterns with extremely few grid points and thus saves computation time even with the simple 3-species ignition delay kinetics. What also comes with the small number of grid cells is the saving of memory which can be an interesting factor for simulations with a lot of stored time steps, multiple domains or when using the code on a laptop. Actually, there is only one major drawback to this method which is rather little compared to its advantages: it is not possible to control the resolution directly. If the users know what resolution is needed for their application, they can only guess the correct combination of β_M and the number of grid cells that yields the desired resolution locally. This behaviour also leads to not being able to guess the computation time in advance, if the lower bound for Δx_i^k is not activated, as time steps sizes depend on the minimum cell width. The most important improvement to this tool would be its applicability to network simulations in the sense of Chapter 2, saving computation time and memory on more than one domain simultaneously.

Chapter 4

One-Dimensional Turbulence

In phase one of the CRC 1029, the simulations were restricted to an SEC which was already working to examine the overall concept. In this context - a full-fledged cyclic auto-igniting combustion process - molecular transport was justifiably neglected as gasdynamics are much faster. Turbulence was omitted to establish the SEC-code in the first place. In phase two, however, it was planned to investigate the starting process. As described in Section 1.2 it takes a diffusion flame to raise the pressure so the SEC can be started. To simulate this process, molecular transport and turbulence must be taken into account.

Fortunately, molecular transport simply implies adding another source term to (1.1) which can easily be coped with in the current code (see Subsection 4.3.2). Turbulence on the other hand is a much more troubling task. There are a lot of approaches with different advantages and drawbacks like direct numerical simulation (DNS), Reynolds-averaged Navier-Stokes (RANS) and large eddy simulation (LES) along with the linear eddy model (LEM) and one-dimensional turbulence (ODT). It was necessary to find a method which works well with one-dimensionality and the current code structure. Also we wanted a turbulence model which would not only give us mean quantities but lets us see the possible variance of outcomes to study the robustness of the combustion processes. This narrows down the variety to the LEM and ODT. LEM is ODT's predecessor and rather a model for mixing than combustion ([37]) and therefore it is unsuitable. Until now ODT has been used in various applications as a sub-grid scale model for LES (introduced by [13]) as well as a stand-alone model for sundry applications from non-reacting buoyancy-driven flow (e.g. [39]) to jet flames (e.g. [22]). Thus ODT is the method of choice, for it matches the requirements of the process to be simulated and the existing software environment very well. Nonetheless, it needs some adjustment.

As a brief introduction or reminder the first section will be about the basics of the physical phenomenon of turbulence and its numerical modelling. A short introduction to ODT will be given in Section 4.2. Section 4.3 will walk the reader through the changes applied to the original ODT idea and the implementation realised in the code at hand. In Section 4.4 an overview of the features of the new ODT formulation is given along with the influences of the model parameters. Results of turbulent flame and SEC simulations will be shown. Conclusions on the new ODT in Section 4.5 and a short outlook therein complete this chapter.

4.1 Basics of Turbulence

Turbulence appertains to this peculiar and fascinating class of natural phenomena which are at once ubiquitous and enigmatic. Fittingly, among others, famous scientists as Richard Feynman and Sir Horace Lamb have categorized turbulence as (most) important and (most) puzzling [20], which is still true today. Thus, this section will not aim to solve this big mystery but rather shed some light on what we know about it and how we try to include it in simulations of fluid dynamics.

The struggle with turbulence already begins with defining it. Intuitively, most

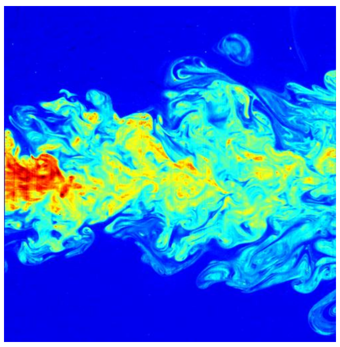

Figure 4.1: Visualization of a turbulent jet via laser-induced fluorescence [28].

people have a rough idea of what is meant by a turbulent flow (like in Figure 4.1), thinking of wild rivers and harsh winds in street canyons. Nonetheless, it gets messy when trying to find the onset of turbulence in an accelerating non-turbulent (usually called laminar) flow undergoing different stages of transition. In our everyday language, things are said to be “turbulent” if they are restless, unruly and unpredictable - somewhat chaotic. This is what definitions of turbulence always include. So our short working definition will be in the style of [18]:

> A flow is turbulent if its velocity field fluctuates randomly in time and is highly disordered in space. It is exceedingly sensitive to changes of initial and boundary conditions, hence mathematically chaotic in the sense of chaos theory.

Note that there are no rotational structures in the definition although they seem to be the first thing which comes to mind, when thinking of turbulent flows. This is due to the fact that turbulence always involves vortices, but vortices do not always indicate turbulence. As already mentioned, between a fully laminar and a fully turbulent flow there are some more stages of flow regimes. Figure 4.2 shows the example of an air stream passing a cylinder at growing mean flow velocity. The first three stages are laminar although we find the famous Karman vortex street in stage three. Nonetheless, the flow is periodic and predictable though more complex. Turbulence only begins in the forth regime as indicated by the ripples in the stream lines, and is fully established in the last one. Figure 4.2 also shows that different scales of length are involved in a turbulent flow.

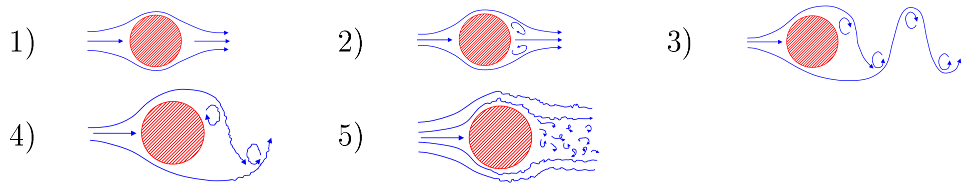

Figure 4.2: Schematic depiction of five stages of a stream passing a cylinder at growing velocity. 1) - 3): laminar, 4) and 5): turbulent. [46]

Now how does turbulence come into existence? It is due to viscosity. When air flows through a pipe, the molecules close to the walls stand still (no-slip condition) while those on the centreline are the fastest. This generates friction between the layers of different velocity. In this case we speak of boundary layers. A similar picture is presented by free shear flows e.g. oil which spills into the quiescent sea: the water molecules rest while the stream of oil carries lots of kinetic energy so there are friction forces at work between the two fluids. Accordingly, viscosity is a prerequisite for friction and friction is the force behind the disturbances of velocity fields and hence the development of complex flows and finally turbulence. But a fluid with high viscosity (think of syrup or resin) is very unlikely to be the protagonist of a turbulent play. This is because high viscosity dissipates the kinetic energy needed for the destabilisation of the flow. Consequentially, what we need to develop turbulence is a minimum ratio of inertial to friction forces

$$Re := \frac{\varrho u_c l_c}{\mu}.$$

Re is called Reynolds number and is widely used in fluid dynamics to quantify the state of a flow with respect to its laminar or turbulent behaviour. u_c and l_c are characteristic velocity and length, which depend on the context, while μ is the dynamic viscosity of the fluid and ϱ denotes its density again. If Re exceeds a critical value, which depends on the (experimental) set-up, a formerly laminar flow develops evermore rotational structures of ever smaller length scales until we finally call it fully turbulent, as seen in Figure 4.2.

If we want to simulate turbulent flows we should know about the features of turbulence. What do flows falling within our definition have in common? Which characteristics must be modelled by our equations? Altogether it is quite hard to talk of a characteristic turbulent flow since most details seem to depend heavily on the specifics. Until today there is no complete overarching theory, just snippets of knowledge based on in-depth observations of a certain set of set-ups. All the more astounding are the hypothesis which have proven to be useful in a wide variety of situations as the energy cascade and Kolmogorov's length scales.

The observation that turbulent flows at high Re numbers exhibit a certain self-similarity due to the fact that their structures span a lot of length scales, led scientist Lewis F. Richardson to the idea of an energy cascade. He brilliantly condensed the concept in his famous poem ([51])

> Big whirls have little whirls
> that feed on their velocity,
> And little whirls have lesser whirls
> and so on to viscosity.

So the biggest eddies are generated by instabilities of the mean flow and pass their kinetic energy on to smaller eddies and so forth until Re based on the smallest vortices is of order unity and viscosity takes on its part to dissipate that energy finally. The life of an eddy is rather short and of the order of its turnover time $\tau \sim \ell/u_e$, where u_e is the eddy's velocity and ℓ its spacial expansion. This hypothesis of the energy cascade includes that, at high Re numbers there exists a so-called inertial subrange with eddies decisively smaller than the biggest ones and as well much bigger than the smallest ones as sketched in Figure 4.3. Within this range the turbulent kinetic energy E_{kin} expressed as a function of eddy wavenumber $1/\ell$ is expected to follow Kolmogorov's five-thirds law: $E_{kin} \sim (1/\ell)^{-\frac{5}{3}}$. The energy cascade and the inertial subrange have been seen in turbulent flows under very different conditions. Nonetheless, when it comes to turbulence, nothing seems to be absolutely universal.

Holding on to the energy cascade, there is another question which has been answered to a surprisingly broad extend: At which length scale η will viscosity take its toll? By using the assumptions that Re for the smallest eddies must be of order unity, passing of energy takes about a turnover time and considering a statistically steady cascade (that is generation and dissipation rate are balanced), one finds

$$\eta \sim \frac{L}{Re^{\frac{3}{4}}}. \tag{4.1}$$

The biggest eddies are of length scale L which is called integral scale. η is called Kolmogorov length scale, together with the velocity of these smallest eddies, it belongs to the Kolmogorov microscales.

From an engineering point of view these results might be pretty but they do not really depict what turbulence does to a flow. Consider two flows - one being laminar, one turbulent - with nominally the same boundary and initial conditions, especially having the same Re number. What would be different in the turbulent flow? At

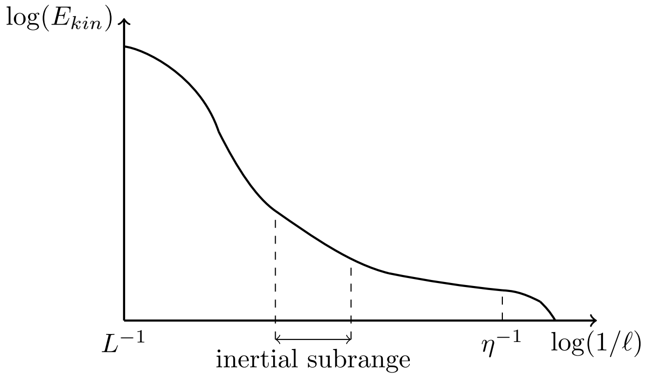

Figure 4.3: Schematic illustration of the turbulent energy cascade with inertial subrange. E_{kin} denotes kinetic energy, ℓ eddy length, L the integral length scale and η the Kolmogorov length scale.

first, of course, we have the random fluctuations of quantities as the velocity and pressure field, the rotation from eddies of all sizes and its dissipation. These things could be seen through velocity probes. But the effects for applications reach deeper: A turbulent flow shows new random extrema but also decays and diffuses faster - all surely due to the eddies and their structure. Turbulent diffusion actually exceeds molecular diffusion by a long shot and hence increases momentum, heat, mass and species transfer remarkably.

Since we now have some examples, pictures and words for turbulent flows at hand, let us take a look at the equations. As we have seen, we need to add viscosity to our system for the development of turbulence. Thus, the Navier-Stokes equations are our candidates. We will regard only the three-dimensional but incompressible version of the momentum equation as it is the most efficient and common way to clarify which problem arises

$$\varrho\frac{D\vec{u}}{Dt} = \varrho\left(\frac{\partial\vec{u}}{\partial t} + (\vec{u}\cdot\nabla)\vec{u}\right) = -\nabla p + \mu\Delta\vec{u}, \tag{4.2}$$

with μ being the dynamic viscosity. This equation is deterministic but nonlinear which is the reason for its tendency towards strong reactions to small perturbations in the initial conditions. This is why real world experiments with turbulent fluids show the typical “random” behaviour: The conditions of two realisations of an experiment can never be exactly the same. So for mathematicians turbulence is *deterministic* chaos, for experimentalists it is not. Hence, it seems reasonable to take on a statistical view on turbulence. Experiments told us that the velocity fluctuation of one point in space and time $u'(x,t)$ is normally distributed over the realisations of the same experiment. So we can define an ensemble averaged flow velocity $\langle u(x,t)\rangle$

(sometimes named $\bar{u}$ which we will use for a different average) which will simply be the mean value of all the measurements in the limit of statistical convergence. Now we perform the so-called Reynolds decomposition by separating the instantaneous velocity field into mean and fluctuation component $u(x,t) = \langle u(x,t)\rangle + u'(x,t)$. Surely, this can be done to other quantities as well. In many applications it is sufficient to know about the average behaviour of a flow, so the idea of reformulating the Navier-Stokes equations (4.2) in terms of the mean values seems natural. The result are the Reynolds-averaged Navier–Stokes equations (RANS) for every mean velocity component $\langle u_i\rangle, i \in \{1,2,3\}$. The summation convention will be used throughout this section - but this section only - as it is a very common practice in the field and enhances readability while maintaining the comprehensibility. So the RANS equations read

$$\varrho\frac{\partial\langle u_i\rangle}{\partial t} + \varrho\frac{\partial\langle u_i\rangle\langle u_j\rangle}{\partial x_j} = -\frac{\partial\langle p\rangle}{\partial x_i} + \frac{\partial}{\partial x_j}\left[2\mu\langle\mathcal{S}_{ij}\rangle - \varrho\langle u_i' u_j'\rangle\right] \tag{4.3}$$

with mean strain-rate tensor $\langle\mathcal{S}_{ij}\rangle := \frac{1}{2}\left(\frac{\partial\langle u_i\rangle}{\partial x_j} + \frac{\partial\langle u_j\rangle}{\partial x_i}\right)$. At first, this might look promising, but a second glance reveals that the term $\tau_{ij}^R := -\varrho\langle u_i' u_j'\rangle$ appeared and cannot be resolved by what we know. For its impact on the equations it is called Reynolds stress (tensor) and it gives rise to the closure problem of turbulence theory since we cannot express it as a function of mean flow quantities but need additional assumptions to cope with it. The closure models most common are called eddy viscosity or turbulent viscosity models. They try to estimate the dynamic turbulent viscosity μ_T in Boussinesq's equations which complement the RANS equations

$$\tau_{ij}^R = 2\mu_T\langle\mathcal{S}_{ij}\rangle - \frac{\varrho}{3}\langle u_k' u_k'\rangle\delta_{ij}. \tag{4.4}$$

The first proposition was Prandtl's mixing length model $\mu_T = \varrho l_m^2\left|\frac{\partial\langle u_i\rangle}{\partial x_j}\right|$, where l_m is that very mixing length, which is situation-dependent and hence to be found through experiments. Within engineering applications the most popular closure attempt is the k-ε model. It describes the eddy viscosity in terms of turbulent kinetic energy k and dissipation rate ε

$$\mu_T = \varrho c_\mu\frac{k^2}{\varepsilon}, \tag{4.5}$$

$$\frac{\partial k}{\partial t} + (\langle u\rangle\cdot\nabla)k = \nabla\cdot\left(\frac{\mu_T}{\varrho\sigma_k}\nabla k\right) + \left(\frac{\tau_{ij}^R}{\varrho}\right)\langle\mathcal{S}_{ij}\rangle - \varepsilon, \tag{4.6}$$

$$\frac{\partial\varepsilon}{\partial t} + (\langle u\rangle\cdot\nabla)\varepsilon = \nabla\cdot\left(\frac{\mu_T}{\varrho\sigma_\varepsilon}\nabla\varepsilon\right) + c_1\left(\frac{\tau_{ij}^R}{\varrho}\right)\langle\mathcal{S}_{ij}\rangle\frac{\varepsilon}{k} - c_2\frac{\varepsilon^2}{k}. \tag{4.7}$$

We see that there are some constants left, which are usually taken to be $c_\mu = 0.09, \sigma_k = 1, \sigma_\varepsilon = 1.3, c_1 = 1.44, c_2 = 1.92$ out of empiricism. Some of the advantages and disadvantages of this (kind of) model are obvious: The additional computational effort is manageable but the model is a rather heuristic interpolation of experimental data: e.g. (4.7) even is a model and not derived as (4.6). The results it yields can be sufficient for a lot of applications where just an estimation of the net effect of turbulence is needed. Studying the turbulence itself is impossible due to averaging. On the more subtle side, we find that the assumption (4.4) leads to problems with strongly anisotropic and non-homogeneous flows.

So let us turn to the alternatives. One could be tempted to just solve the Navier-Stokes equations and at first, there would be nothing wrong about it. This method is commonly called direct numerical simulation (DNS) and used to conduct research on simple forms of turbulence and validate coarser modelling suggestions. The necessity of these limitations is instantly clear, when recalling that the full range of length and time scales must be resolved in order for DNS to yield sensible results. Yet, for reasonable Re numbers of e.g. 10^6 and a domain length scale of order unity, this means hundreds of thousands of grid cells - for each dimension. Even if restricted to two dimensions (which we need to generate turbulence at all in this context) it requires billions of grid cells. Of course the argument stretches out to time scales in a similar way. Therefore, until today DNS can only be used for small domains with low Re numbers and simple boundary conditions and thus strongly reduced closeness to reality even on very powerful state-of-the-art computers - at least if we want to see the results within our own lifespan.

Within the scope of this work, there is not much more to say about the advantages and drawbacks of DNS so we move on to something more enticing: large-eddy simulations (LES). It has been found that (for a lot of set-ups) the largest vortices are the most important structures in terms of turbulent transport. On that account, the somewhat compromising LES was invented. The idea is reminiscent of the RANS ansatz, decomposing the flow quantities into a resolved filtered and an unresolved residual component. In the case of LES we resolve the large scale eddies, hence turbulence is included, and model the small scales for energy dissipation. In a little more detail: We smooth out the quantity - let us stick with velocity here - by convolving the instantaneous function u with a filter function h, e.g.

$$\hat{u}_1(x) = (u_1 * h)(x) = \int_{-\infty}^{\infty} u_1(x-r)h(r)dr,$$

$$h(r) = \begin{cases} 1/L & |r| \leq L/2, \\ 0 & |r| > L/2. \end{cases}$$

The effect of this filtering is that we even out fluctuations much smaller than L. Surly, there are a lot of possible filtering functions and the procedure carries over to more dimensions. Now, we write the new decomposition $u = \hat{u} + \grave{u}$ and insert it into the Navier-Stokes equations

$$\varrho\frac{\partial \hat{u}_i}{\partial t} + \varrho\frac{\partial(\hat{u}_i\hat{u}_j)}{\partial x_j} = -\frac{\partial \hat{p}}{\partial x_i} + \frac{\partial \tau^r_{ij}}{\partial x_j} + \mu\nabla^2\hat{u}_i,$$

$$\tau^r_{ij} = \varrho(\hat{u}_i\hat{u}_j - \widehat{u_i u_j}).$$

This resembles (4.3) very much and again we ran into notional stresses, the residual stresses τ^r_{ij}, and need to employ a model guessing them. Here, too, we can use eddy viscosity models - preferable one that has been created especially for LES like the Smagorinsky model or its offspring - where our filtering scale L is included. With LES we can use much coarser grids of width L instead of η giving us a wider range of possible applications than DNS and still we see the big structures of turbulence yielding more accuracy than RANS simulations. Self-evidently, LES has its restrictions, too. The computations are, nevertheless, much more demanding in terms of resources than k-ε models and the like and although the field of reasonable applications for LES is very large, there are set-ups where the small scales are too important to be neglected like that, e.g. when studying turbulent flows near a wall.

The modelling concepts we have seen so far either use the underlying equations directly or average the flow quantities to different extends. For some this might seem as if we have exhausted all possibilities here but then again we have not yet scraped the bottom of the barrel. When defining turbulence, we used the term "randomly" but none of the models above has actually considered simulating the mean flow plus random fluctuations. Naturally, great care must be taken, when deciding for distribution functions but the same applies to closure models, so this should not stop us from going that way. In fact, this notion is the basis of the one-dimensional turbulence model - though it is not the only one - which will be discussed and worked with in the remainder of this chapter.

For a more detailed introduction on turbulence, the reader is referred to [18], [49], [48], [57], to name only the literature on which this section is based.

4.2 Introduction to ODT

When thinking of turbulence, one-dimensionality seems to be counter-intuitive. Of course turbulence is a highly three-dimensional phenomenon and it is not a coincidence that extremely costly computations are needed to simulate it in a direct manner. Nonetheless, decades of investigations gave some insight on the processes that drive turbulence and are driven by it. These findings enabled the development of phenomenological simulations of turbulence which can also be executed in one dimension. There are different methods of simulating individual realisations of turbulent flows via evolution equations, e.g. one-dimensional Biot-Savart formulation [16], one-dimensional binary-tree formulations [3] and using a stochastic approach, like the linear eddy model (LEM) [36] and the one-dimensional turbulence (ODT)

[39]. We decide for the latter as it is the most advanced in the field of turbulent combustion, exploiting knowledge about the forces behind and characteristics of turbulence in a computationally efficient way.

The most important features of turbulence have been introduced in Section 4.1: random velocity fluctuations, meaningful mean values, wide ranges of time and length scales, high turbulent diffusion and rotational structures. We have also discussed some of the more universal theories: the energy cascade, Kolmogorov's five-thirds law and the Kolmogorov microscales, as well as the generation of turbulence by viscous forcing through velocity differences. ODT is able to reproduce these features without directly simulating turbulence but stochastically choosing eddies which compress and rotate states. The sophisticated algorithm makes the interaction of eddies and current stream possible and reasonable. In the following, ODT will be presented as formulated originally.

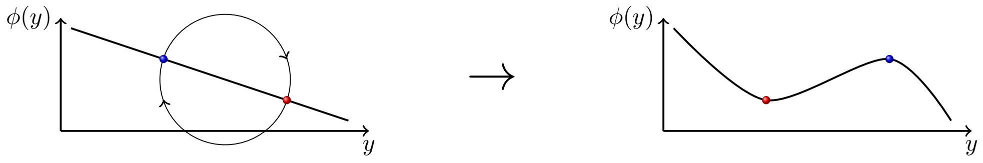

Figure 4.4: Effect of a rotation on a scalar function in one space dimension.

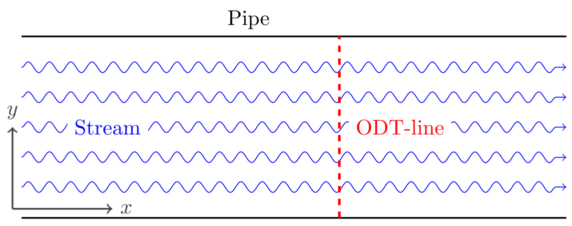

Figure 4.5: Sketch of pipe flow with stream lines, ODT-line and corresponding axis directions x and y.

In 1999 ODT was presented by Kerstein in [39] as a stochastic model, which is to be used in a Monte Carlo simulation. It produces single realisations of a turbulent flow simulation which can be averaged over the ensemble but also studied by themselves. This is a feature which makes ODT more attractive for the application within the SEC-code than RANS or LES as combustion is a highly sensitive process and it might be possible that one realisation yields a proper SEC while another one does not.

Turbulent advection and thus the cause of random quantity fluctuations, is represented by so-called eddy events. These eddy events are the one-dimensional net effect of a vortex, a kind of projection of a rotation (see Figure 4.4). In ODT they are realised as simple mappings which act on an ODT-line with direction y that is perpendicular to the streamwise direction x as illustrated in Figure 4.5. This formulation is due to the shear flow frame where the streamwise velocity profile $u(y)$ drives the turbulence. Generally, the ODT direction y would be the one with the steepest gradients of properties. Calling ℓ length of the eddy and y_0 its bottom corner, an eddy event is implemented as a mapping of the domain segment $[y_0, y_0 + \ell]$ onto itself by the three-valued triplet map

$$\hat{y}(y) := \begin{cases} y_0 + c_1(y - y_0), \\ y_0 + c_2\ell - (c_2 - c_1)(y - y_0), \\ y_0 + c_2\ell + (c_2 - c_1)(y - y_0) \end{cases} \tag{4.8}$$

with $0 < c_1 < c_2 < 1$. In most cases a symmetric formulation with $c_1 = \frac{1}{3}$ and $c_2 = \frac{2}{3}$ is used, but other options might be advantageous for a specific application. This mapping is measure-preserving and hence, all velocity moments like momentum and kinetic energy are preserved, too. Also its inverse is continuous. Applying this map to any given profile $\Phi(y)$ on $[y_0, y_0 + \ell]$ results in three compressed images of this profile with the middle one mirrored for continuity. This produces a profile similar to the one seen in Figure 4.4. Thus $\hat{y}$ reflects the compressive and rotational effects of real turbulence and rightly increases strain intensity as well as it decreases strain length-scale. This is how ODT realises self-similar eddy cascades and therefore the energy cascade. The map can be implemented as a permutation of grid cells which makes it particularly efficient in terms of computation time ([38]). The basis of this mapping is Kerstein's incompressibility condition. For the scope of this thesis we will go along with it to gain a first insight to the new formulation we want to develop.

The application of $\hat{y}$ to the ODT-line presumes that y_0 and ℓ are chosen beforehand. In the ODT context, they are random variables representing a candidate eddy which might be implemented. These variables could just be drawn from a uniform distribution which would not have much effect on the outcomes of the simulation due to the ensuing steps. Nonetheless, at least for ℓ it is computationally advantageous to use a more elaborate distribution ([47]) which we will see later on.

Until now, there was no interaction of eddies and stream condition. This changes when coping with the question of when to implement the sampled eddy. Considering waiting processes, it is common to regard events as Poisson distributed. This means, that

$$\delta_{\Delta t_{eddy}} := \mathcal{R}(t)e^{-\mathcal{R}(t)t}$$

with rate

$$\mathcal{R}(t) = \int_{y_{min}}^{y_{max}} \int_{\ell_{min}}^{\ell_{max}} \lambda(\ell, y_0; t)\, d\ell\, dy_0.$$

The function $\lambda(\ell, y_0; t)$ plays the role of a rate distribution of eddies. $\lambda(\ell, y_0; t)d\ell$ represents the frequency of eddy events of size $[\ell, \ell + d\ell]$ per unit length along the ODT-line. Due to dimensional arguments

$$\lambda(\ell, y_0; t) := \frac{1}{\ell^2 \tau(\ell, y_0; t)}.$$

Kerstein explicitly refrains from inserting a free parameter in his scope of application. Later in this chapter, we will introduce a factor as $\mathcal{C}$. τ is named eddy time scale - usually interpreted as a fraction or multiple of the turnover time $\beta_T \tau$, with β_T being another model parameter which we will use later on. We know that it is the difference in the streamwise velocity along the y-axis that drives the turbulence in a real-world experiment. Hence, the same applies for the eddy within the frame of [39] and therefore $\Delta u(\ell, y_0; t)$ is introduced along with a free parameter $\mathcal{A}$ to compute τ

$$\tau(\ell, y_0; t) := \frac{\ell}{\mathcal{A} \Delta u(\ell, y_0; t)}. \tag{4.9}$$

$\mathcal{A}$ accounts for different definitions of $\Delta u(\ell, y_0; t)$ for there is no preferable one. Kerstein chooses

$$\Delta u(\ell, y_0; t) := 2|u_\ell(y_0 + 0.5\ell, t) - u_\ell(y_0, t)|$$

with

$$u_\ell(y, t) := \frac{2}{\ell} \int_{y}^{y+\ell/2} u(\zeta, t) d\zeta. \tag{4.10}$$

(4.10) smoothes the velocity profile because preceding eddies have made it spiky.

With this set of equations, one could compute $\lambda(\ell, y_0; t)$ for the current state of the flow and all possible combinations of ℓ and y_0. As this is not a very feasible approach a different technique is exploited instead: thinning (see e.g. [43]). The general idea of this method is to oversample the Poisson distributed variable - the time step Δt_{eddy} until the next eddy is sampled - by using a majorant λ^* of $\mathcal{R}$ as rate for the Poisson process. Afterwards the event is only accepted with a probability of

$$\varphi_a := \frac{\text{real distribution}}{\text{numerical distribution}} = \frac{\lambda(\ell, y_0; t)}{\Lambda(\ell, y_0, t)}.$$

As we can write the numerical rate distribution as a product of the numerical rate Δt_{eddy}^{-1} and the joint probability density function (PDF) of y_0 and ℓ - assuming independence - $\delta_{y_0} \delta_\ell$ ([40],[1]), the acceptance probability reads ([39], Appendix A)

$$\varphi_a(\ell, y_0; t, \Delta t_{eddy}) = \frac{\lambda(\ell, y_0; t) \Delta t_{eddy}}{\delta_{y_0} \delta_\ell}. \tag{4.11}$$

Combining the two steps of sampling candidate eddies and rejecting a suitable portion of them recovers the true distribution if $\mathcal{R}\Delta t_{eddy} \ll 1$ (oversampling condition). The advantage of this method is that it only needs the evaluation of $\lambda(\ell, y_0; t)$ of one distinct eddy with fixed size and position. With reasonable ratios of sampled to accepted eddies under the oversampling condition, this approach saves a lot of computation time.

Last but not least, two suppression mechanisms are part of every ODT simulation: small and large eddy suppression. The former is useful for the likelihood $\lambda(\ell, x_0; t)$ is proportional to ℓ^{-2} but the influence of very small eddies on the flow properties is usually minor. A small eddy suppression also is inline with viscous effects at Kolmogorov microscales and formulated as such. Kerstein suggests, that eddies are applied only if

$$\tau < \tau_d \coloneqq \varrho\ell^2/16\mu. \tag{4.12}$$

Large eddy suppression on the other hand is necessary for, depending on the chosen distribution, unphysically large values of ℓ may be drawn occasionally. These rare events would dominate the flow contributing to transport with a square of their size ([39], [41]). To restore scale locality Kerstein's implementation rejects eddies which contain more than a given fraction of laminar fluid. The latter is defined to still have the initial velocity within a reasonable tolerance. The choice of the fraction value is empirical or arbitrary.

Over the past 20 years a lot of different formulations of and extensions to ODT have been developed along with various mechanisms to cover more set-ups (see [54] and [21] for an elaborate overview). In the next section yet another adaptation which was developed to serve the special needs of the SEC simulations will be shown.

4.3 Implementation

The current implementation of ODT within the SEC-code is inspired by the aODT-code of Heiko Schmidt's group at BTU Cottbus-Senftenberg. Although this code can deal with adaptivity of the mesh, the moving mesh feature described in Chapter 3 is disabled for the ODT feature of the SEC-code, as the notion of ODT in the SEC context is a different one. In lieu of involving some of the later extension yielding a more complex mechanism which is included in the aODT-code, the formulae used in the current SEC-code are following the original ODT paper [39] described in Section 4.2 very closely. Since there is no evaluation of the streamwise velocity profile in a perpendicular (lateral) direction in the SEC-code, one of the most important differences to the ideas stated in Section 4.2 is that the ODT-line will not be orthogonal to the space direction x. Instead it will be aligned with this streamwise direction and thus henceforth y_0 will be substituted by x_0. This means that eddies now occur on the centreline of the flow (centreline assumption), where their own centres will be (see Figure 4.6). Nonetheless, as the turbulence-driving force is the difference in streamwise velocity in the lateral y-direction, we make a guess about the $u(y)$-profile based on the cell-averaged momentum that is resolved by the SEC-code.

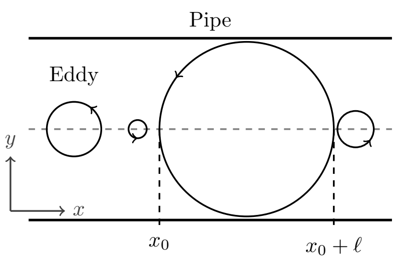

Figure 4.6: ODT eddies in pipe flow in streamwise direction.

The other difference emerges from the usage as a stand-alone subgrid-scale model. Most applications include ODT as a stand-alone turbulence model for fully resolved simulations as in [39] and publications following this path. That would mean $\Delta x \leq \eta$ (Kolmogorov length scale), which grows smaller with higher Reynolds number as can be seen in (4.15). Here ODT is the only turbulence model included. The other type of application is as a subgrid-scale closure model for under-resolved three-dimensional simulations with turbulence-capturing solution methods, e.g. LES (see Section 4.1 for an overview or [13] for details).

The implementation at hand merges the two approaches applying ODT as the only turbulence model but in a subgrid-scale mode. That means we assume Δx to be of the same scale as the maximum eddy size, which is the SEC-tube's diameter. This approach enables the inclusion of a variable cross-sectional area and is justified by the one-dimensional rather qualitative character of the code. Therefore, in contrast to [39], there will not be a discrete triplet map switching cell values. Instead we assume continuous triplet maps working on a scale which is not resolved by our grid. Because they are measure-preserving triplet maps only change a grid cell's state value if they cross the interface between two cells. The current implementation only allows for two cells to be involved in an eddy event. In this case a linear approximation of the flow properties within each grid cell is applied before the triplet mapping as shown in Figure 4.7. By comparing the new cell integral to the old one, the flux between the two cells is computed and hence axial area variation of the domain can be accounted for, using the notion of (3.26) and (3.27). The details to this procedure and more on the current implementation can be found in Subsection 4.3.1.

4.3.1 Detailed Workflow

The overall workflow of an SEC simulation with ODT is shown in Figure 4.8, where we see that the code executes the following steps: At first the sampling time step

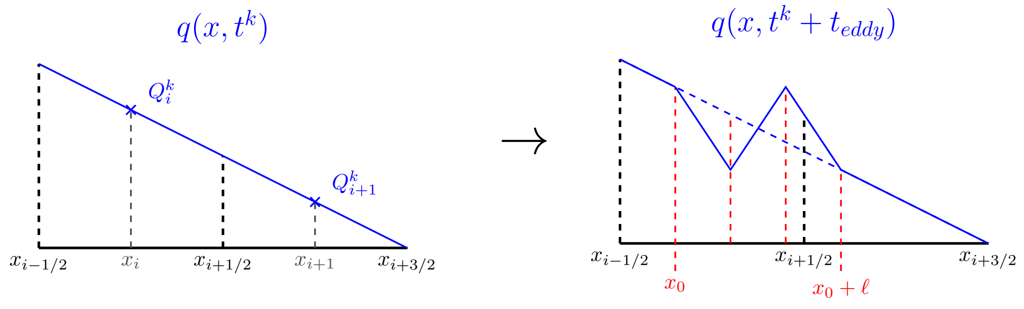

Figure 4.7: Subgrid-scale triplet mapping over the interface of two grid cells.

Δt_{eddy} must be drawn from the exponential distribution

$$\delta_{\Delta t_{eddy}}(t) = \lambda^* e^{-\lambda^* t}.$$

λ^* is the majorant of the real eddy rate $\mathcal{R}$ needed for the thinning method as discussed in Section 4.2. Its first value is chosen by the user. Afterwards the algorithm will change it according to its settings (see below). If the sampling time falls in between the current and next overall time level, the ODT algorithm will be started. It is implemented in C++, which saves some computation time. Needing to simulate a lot of realisations and possibly drawing a huge number of eddies (most of them being rejected or not influencing the cell values), this can be very helpful. The ODT function itself begins with sampling ℓ and x_0 for the potential eddy's size and position. As outlined in [39], the choice of the PDFs δ_ℓ and δ_{x_0} is arbitrary but will effect the algorithm's computational efficiency. This is why S. Wunsch (see [47]) developed a function which is widely used to sample the eddy size from. Generally it looks like

$$\delta_\ell(\ell) = a\ell^b \exp\left(c\left(\frac{\ell_p}{\ell}\right)^{-(b+1)}\right). \tag{4.13}$$

Naturally, a is the normalisation factor which includes ℓ_{min} and ℓ_{max}, the user-defined minimum and maximum eddy size, respectively

$$a = \frac{(b+1)c\ell_p^{-(b+1)}}{\exp\left(c\left(\frac{\ell_p}{\ell_{max}}\right)^{-(b+1)}\right) - \exp\left(c\left(\frac{\ell_p}{\ell_{min}}\right)^{-(b+1)}\right)}.$$

b is a scaling factor which could be $b = -8/3$ if the assumptions leading to Kolmogorov scaling hold true or $b = -2$ like advised in [47] and used in [45], [50] and many others as a precaution. In this way we oversample bigger eddies rather than smaller ones, ensuring to include enough of the energy-containing large scales. To force the distribution to have its maximum at the user-specified ℓ_p, one sets $c = \frac{-b}{b+1}$, making ℓ_p the most probable eddy size by definition. Inserting the above choices in

```
set up domain with ODT parameter (see Appendix B.3 for instructions);
sample first $\Delta t_{eddy}$;
while $t < t_{end}$ do
    compute $\Delta t^k$;
    if $\Delta t^k > \Delta t_{eddy}$ then start ODT function
        $t_{ODT} = t$;
        while $t_{ODT} < t + \Delta t^k$ do
            begin sample eddy:
                sample size $\ell$ from (4.14);
                sample position $x_0$ from (4.16);
                compute $\tau$ (4.9);
                compute acceptance probability $\varphi_a$ (4.11);
            end
            check acceptance;
            raise or lower $\lambda^*$;
            if eddy accepted then
                add eddy to statistics;
                if eddy crosses cell interface then
                    set $\Delta t^k = t_{ODT} - t$;
                    sample next $\Delta t_{eddy}$;
                    break;
                end
            end
            set $t_{ODT} = t_{ODT} + \Delta t_{eddy}$;
            sample next $\Delta t_{eddy}$;
        end
    end
    if last sampled eddy crosses cell interface then
        apply triplet map;
        update statistics;
    end
    advance governing equations;
    $t \leftarrow t + \Delta t^k$;
end
```

Figure 4.8: Workflow with ODT

(4.13) yields

$$\delta_\ell(\ell) = a\ell^{-2} \exp\left(-\frac{2\ell_p}{\ell}\right) \tag{4.14}$$

with

$$a = \frac{2\ell_p}{\exp\left(-\frac{2\ell_p}{\ell_{max}}\right) - \exp\left(-\frac{2\ell_p}{\ell_{min}}\right)}.$$

Of course, an exponential function never really reaches zero so the support of δ_ℓ actually is $\mathbb{R}$. Still, we can prevent drawing eddies that are too big or too small by computing ℓ as

$$\ell = -2\ell_p \left[\log\left(\sigma \exp\left(\frac{-2\ell_p}{\ell_{max}}\right) - (\sigma+1)\exp\left(\frac{-2\ell_p}{\ell_{min}}\right)\right)\right]^{-1},$$

where σ is drawn from the uniform distribution over $[0, 1]$

$$\sigma \sim \mathrm{U}(0, 1).$$

This gets us the desired distribution of sizes but only within $[\ell_{min}, \ell_{max}]$ ([50]). In [50] the user-set lengths are linked to turbulence theory (cf. (4.1)) and scaling analysis by choosing

$$\eta = \frac{\ell_{max}}{Re^{0.75}}, \quad \ell_{min} = 6\eta, \quad \ell_p = \exp\left(\frac{\ln(\ell_{max}) + \ln(\eta)}{2}\right) = \sqrt{l_{max}\eta}. \tag{4.15}$$

Again Re is the Reynolds number of the flow and ℓ_{max} is reasonably chosen to match the integral length scale. ℓ_{min} is set to six times Kolmogorov length scale because in the framework of [50] the triplet maps are discontinuous and implemented to permute grid cell values. The resolution of said grid is $\Delta x = \eta$, hence an eddy event can only be performed if at least six grid cells are involved. As the triplet map implemented in the SEC-code is continuous, ℓ_{min} could just as well be set to η. An example of δ_ℓ for $Re = 10^6$, (a value relevant to the simulations of Section 4.4) is shown in Figure 4.9.

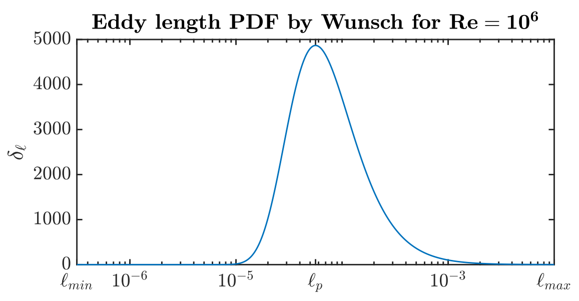

Figure 4.9: Example probability distribution function for eddy sizes after Wunsch with $Re = 10^6$.

Examining the simulations in Section 4.4, we will see that the position distribution greatly depends on the setting and therefore any other ansatz would be very specific to the task at hand and need a fair portion of a priori knowledge. Accordingly, no complex functions have been created for the eddy's leftmost edge x_0 and a uniform distribution is used

$$\delta_{x_0}(x_0;\ell) = \frac{1}{\mathcal{L}-\ell}, \tag{4.16}$$

where $\mathcal{L}$ is the domain's size.

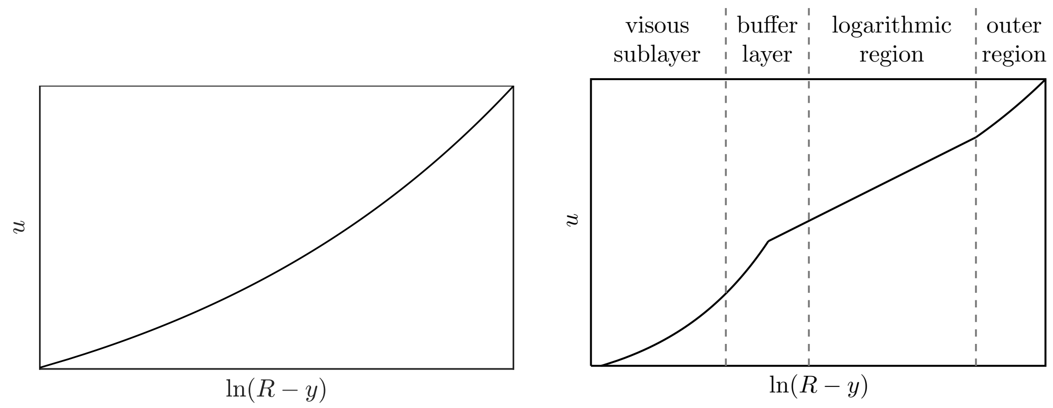

Figure 4.10: Qualitative comparison of implemented lateral u profile (left) with "law of the wall" (right), $y \in [0,R]$ from centreline to wall.

Afterwards the eddy's time scale τ is computed using Kerstein's original formulation $\tau(\ell,x_0;t) = \frac{\ell}{\mathcal{A}\Delta u(\ell,x_0;t)}$, $\mathcal{A}$ being a model parameter. To be able to evaluate $\Delta u(\ell,x_0;t)$ we need an assumption for the profile of the streamwise velocity in lateral direction because we do not resolve the velocity along this dimension. Since the SEC-code is designed for pipe flows an expression close to the "law of the wall" (see [18], section 4.2, for details), which describes the desired profile in a fully developed turbulent flow between walls, seems to be the obvious choice. Looking into the details, though, it divides the flow into four regions. This is already too complex for the question at hand. We just need a rough estimation for the turbulence driving force, i.e., the velocity difference Δu between eddy centre and eddy edge. This can be done by a much simpler formula common in technical applications (see [4], p. 182)

$$u(y) := u_{max}\left(1-\frac{y}{R}\right)^{1/\psi} \quad \text{with} \quad \psi = 2.1\log_{10}(Re) - 1.9. \tag{4.17}$$

The spacial variable y is defined from the centreline $y=0$ to the wall $y=R$, where R is the radius of the pipe (in our case: the combustion chamber). Please note that the definition of Re in this context is

$$Re = \frac{2R\overline{\varrho u}}{\mu}.$$

Figure 4.10 shows an exemplary profile based on (4.17) in comparison to the much more complex "law of the wall". Our chosen profile seems to be a good approximation under the given requirements. Since we only have the momentum averaged over the control volume $\overline{\varrho u}$ available, a link between $\overline{\varrho u}$ and u_{max} is needed. First, we estimate $\bar{u}$ through $\frac{\overline{\varrho u}}{\bar{\varrho}}$. Next, we follow the approach of [34] (ansatz can be checked in [4], p.338, 1.28). Using the definition of the volume flow $\dot{V}$ as $\bar{u}$ times cross-sectional area A and inserting (4.17) yields

$$\bar{u} = \dot{V}/A = \frac{1}{\pi R^2}\int\limits_{-R}^{R} \pi y u(y) dy = \frac{2}{R^2}\int\limits_{0}^{R} y u(y) dy = 2\int\limits_{0}^{1} r u(r) dr$$

$$= 2u_{max}\int\limits_{0}^{1} r\,(1-r)^{1/\psi}\, dr = 2u_{max}\cdot\frac{\psi^2}{2\psi^2+3\psi+1}.$$

For the partial integration in the last step we used $\psi > 0$, which is always true for $Re > 1$. This is a reasonable assumption for turbulent flows. Now Δu can be set to $|u_{max} - u(\ell/2)|$. A smoothed out function for u is not necessary in our context since the $u(y)$-profile is not subject to triplet mapping as in [39] but already a smooth analytic function.

Having found an estimation for Δu, we can compute τ. Small and large eddy suppression mechanisms are included in this substep. In reality small eddies fall prey to viscous dissipation, which is modelled following Kerstein's suggestion in [39] by demanding $\tau < \varrho\ell^2/16\mu$ and otherwise rejecting the sampled eddy. Overly large eddies are not as big a problem in our subgrid-scale framework since eddy size is restricted to combustion chamber diameter and still smaller than or equal to Δx. Nonetheless, there is a mechanism included which gives the user control over the onset of slow eddies. Like in e.g. [22], the eddy turnover time is interpreted as some model parameter β_T times eddy time scale τ. Unphysically slow eddies are now avoided by simply imposing the restriction that the elapsed simulation time must be greater than the eddy turnover time: $t \geq \beta_T\tau$. This seems to be a sensible requirement. Why this does actually not suppress large eddies although it is a large eddy suppression mechanism will be explained in Subsection 4.4.3.

With τ and the model parameter $\mathcal{C}$, $\lambda(\ell, x_0; t) = \frac{\mathcal{C}}{\tau\cdot\ell^2}$ can be calculated and therefore the acceptance probability. As our triplet map is used continuously, we do not need to account for different mean-square displacement of the discrete triplet map ([40]) and readily compute φ_a from (4.11). The sampled eddy is implemented, if a number randomly drawn from U(0,1) is lower than φ_a. Please note that it is not guaranteed that $\varphi_a < 1$, at least because $\mathcal{A}$ and $\mathcal{C}$ are involved but it is also true if both parameters are set to unity. If φ_a is greater than 1, Δt_{eddy} is too big and λ^* not a majorant of the true eddy rate as required for the thinning procedure. This is why there is a mechanism raising λ^* if φ_a is greater than some given constant in $(0, 1)$. Also, after a preset number of drawn eddies λ^* may be lowered to enhance computational efficiency.

If the sampled eddy is rejected, the next waiting time and corresponding eddy will be sampled until the governing equations of the flow need to be advanced. Properties of accepted eddies are stored for statistical evaluation. If the eddy does not cross a grid cell interface, the ODT algorithm continues sampling eddies. Otherwise, the eddy is to be implemented and the global Δt^k is changed to match the eddy's time of occurrence, which is current time level plus the waiting time of all eddies sampled within this step (including the rejected ones).

If an eddy occured which spans two grid cells $i \in \{1, ..., N-1\}$ and $i+1$, the triplet map must be applied. To this end, we adopt a linear approximation of q over the two involved cells (see Figure 4.7). If there is no variation of cross-sectional area, we only need to compute the new cell-averaged integral values Q_i^k and Q_{i+1}^k and we are done. Otherwise, we need to find the flux per unit area. Remember

$$Q_i^k = \frac{1}{A_i \Delta x} \int_{x_{i-1/2}}^{x_{i+1/2}} q(\zeta, t_k) A(\zeta) d\zeta$$

with $A_i = \frac{1}{2}\left(A_{i-1/2} + A_{i+1/2}\right)$. Now the flux over the cell interface equals the difference of the grid cell integrals after and before the eddy divided by its turnover time

$$\mathcal{E}_{i+1/2}^k := \frac{1}{\beta_T \tau} \left(\int_{x_{i-1/2}}^{x_{i+1/2}} q(\zeta, t_k + \beta_T \tau)\, d\zeta - \int_{x_{i-1/2}}^{x_{i+1/2}} q(\zeta, t_k)\, d\zeta \right). \tag{4.18}$$

Because only one interface is involved, we just need to calculate the flux once for grid cell i, due to conservation laws. Also it follows that $\mathcal{E}_{i-1/2}^k = \mathcal{E}_{i+3/2}^k = 0$. Accordingly, the new state vectors of the cells are calculated like in (3.26) and (3.27) via

$$Q_i^{k+1} = Q_i^k + \frac{\beta_T \tau}{\Delta x} \frac{A_{i+1/2} \mathcal{E}_{i+1/2}^k}{A_i}, \tag{4.19}$$

$$Q_{i+1}^{k+1} = Q_{i+1}^k - \frac{\beta_T \tau}{\Delta x} \frac{A_{i+1/2} \mathcal{E}_{i+1/2}^k}{A_{i+1}}. \tag{4.20}$$

Notice that $\beta_T \tau$ cancels out.

After this step λ^* and the statistics of implemented eddies are updated and the governing equations are advanced to catch up with the eddy time line.

4.3.2 Molecular Transport

Since turbulence requires viscosity to come to life the governing equations (1.1) were extended to the full set of Navier-Stokes equations with chemistry

$$\frac{\partial}{\partial t}\begin{pmatrix}\varrho\\ \varrho u\\ \varrho E\\ \varrho Y\end{pmatrix} = -\frac{\partial}{\partial x}\begin{pmatrix}\varrho u\\ \varrho u^2+p\\ (\varrho E+p)u\\ \varrho Y u\end{pmatrix} + \begin{pmatrix}0\\ 0\\ \varrho \dot{E}_{chem}\\ \varrho \dot{Y}_{chem}\end{pmatrix} + \frac{\partial}{\partial x}\begin{pmatrix}0\\ \frac{4}{3}\mu\frac{\partial u}{\partial x}\\ \frac{4}{3}\mu\frac{\partial u}{\partial x}u+\kappa\frac{\partial T}{\partial x}\\ \varrho D\frac{\partial Y}{\partial x}\end{pmatrix} \tag{4.21}$$

with the transport coefficients dynamic viscosity μ, thermal conductivity κ and vector of mass diffusion coefficients D. All of these coefficients depend on the specific composition of the gas and its temperature. Therefore, for real chemistry set-ups, they are calculated as mixture-averaged values following [35], which is also the basis of the often used Cantera code

$$\begin{aligned}
\kappa =& \frac{1}{2}\left(\sum_{i=1}^{N_{spec}} X_i\kappa_i + \frac{1}{\sum_{j=1}^{N_{spec}} X_j/\kappa_j}\right),\\
D_i =& \frac{1-Y_i}{\sum_{\substack{j=1\\ j\neq i}}^{N_{spec}} X_j/D_{ij}}, \quad \forall i \in \{1,...,N_{spec}\},\\
\mu =& \sum_{i=1}^{N_{spec}} \mu_i \frac{Y_i}{\sum_{j=1}^{N_{spec}} \Gamma_{ij}\frac{M_i}{M_j}Y_j} \quad \text{with}\\
\Gamma_{ij} =& \frac{1}{\sqrt{8}}\left(1+\frac{M_i}{M_j}\right)^{-1/2}\left[\left(1+\frac{\mu_i}{\mu_j}\right)^{-1/2}\left(\frac{M_j}{M_i}\right)^{1/4}\right]^2.
\end{aligned}$$

Here X_i are mole fractions, N_{spec} the number of species, μ_i and κ_i refer to single species viscosity or conductivity, respectively. D_{ij} is the binary mass diffusion coefficient. As D_i is not defined should the considered state be composed of a single species, we use $D_i = D_{ii}$, the self-diffusion coefficient, as a fall back solution for this case. Additionally, [35] notes that mixture-averaged mass diffusion coefficients violate mass conservation. Hence, a flux correction was incorporated following the "conservation diffusion velocity" idea of [14] as stated in [35]. First we rewrite the species diffusion flux from (4.21) using the diffusion velocity W

$$\varrho D \frac{\partial Y}{\partial x} \to \varrho Y W. \tag{4.22}$$

Now we decompose $W = \widetilde{W} + W_c$, where $\widetilde{W}$ is the diffusion velocity as usual

$$\widetilde{W} = \begin{cases} D \frac{1}{Y} \frac{\partial Y}{\partial x} & Y > 0, \\ 0 & Y = 0 \end{cases}$$

and W_c the correction

$$W_c = - \sum_{i=1}^{N_{spec}} Y_i \widetilde{W}_i.$$

Note, that the latter only depends on space and time and no longer on the species.

Molecular transport is included in the SEC-code via Strang splitting. The numerical flux function $\mathcal{D}$ is taken directly from (4.21), employing (4.22), and evaluated for states reconstructed at the cell edges by common linear approximation

$$Q^k_{i-1/2} = \left(Q^k_{i-1} + Q^k_i\right)/2, \quad \forall i \in \{2, ..., N\}.$$

Derivatives are calculated as central differences. For the time update, an explicit-Euler-like one-step method

$$Q^{k+1}_i = Q^k_i + \frac{\Delta t^k}{\Delta x}\left(\mathcal{D}^k_{i+1/2} - \mathcal{D}^k_{i-1/2}\right)$$

and a Runge-Kutta-like two-step method

$$Q^{k+1/2}_i = Q^k_i + \frac{\Delta t^k}{2\Delta x}\left(\mathcal{D}^k_{i+1/2} - \mathcal{D}^k_{i-1/2}\right),$$

$$Q^{k+1}_i = Q^k_i + \frac{\Delta t^k}{\Delta x}\left(\mathcal{D}^{k+1/2}_{i+1/2} - \mathcal{D}^{k+1/2}_{i+1/2}\right)$$

were implemented in FV formulation.

Molecular transport has also been introduced in the MM-variant of the SEC-code although it is not used with the full ODT. To achieve this, the above approximations had to be adjusted for the non-equidistant mesh. The derivatives are evaluated using the Fornberg-algorithm described in Subsection 3.2.3 and one of the limiter functions included in the SEC-code. The quantities on cell edges are calculated using these derivatives.

governing equations	quasi one-dimensional Euler with ODT-FHD
chemistry	2-species Arrhenius kinetics; turned off
domain	length: 1, area variation: none
initial values	$p = 2$, $T = 1.219$, $u = 1$, fuel at $x \in [0, 0.1]$ and product else
boundaries	left: $p = 2$, $T = 1.219$, $u = 1$, product species; right: isentropic expansion to $p = 2$
ODT setting	150 realisations, $\mathcal{A}, \mathcal{C} \in \{0.5, 1, 2, 3, 5\}$, $\beta_T \in \{0.1, 0.5, 1, 2, 3, 5, 10\}$, $\ell_{min} = 10^{-5}$, $\ell_{max} = 0.01$
grid cells	100
time steps	step size chosen automatically, snapshots stored at multiples of 4×10^{-3}

Table 4.1: Settings for homogeneous pipe flow simulations.

4.4 The ODT-FHD

The ODT implementation in the current SEC-code differs conceptually from the original ODT and therefore, surely bears other features which will be studied in the following. Towards this aim, an idealised simulation of a classic experiment has been carried out: a homogeneous steady pipe flow with transported fuel species package (see Table 4.1). The domain of length 1 is initialised with $p = 2$, $T = 1.219$, $u = 1$ everywhere. Fuel species is set to the first 10% of the domain, product species fills the rest. Chemistry is disabled to rededicate the fuel species as a tracker. The boundary conditions keep the flow steady for the left one is constant with $p = 2$, $T = 1.219$, $u = 1$ and the right one is an isentropic expansion to $p = 2$. 150 realisations are simulated for each set of model parameter which were taken from a small range. ℓ_{max} is set to 0.01 which is the order of a combustion chamber diameter and equal to Δx, $\ell_{min} = 10^{-5}$ for no smaller eddies are accepted by the algorithm (cf. Subsection 4.4.3).

All flow properties except the fuel species remain untouched by turbulence and unchanged by gasdynamics as there are no gradients within them. Hence, this setting is eminently suitable to investigate this ODT's features without having to account for complex mutual interaction of turbulence and mean flow. The mean flow merely drives the turbulence in this case and advects the fuel package.

We will see that the new ODT formulation gives us a tool for controlling turbulent diffusion. As implemented, there will be diffusive fluxes rather than generation of new extrema in flow properties due to the under-resolved nature of the set-up. This might remind the reader of fluctuating hydrodynamics (FHD), where one retreats from the solely continuous view of the Navier-Stokes equations also used within our framework and includes the microscopic scales considering molecules. This comes in handy e.g. when very highly resolving fluid simulations or studying flows that have an extremely low density because the fluid's behaviour is then governed by the movement of individual particles. When looking for a method of solving a hydrodynamic problem which includes Brownian motion but avoids costly particle tracking one is bound to trip over FHD. Here stochastic fluxes model the thermal fluctuations just as our stochastic fluxes model turbulence. Therefore, the new ODT formulation will henceforth be called ODT-FHD. Interested readers may find an introduction to FHD in the original text [42], in [2] (chap. 5) or briefer in [9].

As we will consider realisations of stochastic variables and sets of events a lot hereafter, we need to introduce some symbols:

$\mathfrak{T}$: time of occurrence of eddy, $\mathfrak{L}$: length of eddy, $\mathfrak{X}$: left edge of eddy;
`A`: eddy is accepted, `I`: eddy gets implemented.

4.4.1 Energy Cascade and Length Scales

In Section 4.1 the energy cascade of turbulent flows was discussed and in Section 4.2 we briefly introduced how the original ODT formulation realises it. The key is the compression effect of the triplet maps on lateral profiles of the streamwise velocity which leads to higher strain with smaller length scales. Thus, large eddies are followed by smaller ones and so on. On the other hand, if length scales are small enough for viscosity to take over, gradients are generally smoothed out faster than eddies, feeding on the velocity differences, can form in that region. These, of course, are features that the ODT-FHD does not support, since the distortions within the lateral velocity profiles are not kept and eddies only act on the centreline. Nonetheless, we can look at the kinetic energy spectrum of eddies and find a cascade-like structure. The evaluation of $\langle u'(x)u'(x+r)\rangle$, $r \in (0, \mathcal{L} - x]$, the velocity correlation function commonly used to compute an energy spectrum, does not stand to reason owing to the under-resolved nature of ODT-FHD with $\ell_{max} \leq \Delta x$. However, we can make use of the interpretation of

$$\varrho\ell^3/\tau^2 = \varrho\ell\mathcal{A}^2(\Delta u)^2$$

as a measure of the kinetic energy of eddy motion ([41], [54]). In Figure 4.11 we see an example spectrum for the homogeneous flow setting. Let us compare this to

Figure 4.3, where we saw a qualitative representation of the spectrum in real turbulence. We observe that the ODT-FHD spectrum is mainly governed by a power law suddenly breaking at the lowest wavenumber which means the biggest eddies. Unlike the spectrum in Figure 4.3, the slope rises instead of falling. Furthermore, there is no fast energy drop at the Kolmogorov length scale. These discrepancies are due to the chosen lateral velocity profile and the centreline assumption, which entails that bigger eddies are always faster than smaller ones (illustrated by τ over ℓ in Figure 4.21). Also we find that in contrast to Kolmogorov's five-thirds law the exponent of the power-law part is rather -3, maximally. Hence, our eddies pass less energy down the length scales. The influence of the mean flow velocity is that of a scaling factor as an increase just shifts the graph up.

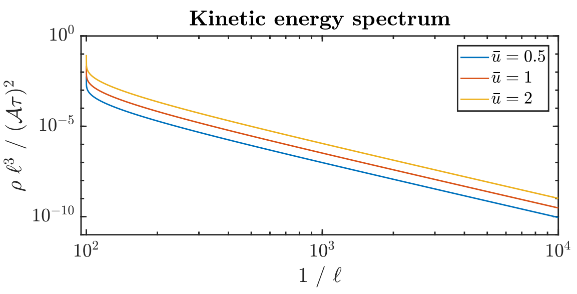

Figure 4.11: Example spectrum of kinetic energy of eddies for homogeneous pipe flow with different mean flow velocity.

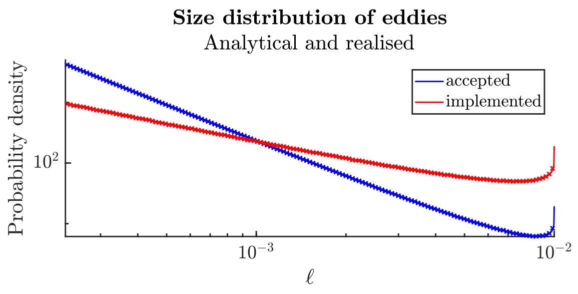

Figure 4.12: Normalised theoretical probability density function of accepted eddies $P_{\mathtt{A}}$ and of implemented eddies $P_{\mathtt{I}}$ depending on eddy size ℓ with histogram data points of numerically realised distribution functions from homogeneous pipe flow simulation.

Closely related to the energy spectrum is the distribution of length scales. So we will now regard the distribution of realised eddy sizes. This function is governed by the acceptance probability

$$\begin{aligned}
P(\mathtt{A} \cap \mathfrak{T} \in [t, t+dt] | \mathfrak{L} = \ell \cap \mathfrak{X} = x_0)& \\
&= \varphi_a(\ell, x_0; t, dt) \\
&= \frac{\lambda(\ell, x_0; t)dt}{\delta_\ell(\ell) \cdot \delta_{x_0}(x_0; \ell)} \\
&= \frac{\mathcal{C}}{\tau(\ell; x_0, t)\ell^2} \frac{dt}{\delta_\ell(\ell) \cdot \delta_{x_0}(x_0; \ell)} \\
&= \frac{\mathcal{AC}\Delta u(\ell, x_0; t)}{\ell^3} \frac{dt}{\delta_\ell(\ell) \cdot \delta_{x_0}(x_0; \ell)}
\end{aligned} \tag{4.23}$$

where dt is infinitely small. (4.23) is the probability of an eddy to be accepted when its time of occurrence is realised within $[t, t+dt]$ and under the condition that its length is ℓ and its left edge is x_0. By definition of conditional probability, the probability of such an eddy to realise and get accepted is

$$\begin{aligned}
P(\mathtt{A} \cap \mathfrak{T} \in [t, t+dt] \cap \mathfrak{L} = \ell \cap \mathfrak{X} = x_0)& \\
&= P(\mathtt{A} \cap \mathfrak{T} \in [t, t+dt] | \mathfrak{L} = \ell \cap \mathfrak{X} = x_0) \cdot \delta_\ell \cdot \delta_{x_0} \\
&= dt \frac{\mathcal{AC}\Delta u(\ell, x_0; t)}{\ell^3} \\
&=: P_{\mathtt{A}}.
\end{aligned} \tag{4.24}$$

With this equation at hand, we can also find the probability of an eddy to be implemented

$$\begin{aligned}
P(\mathtt{A} \cap \mathtt{I} \cap \mathfrak{T} \in [t, t+dt] \cap \mathfrak{L} = \ell \cap \mathfrak{X} = x_0)& \\
&= P_{\mathtt{A}} \cdot P(x_0 \in (x_{i+1/2} - \ell, x_{i+1/2}), i \in \{1, ..., N-1\}) \\
&= dt\mathcal{AC} \frac{\Delta u(\ell, x_0; t)}{\Delta x \ell^2} \\
&=: P_{\mathtt{I}}.
\end{aligned} \tag{4.25}$$

Normalising $P_{\mathtt{A}}$ and $P_{\mathtt{I}}$ yields the PDFs for accepted and implemented eddies as can be seen in Figure 4.12. The data points are produced by considering all eddies within a small interval $[\ell, \ell + d\ell]$ for any time t and position x_0 over all 150 realisations. This only makes sense because $\Delta u(\ell, x_0; t)$ - and therefore $P_{\mathtt{A}}$ and $P_{\mathtt{I}}$ - is actually independent of x_0 and t in the framework of our homogeneous pipe flow simulation and can thus be integrated over space and time easily. Figure 4.12 shows how well the histograms of numerical realisations match the theoretic distributions and that, in general, smaller eddies outnumber bigger ones as is expected. Apart from that, we recognise that the eddy sizes do stretch over some length scales but also that there is a sudden cut-off of small eddies at about 2×10^{-4} well above the Kolmogorov length scale - in this case $\eta = \ell_{max}/Re^{3/4} \approx 1.6 \times 10^{-6}$. That happens because of the small eddy suppression (4.12) which can be influenced by the choice of $\mathcal{A}$ as we will discuss in Subsection 4.4.3. Further, we perceive the distinct favouring of larger eddies by the implementation condition and we can also spot a slight uptrend for the biggest eddies in both functions. This, too, is a consequence of velocity profile (4.17) and the centreline assumption.

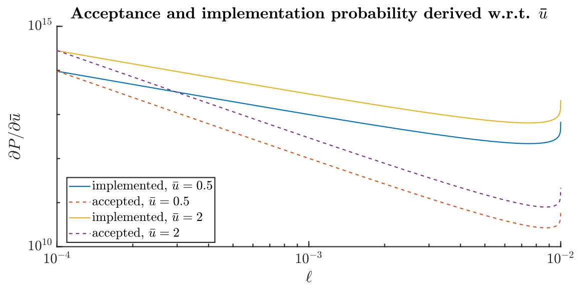

Figure 4.13: $\partial P_{\mathtt{A}}/\partial \bar{u}$ and $\partial P_{\mathtt{I}}/\partial \bar{u}$ over ℓ for two different values of mean flow velocity $\bar{u}$.

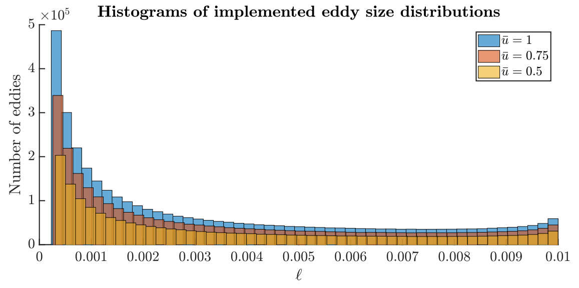

Figure 4.14: Number of implemented eddies over ℓ for three different values of $\bar{u}$.

Staying with the length distributions, we turn towards their dependence on the turbulence generating quantity. From Section 4.1 we know that with increasing mean flow velocity and hence Reynolds number we should see more turbulence and a wider range of length scales, i.e., η decreases. To verify that our ODT-FHD features this behaviour, we will derive $P_{\mathtt{A}}$ and $P_{\mathtt{I}}$ with respect to $\bar{u}$. For our purpose it suffices to assume, that $\bar{u} > 0$. We obtain

$$\begin{aligned}\frac{\partial P_{\mathtt{A}}}{\partial \bar{u}} =& \frac{dt}{\ell^3}\frac{\partial \Delta u(\ell, x_0; t)}{\partial \bar{u}} \\ =& \frac{dt}{\ell^3}\left\{\left(\frac{2\psi^2+3\psi+1}{2\psi^2} - \frac{3.15\psi+2.1}{\ln(10)\psi^3}\right)\left[1-\left(1-\frac{\ell}{2R}\right)^{1/\psi}\right]\right. \\ &\left. + u_{max}(x_0,t)\,\psi^{-2}\left(1-\frac{\ell}{2R}\right)^{1/\psi}\ln\left(1-\frac{\ell}{2R}\right)\cdot\frac{2.1}{\bar{u}(x_0,t)\ln(10)}\right\},\end{aligned} \tag{4.26}$$

$$\frac{\partial P_{\mathtt{I}}}{\partial \bar{u}} = \frac{\partial P_{\mathtt{A}}}{\partial \bar{u}} \frac{\ell}{\Delta x}.$$

Appendix A.1 provides additional computation details. In Figure 4.13 we see these two functions over ℓ for two different values of $\bar{u}$. It shows that increasing the mean flow velocity emphasises smaller eddies rather than bigger ones - except for the biggest as in all our considerations. We can also see that the difference is smaller for implemented than for the accepted eddies due to the implementation condition. Figure 4.14 shows the overall larger number of eddies for faster flows as well as a shift to the left of the smallest length implemented. Therefore, we are content with this feature.

4.4.2 Fluxes and Turbulent Diffusion

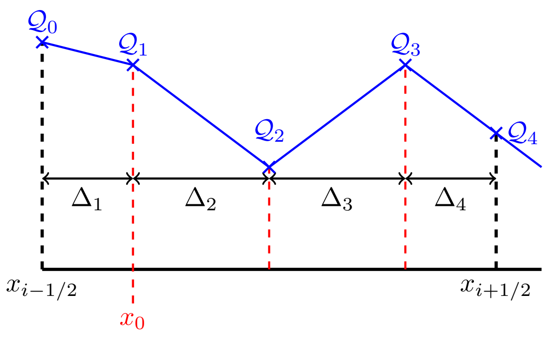

Figure 4.15: Points and distances for cell integration after eddy event (case 3).

As stated in Section 4.1 what engineers usually are most interested in when simulating a turbulent flow are the fluctuations and turbulent diffusion. Hence, these are our next objects of study. First of all, it needs to be clarified that new maxima and minima of flow properties arising along the x-axis due to eddies shuffling the fluid, will not occur using ODT-FHD in its current form. We will see later why this is impossible but for now we turn to what will be attained: turbulent diffusion. Surely, for our under-resolved grid, we cannot simply track a marked particle to evaluate turbulent diffusion. But what we can study the expected value of the flux (4.18) between certain grid cells $i \in \{1, ..., N-1\}$ and $i+1$ due to eddy events with respect to the eddy's length ℓ. To this aim, we consider the expected value of

$$\beta_T \tau \cdot \mathcal{E}^k_{i+1/2} = \int\limits_{x_{i-1/2}}^{x_{i+1/2}} q(\zeta, t_k + \beta_T \tau)\, d\zeta - \int\limits_{x_{i-1/2}}^{x_{i+1/2}} q(\zeta, t_k)\, d\zeta$$

under the condition that $\mathfrak{L} = \ell$. In order to do this, we regard the integration procedure implemented in the SEC-code, using

$$\int_{x_{i-1/2}}^{x_{i+1/2}} q(\zeta, t_k + \beta_T \tau)\, d\zeta = \sum_{j=0}^{3} \frac{\Delta_{j+1}}{2} \left(\mathcal{Q}_j + \mathcal{Q}_{j+1}\right)$$

with $\mathcal{Q}_j$ and Δ_j as depicted in Figure 4.15. Depending on the position of the eddy, we must distinguish three different cases, which are defined as

case 1: $0 < x_{i+1/2} - x_0 \leq \ell/3$,
case 2: $\ell/3 < x_{i+1/2} - x_0 \leq 2\ell/3$ and
case 3: $2\ell/3 < x_{i+1/2} - x_0 \leq \ell$.

If $x_{i+1/2} - x_0 > \ell$ the eddy does not stretch over the interface of two grid cells and will therefore not be implemented. Using the shorthands

$$\begin{aligned} S :=& \frac{Q_{i+1}^k - Q_i^k}{\Delta x}, \\ \Delta\zeta_r :=& x_{i+1/2} - x_0, \\ \Delta\zeta_l :=& x_0 - x_{i-1/2} = \Delta x - \Delta\zeta_r, \end{aligned}$$

the according intermediate values $\mathcal{Q}_j$ and step widths Δ_j are computed as follows

	case 1	**case 2**	**case 3**
$\mathcal{Q}_0$	$Q_i^k - \frac{S}{2}\Delta x$	$Q_i^k - \frac{S}{2}\Delta x$	$Q_i^k - \frac{S}{2}\Delta x$
$\mathcal{Q}_1$	$\mathcal{Q}_0 + S\Delta\zeta_l$	$\mathcal{Q}_0 + S\Delta\zeta_l$	$\mathcal{Q}_0 + S\Delta\zeta_l$
Δ_1	$\Delta\zeta_l$	$\Delta\zeta_l$	$\Delta\zeta_l$
$\mathcal{Q}_2$	$\mathcal{Q}_1 + 3S\Delta\zeta_r$	$\mathcal{Q}_1 + 3S\frac{\ell}{3}$	$\mathcal{Q}_1 + 3S\frac{\ell}{3}$
Δ_2	$\Delta\zeta_r$	$\frac{\ell}{3}$	$\frac{\ell}{3}$
$\mathcal{Q}_3$	0	$\mathcal{Q}_2 - 3S\left(\Delta\zeta_r - \frac{\ell}{3}\right)$	$\mathcal{Q}_2 - 3S\frac{\ell}{3}$
Δ_3	0	$\Delta\zeta_r - \frac{\ell}{3}$	$\frac{\ell}{3}$
$\mathcal{Q}_4$	0	0	$\mathcal{Q}_3 + 3S\left(\Delta\zeta_r - \frac{2\ell}{3}\right)$
Δ_4	0	0	$\Delta\zeta_r - \frac{2\ell}{3}$

Now we can turn to expressing the expected value

$$E\left(\int_{x_{i-1/2}}^{x_{i+1/2}} q(\zeta, t_k + \beta_T\tau)\, d\zeta - \int_{x_{i-1/2}}^{x_{i+1/2}} q(\zeta, t_k)\, d\zeta\right)$$

in terms of ℓ by regarding ℓ as fixed and x_0 as a uniformly distributed random variable. As the computations follow the same rules for all three cases, only the first one will be shown here. Due to extra terms, the second and third case's formulae are much less convenient to read. Dedicated readers are referred to Appendix A.2 where the according calculations are carried out, nonetheless.

As a first step we will rearrange $\int_{x_{i-1/2}}^{x_{i+1/2}} q(\zeta, t_k + \beta_T\tau)\, d\zeta$ using

$$\Delta_1 + \Delta_2 = \Delta x$$

and

$$\mathcal{Q}_2 = \mathcal{Q}_1 + 3S\Delta\zeta_r = \mathcal{Q}_0 + S\Delta\zeta_l + 3S\Delta\zeta_r.$$

Thus for case 1 with $0 < \Delta\zeta_r \leq \ell/3$ it holds

$$\begin{aligned}
\int_{x_{i-1/2}}^{x_{i+1/2}} q(\zeta, t_k + \beta_T\tau)\, d\zeta =& \sum_{j=0}^{3} \frac{\Delta_{j+1}}{2}\left(\mathcal{Q}_j + \mathcal{Q}_{j+1}\right) \\
=& \frac{\Delta_1}{2}\left(\mathcal{Q}_0 + \mathcal{Q}_1\right) + \frac{\Delta_2}{2}\left(\mathcal{Q}_1 + \mathcal{Q}_2\right) \\
=& \frac{1}{2}\left[\Delta_1\mathcal{Q}_0 + \Delta x\mathcal{Q}_1 + \Delta_2\mathcal{Q}_2\right] \\
=& \frac{1}{2}\left[\Delta_1\mathcal{Q}_0 + \Delta x(\mathcal{Q}_0 + S\Delta\zeta_l) + \Delta_2(\mathcal{Q}_0 + S\Delta\zeta_l + 3S\Delta\zeta_r)\right] \\
=& \Delta x\mathcal{Q}_0 + \frac{S}{2}\left(\Delta x\Delta\zeta_l + \Delta\zeta_r\Delta\zeta_l + 3\Delta\zeta_r^2\right).
\end{aligned}$$

In the next step, we subtract $\Delta x Q_i^k$ and substitute $\Delta\zeta_l$ with $\Delta x - \Delta\zeta_r$, yielding

$$\begin{aligned}
\beta_T\tau \cdot \mathcal{E}_{i+1/2}^k =& -\frac{S}{2}\Delta x^2 + \frac{S}{2}\left(\Delta x^2 - \Delta\zeta_r\Delta x + \Delta x\Delta\zeta_r - \Delta\zeta_r^2 + 3\Delta\zeta_r^2\right) \\
=& S\Delta\zeta_r^2.
\end{aligned}$$

Since $\Delta\zeta_r^2$ depends on the random variable x_0 the next step is to compute the expected value of this variable. Note that it also depends on the case considered, as this restricts the possible values of $\Delta\zeta_r$ and hence we are computing conditional expectation values. The index ℓ shall remind us, that within this considerations we impose the condition $\mathfrak{L} = \ell$ and thus ℓ is fixed instead of random. The number in the index indicates the case regarded

$$E_{\ell,j}(X) = E\left(X \middle| \mathfrak{L} = \ell \cap \Delta\zeta_r \in \left((j-1)\frac{\ell}{3}, j\frac{\ell}{3}\right]\right).$$

Using this shorthand we compute

$$E_{\ell,1}(\Delta\zeta_r^2) = \int_0^{\ell/3} \zeta^2 \cdot \frac{1}{\ell/3}\, d\zeta = \frac{3}{\ell}\left[\frac{1}{3}\zeta^3\right]_0^{\ell/3} = \frac{\ell^2}{27}.$$

Exploiting the linearity of expectation and inserting the above value, we get the expected value of the eddy flux over the turnover time for case 1

$$E_{\ell,1}\left(\beta_T\tau \cdot \mathcal{E}_{i+1/2}^k\right) = \frac{S}{27}\ell^2.$$

Following this chain of arguments and computations for the other two cases, we arrive at

$$E_{\ell,2}\left(\beta_T\tau \cdot \mathcal{E}_{i+1/2}^k\right) = \frac{4}{27}S\ell^2,$$

$$E_{\ell,3}\left(\beta_T\tau \cdot \mathcal{E}_{i+1/2}^k\right) = \frac{S}{27}\ell^2.$$

Since x_0 is uniformly distributed, the three different cases have the same likelihood of $\frac{1}{3}$. Hence, utilising the law of total expectation, we find the expected value for $\beta_T\tau \cdot \mathcal{E}_{i+1/2}^k$ disregarding the case, to be

$$E_\ell\left(\beta_T\tau \cdot \mathcal{E}_{i+1/2}^k\right) = \frac{1}{3}\sum_{j=1}^{3} E_{\ell,j}\left(\beta_T\tau \cdot \mathcal{E}_{i+1/2}^k\right) = \frac{2}{27}S\ell^2.$$

So we see that the expected value of the flux depends on the square of the eddy length which was also stated by Kerstein for his version of ODT ([39]). Moreover, we find, that the flux is greatest for case 2. In fact, taking a step back and knowing that $\Delta\zeta_r = c\ell$ for $c \in (0,1)$, we can infer

$$\beta_T\tau \cdot \mathcal{E}_{i+1/2}^k = \begin{cases} S\ell^2c^2 & c \in \left(0, \frac{1}{3}\right], \\ S\ell^2\left(-2c^2 + 2c - \frac{1}{3}\right) & c \in \left(\frac{1}{3}, \frac{2}{3}\right], \\ S\ell^2\left(c^2 - 2c + 1\right) & c \in \left(\frac{2}{3}, 1\right). \end{cases} \tag{4.27}$$

It was stated before that sudden increases of gradients due to eddy events will never arise when using ODT-FHD in its current implementation. Now we can see why: according to (4.27) any eddy induced flux will always decrease S mimicking the effect of turbulent diffusion only. Figure 4.16 shows the flux distribution from (4.27) for $S\ell^2 = 1$ in direct comparison to a fitted and scaled normal distribution $a\exp(-(\frac{x-b}{d})^2)$. Interestingly, we see quite a close match. If S or ℓ^2 change, only the scaling factor a would need an according update.

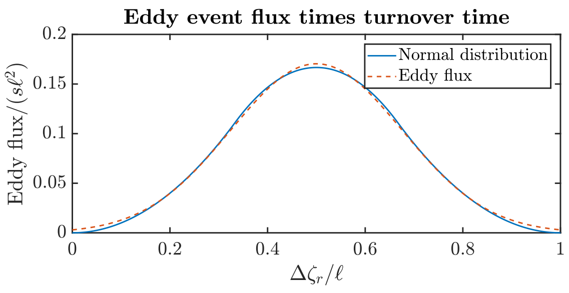

Figure 4.16: Scaled $\beta_T\tau \cdot \mathcal{E}^k_{i+1/2}$ for all three cases of $\Delta\zeta_r$ compared to a fitted and scaled normal distribution.

To calculate the unconditional expected value of $\beta_T\tau \cdot \mathcal{E}^k_{i+1/2}$ we need P_{I}, the probability for an eddy to be implemented, which we already calculated in (4.25). Again we assume $\bar{u} > 0$ yielding

$$E\left(\beta_T\tau \cdot \mathcal{E}^k_{i+1/2}\right)$$

$$= \int_{\ell_{min}}^{\ell_{max}} E_\ell\left(\beta_T\tau \cdot \mathcal{E}^k_{i+1/2}\right) P_{\mathrm{I}}\, d\ell$$

$$= dt \frac{2S}{27}\frac{\mathcal{AC}}{\Delta x} \int_{\ell_{min}}^{\ell_{max}} \Delta u(\ell, x_0; t) d\ell$$

$$= dt \frac{2S}{27}\frac{\mathcal{AC}}{\Delta x} u_{max}(x_0,t)(\ell_{max}-\ell_{min}) \left[1 - \frac{\psi}{\psi+1}(\ell_{max}-\ell_{min})\left(1-\frac{\ell_{min}}{\ell_{max}}\right)^{1/\psi}\right]$$

The last line results from integrating

$$\int_{\ell_{min}}^{\ell_{max}} \Delta u(\ell, x_0; t) d\ell$$

$$= \int_{\ell_{min}}^{\ell_{max}} u_{max}(x_0,t)\, d\ell - \int_{\ell_{min}}^{\ell_{max}} u_{max}(x_0,t)\left(1-\frac{\ell}{2R}\right)^{1/\psi} d\ell$$

$$=u_{max}(x_0,t)\left((\ell_{max}-\ell_{min})-\left[\frac{\psi}{\psi+1}(\ell-2R)\left(1-\frac{\ell}{2R}\right)^{1/\psi}\right]_{\ell_{min}}^{\ell_{max}}\right)$$

and using $2R = \ell_{max}$. As in general ℓ_{max} and Δx are constants for any set-up and $\ell_{min} \ll \ell_{max}$, the eddy flux is mainly governed by the mean flow velocity and the model parameters $\mathcal{A}$ and $\mathcal{C}$ as well as the solution gradients within the flow which makes sense for a turbulence model.

Actually, we were a bit lax with the term “unconditional” in the above reflections. When considering $E\left(\beta_T\tau \cdot \mathcal{E}^k_{i+1/2}\right)$ we assumed that an eddy was implemented crossing the interface $x_{i+1/2}$ at $t_k + \beta_T\tau$. If we were to compute the really unconditional expected value of eddy flux within a certain time interval at a specific interface, we would need to incorporate the eddy occurrence distribution which heavily depends on $\Delta u(\ell, x_0; t)$. A true calculation of the expected value of Q_i at any time level with possibly multiple eddy occurrences would even include a binary tree of the cell interface the eddy crosses. The space and time dependence of $\Delta u(\ell, x_0; t)$ would impose a greater complexity to say nothing of the other fluid dynamical processes taking place in-between implemented eddies. Finally, we arrive at the reason why all this is carried out within a numerical simulation instead of analytically solving equations. Hence, we content ourselves using the above formulae to have a look at the influence of the model parameter $\mathcal{A}$, β_T and $\mathcal{C}$ on the possible outcomes of the simulations in the next Subsection.

4.4.3 Influence of the Model Parameter

If we want to use ODT for a simulation, the first task is always to find the necessary number of realisations to average over as well as fitting values for the model parameters. In the current implementation these are $\mathcal{A}$, β_T and $\mathcal{C}$. To test the effect of these parameters the homogeneous pipe flow summarised in Table 4.1 is used again here. All flow properties shown are ensemble averages.

Number of Realisations

In Section 4.1 we have seen, that the ensemble mean of a quantity which fluctuates due to turbulence, e.g. $\langle u(x,t)\rangle$, is convergent with a growing number of realisations. Ergo, this could be a good way to find the needed number of realisations. For the implementation and test case at hand, a very small number of only 20 realisation would already meet the convergence condition but as we also want to have a close look on the statistics of eddies 150 realisations were simulated. This ensured also the statistical convergence of distributions.

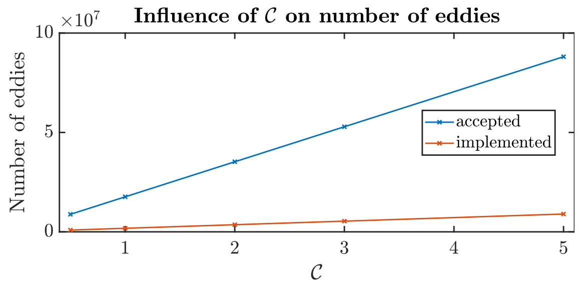

Figure 4.17: Dependency of numbers of accepted and implemented eddies on $\mathcal{C}$.

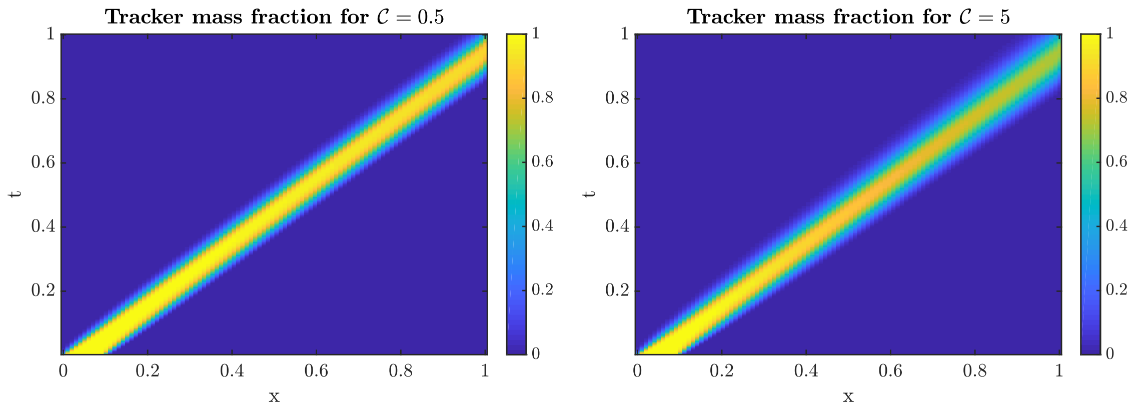

Figure 4.18: Tracker mass fraction over space and time for $\mathcal{C} = 0.5$ (left) and $\mathcal{C} = 5$ (right).

Parameter $\mathcal{C}$

Recall that the model parameter $\mathcal{C}$ is a proportionality factor influencing the distribution rate of eddies

$$\lambda(\ell, x_0; t) = \mathcal{C} \cdot \frac{1}{\tau(\ell, x_0; t)\ell^2}.$$

Hence, we would expect to see the number of trials and therefore accepted and implemented eddies depending linearly on $\mathcal{C}$ which is the case as can be concluded from Figure 4.17. All other eddy properties like distribution of sizes, turnover times or fluxes are left untouched by changing $\mathcal{C}$ because the eddy turnover time scale τ is independent of $\mathcal{C}$.

Implementing more eddies should result in stronger turbulence. Comparing the x-t-map of the tracker species mass fraction for two different cases of $\mathcal{C}$ in Figure 4.18 verifies this expectation.

Parameter β_T

The model parameter β_T is part of the large eddy suppression strategy, that requests $t > \beta_T \tau$. Therefore, it should govern the onset of larger eddies in the simulation. In Figure 4.19 we recognise the lower numbers of eddies at the beginning of the simulation for the higher β_T value which is a desired effect. On the other hand,

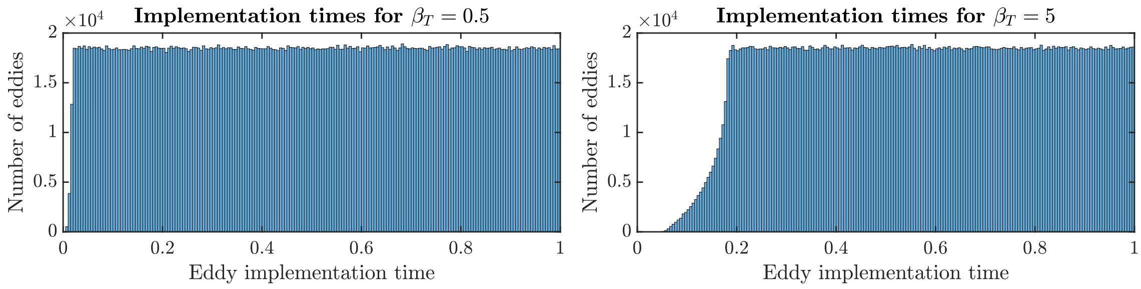

Figure 4.19: Histogram of implementation times of eddies for $\beta_T = 0.5$ (left) and $\beta_T = 5$ (right).

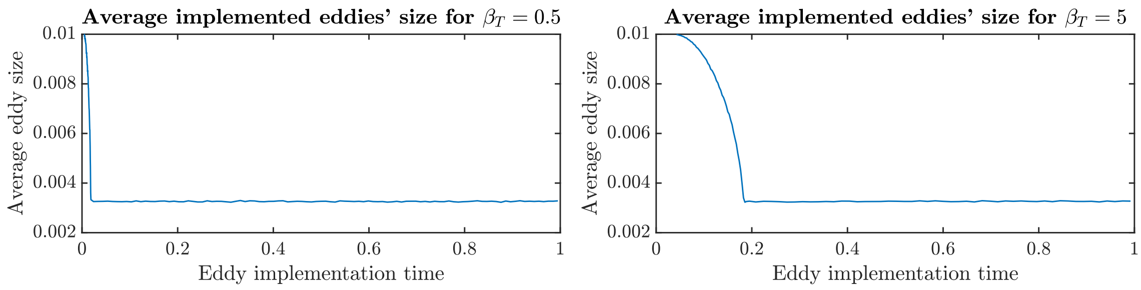

Figure 4.20: Average size of implemented eddies over implementation time for $\beta_T = 0.5$ (left) and $\beta_T = 5$ (right).

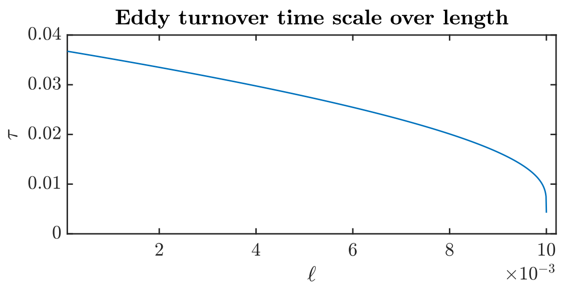

Figure 4.21: Representative example of eddy turnover time scale τ over ℓ for $\mathcal{A} = 2$ and $\bar{u} = 1$.

studying the average length of implemented eddies as shown in Figure 4.20 we find that the eddies rejected due to the large eddy suppression must be the smaller ones. To explain this odd outcome we consult Figure 4.21 where the eddy time scale τ is depicted as a function of eddy size ℓ. As τ is monotonically decreasing, the biggest eddies are also the fastest. This is a result of the centreline assumption and directly leads to the phenomenon seen in Figure 4.20. Consequentially, we should rather speak of a “slow eddy suppression”. As small eddies have a very limited impact on the eddy flux, the effect of different β_T is as minor. So Figure 4.22 features an especially great difference of β_T to demonstrate the weaker onset of turbulent diffusion.

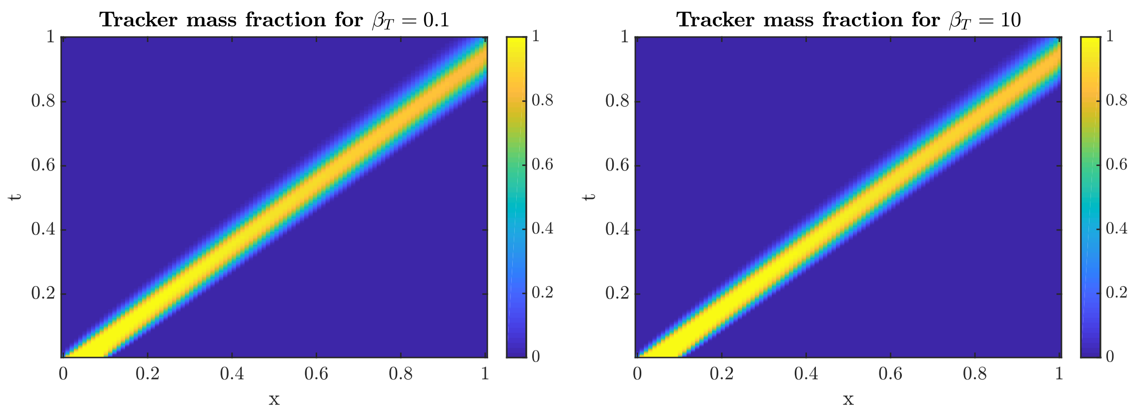

Figure 4.22: Tracker mass fraction over space and time for $\beta_T = 0.1$ (left) and $\beta_T = 10$ (right).

Parameter $\mathcal{A}$

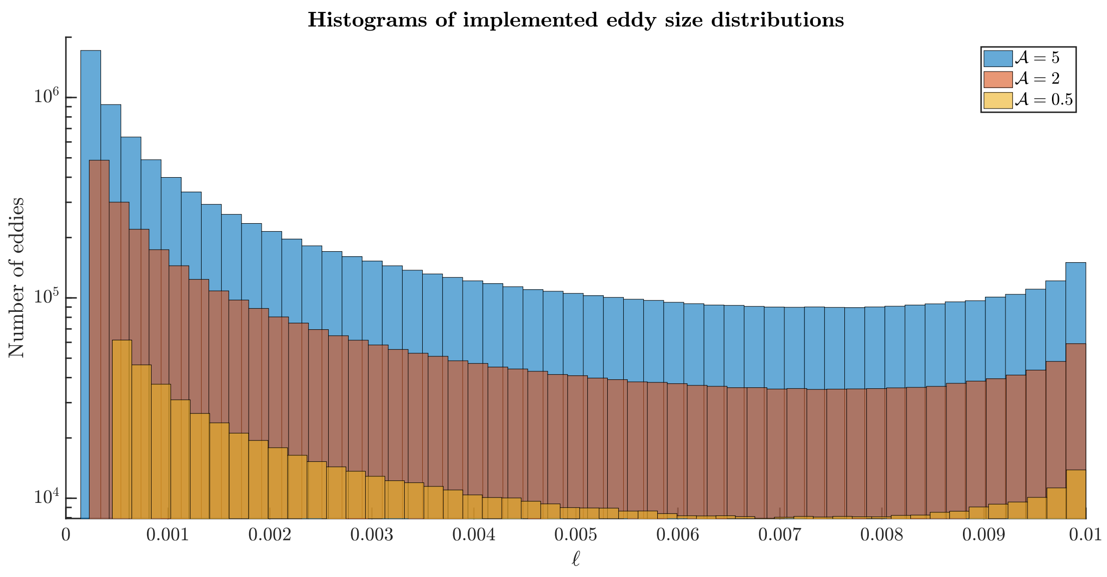

Figure 4.23: Eddy size distributions for $\mathcal{A} = 0.5$, $\mathcal{A} = 2$ and $\mathcal{A} = 5$.

The model parameter $\mathcal{A}$ is present in the calculation of the eddy time scale

$$\tau(\ell, x_0; t) = \frac{\ell}{\mathcal{A}\Delta u(\ell, x_0; t)}$$

and hence in the eddy distribution rate

$$\lambda(\ell; x_0, t) = \frac{\mathcal{C}}{\ell^2 \tau(\ell; x_0, t)} = \mathcal{A}\mathcal{C}\frac{\Delta u(\ell, x_0; t)}{\ell^3}$$

and acceptance probability (4.24).

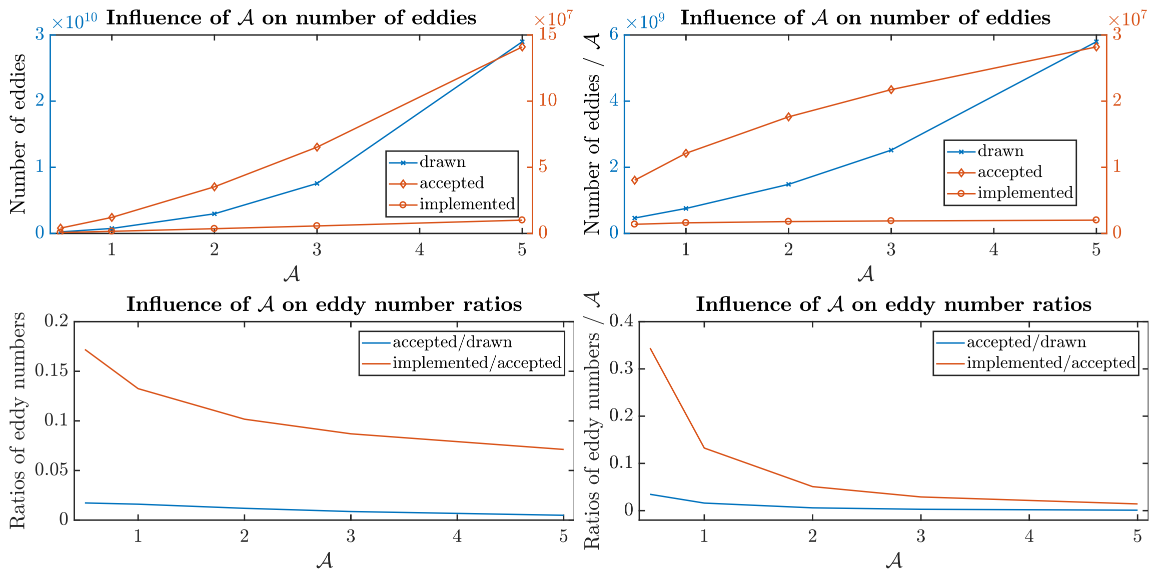

Figure 4.24: Dependency of drawn, accepted and implemented eddies on $\mathcal{A}$ in absolute numbers (top left) and relative numbers with respect to $\mathcal{A}$ (top right) as well as ratios of accepted to drawn and implemented to accepted eddies, respectively, in absolute numbers (bottom left) and relative numbers with respect to $\mathcal{A}$ (bottom right).

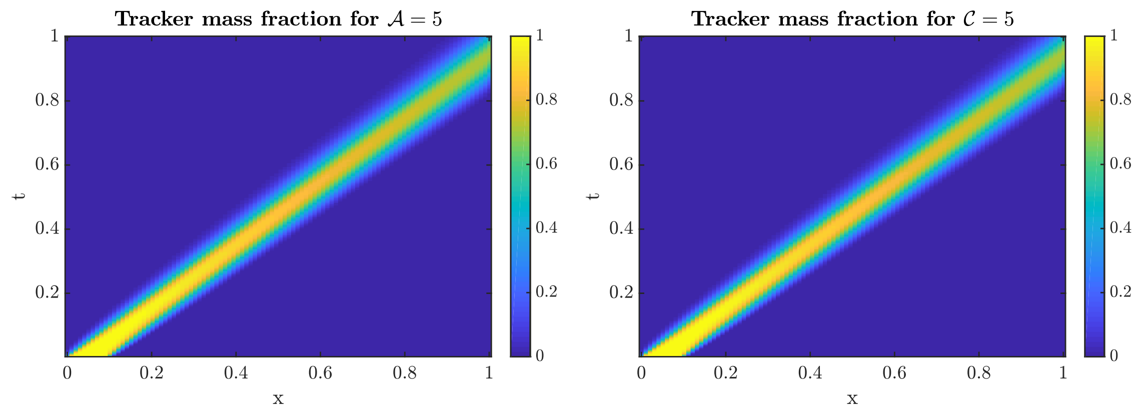

Figure 4.25: Comparison of influence of $\mathcal{A}$ (left) and $\mathcal{C}$ (right) on tracker mass fraction.

So from this point of view, the influence of $\mathcal{A}$ on the numbers of drawn, accepted and implemented eddies and their length distribution is the same as the one of $\mathcal{C}$: it is linear, multiplying to the absolute numbers but leaving the distributions unchanged. Nonetheless, this is not what we see in histograms as in Figure 4.23 where we spot that there is a shift towards smaller eddies in the length distribution. In Figure 4.24 it is shown that although the absolute numbers of drawn, accepted and implemented eddies in fact rise superlinearily, the ratios of accepted to drawn and implemented to accepted eddies, respectively, fall sublinearily. This is due to the small eddy suppression mechanism implemented in the manner of [39] which cuts off eddies which are slower - and thus smaller - than a certain threshold. As this requirement is formulated in terms of τ, it is influenced by $\mathcal{A}$

$$\tau(\ell, x_0; t) = \frac{\ell}{\mathcal{A}\Delta u(\ell, x_0; t)} < \tau_d = \frac{\varrho(x_0; t)\ell^2}{16\mu(x_0; t)}$$
$$\Rightarrow \mathcal{A} = \frac{16\mu(x_0; t)}{\ell_{inf}\varrho(x_0; t)\Delta u(\ell_{inf}, x_0; t)}. \tag{4.28}$$

Here ℓ_{inf} is the infimum of the length of accepted eddies. Using (4.28) and a good guess for the maximum Reynolds number of the flow to be simulated we could choose $\mathcal{A}$ such that $\ell_{inf} \sim \eta$, the Kolmogorov length scale. Albeit, we know from Section 4.4 that the impact of such small eddies on the outcome is minor.

Still, $\mathcal{A}$'s influence reaches even deeper: Because it modulates τ, it also changes the threshold for large eddy suppression, countering the effect of β_T as

$$t \geq \frac{\beta_T}{\mathcal{A}} \frac{\ell}{\Delta u(\ell, x_0; t)}.$$

These more subtle influences can only be seen in the statistics of ODT while the ensemble averaged flow properties seem to care less (see Figure 4.25) - at least for our simple homogeneous test case where the model parameter ranges are small.

4.4.4 Turbulent Flame

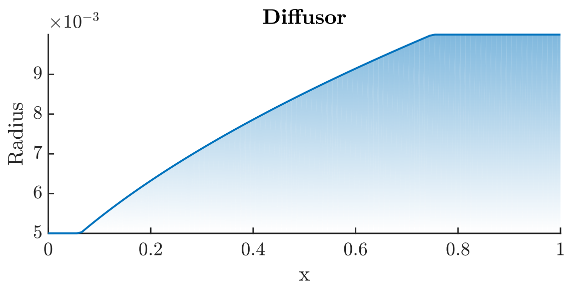

Figure 4.26: Radius of combustion chamber with diffusor configuration.

As explained in Section 1.2 ODT was included in the SEC-code to enable the simulation of the starting process which requires a turbulent deflagration. Accordingly, we need to test whether the ODT-FHD is capable of creating such a flame. After some trials a configuration was come across which met this requirement. Table 4.2 summarises the settings. As a frame of reference a laminar diffusion flame was simulated using a thickened flame ansatz. Roughly speaking this spreads the flame front over several grid cells by using increased transport coefficients while keeping the flame speed. This is done to be able to resolve the otherwise very thin reaction front with less numerical effort. The reader is referred to [12] for an insight to this method.

governing equations	quasi one-dimensional Navier-Stokes (thickened flame); quasi one-dimensional Euler with ODT-FHD
chemistry	2-species Arrhenius kinetics
domain	length: 1, area variation: diffusor
initial values	$p = 1.9$, $T \approx 11.2$, $u = 0$, product species everywhere
boundaries	left: fuelling valve, right: isentropic expansion to $p = 1$
ODT setting	20 realisations, $\mathcal{A} = 1$, $\mathcal{C} = 200$, $\beta_T = 1$, $\ell_{min} = 10^{-5}, \ell_{max} = 0.01$
grid cells	100
time steps	step size chosen automatically, snapshots stored at multiples of 4×10^{-3}

Table 4.2: Settings for turbulent/laminar flame simulations.

Like designated for deflagration flames, the 2-species Arrhenius kinetics were used. For the ODT simulation molecular transport was disabled since we wanted to see the "pure" effects of our turbulence model. The computational domain is a diffusor tube of length one as depicted in Figure 4.26. The simulations were initialised with hot exhaust (non-reactive) gas at $p = 1.9$, $T \approx 11.2$ and $u = 0$ everywhere. The downstream boundary was set as an isentropic expansion to a fixed outer pressure $p = 1$ while the upstream boundary is a fuelling valve. It expands slightly heated and compressed gas at $p = 2$, $T \approx 1.2$, consisting only of fuel species, isentropically into the combustion chamber. In accordance with the conditions of ODT-FHD simulations, the domain was resolved with 100 grid cells, achieving $\Delta x = \ell_{max} = 2\min(R)$. We still had to find fitting model parameters but since this is supposed to be a simple proof of concept we merely adjusted $\mathcal{C}$ to yield a turbulent diffusion high enough to create a stabilising flame.

In Figure 4.27 we see the results of both simulations in direct comparison. First of all, we need to notice that the turbulent flame eventually stabilises as expected. Apart from that there is a big blast at the beginning of the turbulent simulation which catches the eye. The color map range was clipped such that enough details remained in the lower sections. The actual maximum velocity, temperature and pressure are higher. This sudden outburst is a result of the initial conditions - quiescent hot exhaust gas in the combustion chamber - meeting the left boundary condition - a valve injecting cooler, fuelled gas to the right. In the laminar simulation, the interface of this explosive contrast was smeared fast enough whereas in the ODT-FHD simulation the turbulence did not yet set in due to a lack of substantial lateral velocity differences. In both simulations we perceive an expansion front travelling upstream. It originates from the right boundary condition. When the pressure wave reaches the inlet fuel is sucked deeper into the combustion chamber for both configurations. That is when the flame front gets softer in the turbulent simulation because the velocity difference finally suffices to diffuse the flow properties enough. Lastly, both flames stabilise further downstream.

This comparison elucidates the essential difference between turbulent diffusion from ODT-FHD and molecular diffusion from the Navier-Stokes equations. Even if the diffusion effects are of comparable size turbulence still depends on the mean flow velocity while molecular transport changes with pressure, temperature and mixture of species.

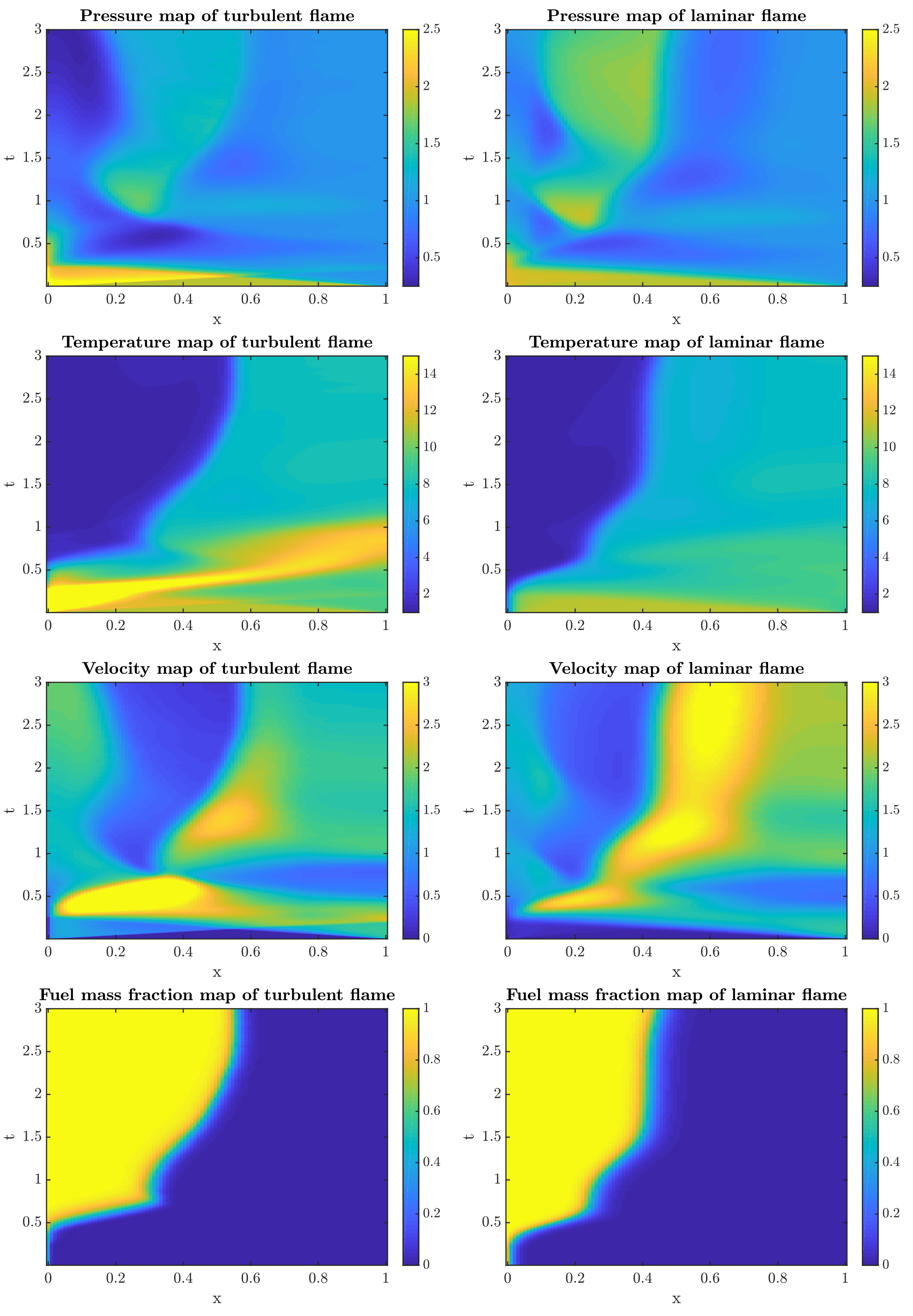

Figure 4.27: Maps of the flow properties pressure, temperature, velocity and fuel mass fraction (top to bottom) over space and time for turbulent (left) and laminar (right) flame simulation.

4.4.5 Turbulent SEC

governing equations	quasi one-dimensional Euler (with ODT-FHD)
chemistry	3-species ignition delay kinetics
domain	length: 1, area variation: none
initial values	stabilised cyclic SEC at ignition
boundaries	left: fuelling pressure valve, right: isentropic expansion to $p = 1$
ODT setting	20 realisations, $\mathcal{A} = 1$, $\mathcal{C} = 200$, $\beta_T = 0$, $\ell_{min} = 10^{-5}$, $\ell_{max} = 0.01$
grid cells	100
time steps	step size chosen automatically, snapshots stored at multiples of 0.022

Table 4.3: Settings for turbulent/laminar SEC simulations.

Last but not least, we are interested in the effect of turbulent diffusion on the SEC. A complete study of this scenario goes far beyond the scope of this thesis. So for simplicity the model parameter for the ODT-FHD simulation were chosen like in the simulation of the turbulent flame except for β_T. As the starting process with turbulent deflagration is also left for future research, the simulations were initialised at the beginning of an already cyclic (laminar) SEC right before ignition. That is why β_T was set to 0 although this somewhat collides with the notion of the eddy turnover time being $\beta_T \tau$. The left boundary condition models a pressure valve like in Subsection 3.3.2, opening at underpressure fuelling the combustion chamber with a profile well-tuned for a laminar single-tube SEC. The right boundary creates the necessary open end for the pressure wave to be reflected as explained in Chapter 1. Table 4.3 summarises this set-up. This time a laminar SEC without molecular transport - as used in the first phase of the CRC 1029 - was simulated for reference. In Figure 4.28 this comparison shows that, although the stark turbulence drastically changes the fuelling profile and the pressure waves, the SEC is able to stabilise. Most interestingly, the periodicity establishes with two fuelling cycles: a bigger and a smaller one. This seems to be the result of the first injected fuel package igniting prematurely due to higher temperatures because the cooler air buffer gets diffused heavily. When the suction wave from the downstream end of the SEC-tube returns a second fuel package gets injected. As the temperature peak from the preceding ignition was lower and smaller this loading process has more time before the fuel ignites. After that the two-ignitions-cycle repeats.

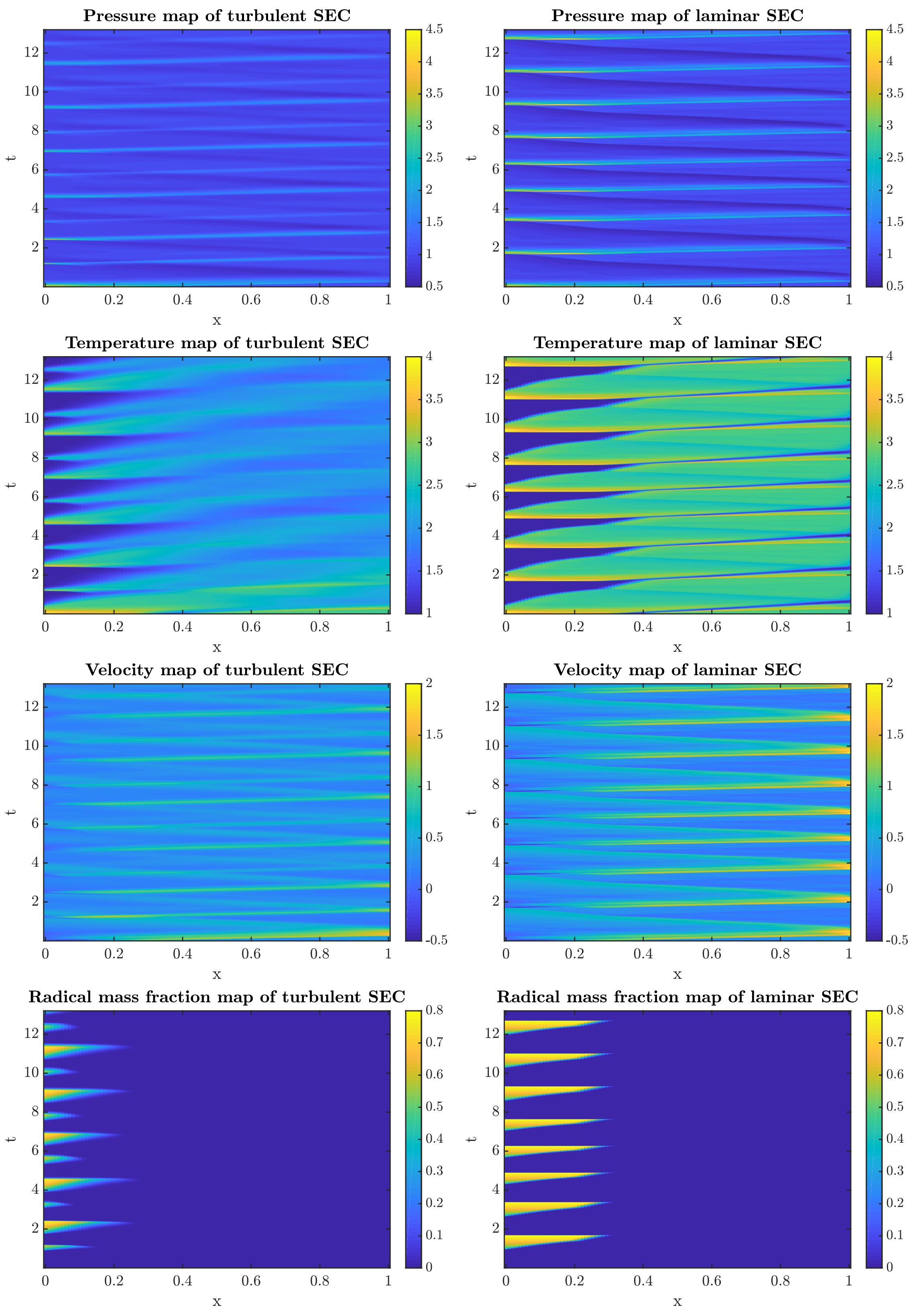

Figure 4.28: Maps of the flow properties pressure, temperature, velocity and radical mass fraction (top to bottom) over space and time for turbulent (left) and laminar (right) SEC simulation.

4.5 Conclusions

In this section, we have studied a new formulation of Kerstein's ODT: the ODT-FHD. It was shown that with this approach it is possible to carry out quasi one-dimensional turbulent pipe flow simulations with the ODT-line aligned with the streamwise direction x. Although this is a completely new way of using ODT this formulation still catches some of the most important features of the original ODT as turbulent length scale distribution, the energy cascade and the squared influence of eddy size on turbulent fluxes. It also shows the same dependence on the mean flow and is capable of mimicking turbulent diffusion. Thus, it will be a useful tool for studying the starting process of the SEC with a turbulent deflagration as desired.

We have also seen the influence of the model parameters and how much control they give users over the behaviour of the turbulence model. Whilst $\mathcal{A}$ is the last parameter studied here, it should always be the first one chosen taking the length distribution and viscous cut-off as well as ratios of accepted and implemented eddies for computational efficiency into account. $\mathcal{C}$ and β_T are independent of each other and can be chosen to correct the number of eddies drawn and onset of turbulence. In the current implementation, this gives a good set of handles on turbulence modelling without introducing too much degrees of freedom. To reduce them even more, one could also set $\beta_T = 0$ and disable "slow eddy suppression" although it makes nonsense of the notion of $\beta_T \tau$ as the turnover time of an eddy.

Despite the fact that some important features of real turbulence are modelled by the ODT-FHD, certain details do not match and it is currently not able to produce new extrema of flow properties by swapping fluid packages as the original ODT can do. This is due to the under-resolved nature of the formulation and could be addressed in future research. To improve the minutiae of turbulence features $u(y)$ could probably be tuned towards the desired effects.

We should also keep in mind that the basic ODT formulation used assumes incompressibility. In [1] an extension for compressible flows is suggested which should be considered in future works with ODT-FHD.

I am still confused. But on a higher level.
- *Enrico Fermi*

Appendices

Appendix A

Additional Computations

Computations which are too lengthy and detailed for the running text of this thesis can be found here.

A.1 Derivative of Accepted Eddy Probability w.r.t. Mean Flow Velocity

On Page 75 we compute $\frac{\partial P_{\mathtt{A}}}{\partial \bar{u}}$. The following are intermediate steps towards this goal

$$\frac{\partial \psi}{\partial \bar{u}} = \frac{2.1}{\bar{u}\ln(10)},$$

$$\frac{\partial}{\partial \bar{u}}\left(\frac{2\psi^2+3\psi+1}{\psi^2}\right) = -\frac{3\psi+2}{\psi^3}\cdot\frac{2.1}{\bar{u}\ln(10)} = -\frac{6.3\psi+4.2}{\bar{u}\ln(10)\psi^3},$$

$$\begin{aligned}\frac{\partial u_{max}}{\partial \bar{u}} &= \frac{1}{2}\cdot\frac{2\psi^2+3\psi+1}{\psi^2} + \frac{\bar{u}}{2}\frac{\partial}{\partial \bar{u}}\left(\frac{2\psi^2+3\psi+1}{\psi^2}\right)\\ &= \frac{2\psi^2+3\psi+1}{2\psi^2} - \frac{3.15\psi+2.1}{\ln(10)\psi^3},\end{aligned}$$

$$\frac{\partial}{\partial \bar{u}}\left[1-\left(1-\frac{\ell}{L}\right)^{1/\psi}\right] = \psi^{-2}\left(1-\frac{\ell}{L}\right)^{1/\psi}\ln\left(1-\frac{\ell}{L}\right)\cdot\frac{2.1}{\bar{u}\ln(10)},$$

$$\begin{aligned}\frac{\partial \Delta u}{\partial \bar{u}} &= \frac{\partial u_{max}}{\partial \bar{u}}\left[1-\left(1-\frac{\ell}{L}\right)^{1/\psi}\right] + u_{max}\frac{\partial}{\partial \bar{u}}\left[1-\left(1-\frac{\ell}{L}\right)^{1/\psi}\right]\\ &= \left(\frac{2\psi^2+3\psi+1}{2\psi^2} - \frac{3.15\psi+2.1}{\ln(10)\psi^3}\right)\left[1-\left(1-\frac{\ell}{L}\right)^{1/\psi}\right]\end{aligned}$$

$$+u_{max}\,\psi^{-2}\left(1-\frac{\ell}{L}\right)^{1/\psi}\ln\left(1-\frac{\ell}{L}\right)\cdot\frac{2.1}{\bar{u}\ln(10)}.$$

A.2 Conditional Expected Value of Eddy Flux

In Subsection 4.4.2 we compute the expected value of the eddy flux $\beta_T\tau\cdot\mathcal{E}^k_{i+1/2}$ for case 1. In the following the other cases are considered.

A.2.1 Case 2

Cell integral after eddy:

$$\int\limits_{x_{i-1/2}}^{x_{i+1/2}} q(\zeta, t_k+\beta_T\tau)\,d\zeta$$

$$=\sum_{j=0}^{3}\frac{\Delta_{k+1}}{2}\left(\mathcal{Q}_k+\mathcal{Q}_{k+1}\right)$$

$$=\frac{\Delta_1}{2}\left(\mathcal{Q}_0+\mathcal{Q}_1\right)+\frac{\Delta_2}{2}\left(\mathcal{Q}_1+\mathcal{Q}_2\right)+\frac{\Delta_3}{2}\left(\mathcal{Q}_2+\mathcal{Q}_3\right)$$

$$=\frac{1}{2}\left[\Delta_1\mathcal{Q}_0+(\Delta_1+\Delta_2)\mathcal{Q}_1+(\Delta_2+\Delta_3)\mathcal{Q}_2+\Delta_3\mathcal{Q}_3\right]$$

Inserting Δ_k and $\mathcal{Q}_k$ and reformulating in terms of $\mathcal{Q}_0$ yields:

$$=\frac{1}{2}\Bigg[\Delta_1\mathcal{Q}_0+(\Delta_1+\Delta_2)(\mathcal{Q}_0+S\Delta\zeta_l)+(\Delta_2+\Delta_3)\left(\mathcal{Q}_1+3S\frac{\ell}{3}\right)$$
$$+\Delta_3\left(\mathcal{Q}_2-3S\left(\Delta\zeta_r-\frac{\ell}{3}\right)\right)\Bigg]$$

$$=\frac{1}{2}\Bigg[\Delta_1\mathcal{Q}_0+(\Delta_1+\Delta_2)(\mathcal{Q}_0+S\Delta\zeta_l)+(\Delta_2+\Delta_3)\left(\mathcal{Q}_1+3S\frac{\ell}{3}\right)$$
$$+\Delta_3\left(\mathcal{Q}_2-3S\left(\Delta\zeta_r-\frac{\ell}{3}\right)\right)\Bigg]$$

$$=\Delta x\mathcal{Q}_0+\frac{S}{2}\Bigg[\Delta\zeta_l^2+\frac{2}{3}\ell\Delta\zeta_l+\frac{1}{3}\ell^2+2\Delta\zeta_r\Delta\zeta_l-\frac{2}{3}\ell\Delta\zeta_l+3\ell\Delta\zeta_r$$
$$-\ell^2-3\Delta\zeta_r^2+\ell\Delta\zeta_r\Bigg]$$

$$=\Delta x\mathcal{Q}_0+\frac{S}{2}\left[\Delta\zeta_l^2+2\Delta\zeta_r\Delta\zeta_l+4\ell\Delta\zeta_r-\frac{2}{3}\ell^2-3\Delta\zeta_r^2\right]$$

Now we substitute $\Delta\zeta_l = \Delta x - \Delta\zeta_r$:

$$= \Delta x \mathcal{Q}_0 + \frac{S}{2}\Big[\Delta x^2 - 2\Delta x\Delta\zeta_r + \Delta\zeta_r^2 + 2\Delta x\Delta\zeta_r - 2\Delta\zeta_r^2$$

$$+4\ell\Delta\zeta_r - \frac{2}{3}\ell^2 - 3\Delta\zeta_r^2\Big]$$

$$= \Delta x \mathcal{Q}_0 + \frac{S}{2}\left[\Delta x^2 + 4\ell\Delta\zeta_r - \frac{2}{3}\ell^2 - 4\Delta\zeta_r^2\right].$$

Eddy flux:

$$\beta_T\tau \cdot \mathcal{E}^k_{i+1/2} = \frac{S}{2}\left[4\ell\Delta\zeta_r - \frac{2}{3}\ell^2 - 4\Delta\zeta_r^2\right].$$

Expected values of $\Delta\zeta_r$ and $\Delta\zeta_r^2$:

$$E_{\ell,2}(\Delta\zeta_r) = \int\limits_{\ell/3}^{2\ell/3} \zeta \cdot \frac{1}{\ell/3}\, d\zeta = \frac{3}{\ell}\left[\frac{1}{2}\zeta^2\right]_{\ell/3}^{2\ell/3} = \frac{\ell}{2},$$

$$E_{\ell,2}(\Delta\zeta_r^2) = \int\limits_{\ell/3}^{2\ell/3} \zeta^2 \cdot \frac{1}{\ell/3}\, d\zeta = \frac{3}{\ell}\left[\frac{1}{3}\zeta^3\right]_{\ell/3}^{2\ell/3} = \frac{7}{27}\ell^2.$$

This leads to the expected value of the eddy flux

$$E_{\ell,2}(\beta_T\tau \cdot \mathcal{E}^k_{i+1/2}) = \frac{4}{27}S\ell^2.$$

A.2.2 Case 3

Cell integral after eddy:

$$\int\limits_{x_{i-1/2}}^{x_{i+1/2}} q(\zeta, t_k + \beta_T\tau)\, d\zeta$$

$$= \sum_{j=0}^{3} \frac{\Delta_{k+1}}{2}\left(\mathcal{Q}_k + \mathcal{Q}_{k+1}\right)$$

$$= \frac{\Delta_1}{2}\left(\mathcal{Q}_0 + \mathcal{Q}_1\right) + \frac{\Delta_2}{2}\left(\mathcal{Q}_1 + \mathcal{Q}_2\right) + \frac{\Delta_3}{2}\left(\mathcal{Q}_2 + \mathcal{Q}_3\right) + \frac{\Delta_4}{2}\left(\mathcal{Q}_3 + \mathcal{Q}_4\right)$$

$$= \frac{1}{2}\left[\Delta_1\mathcal{Q}_0 + (\Delta_1 + \Delta_2)\mathcal{Q}_1 + (\Delta_2 + \Delta_3)\mathcal{Q}_2 + (\Delta_3 + \Delta_4)\mathcal{Q}_3 + \Delta_4\mathcal{Q}_4\right]$$

$$
\begin{aligned}
&= \frac{1}{2}\left[\Delta_1 \mathcal{Q}_0 + (\Delta_1+\Delta_2)(\mathcal{Q}_0 + S\Delta\zeta_l) + (\Delta_2+\Delta_3)\left(\mathcal{Q}_1 + 3S\frac{\ell}{3}\right)\right.\\
&\qquad \left. + (\Delta_3+\Delta_4)\left(\mathcal{Q}_2 - 3S\frac{\ell}{3}\right) + \Delta_4\left(\mathcal{Q}_3 + 3S\left(\Delta\zeta_r - \frac{2\ell}{3}\right)\right)\right]\\
&= \Delta x \mathcal{Q}_0 + \frac{S}{2}\left[\Delta\zeta_l^2 + 2\ell^2 + 2\Delta\zeta_l\Delta\zeta_r + 3\Delta\zeta_r^2 - 4\ell\Delta\zeta_r\right]\\
&= \Delta x \mathcal{Q}_0 + \frac{S}{2}\left[\Delta x^2 + 2\ell^2 + 2\Delta\zeta_r^2 - 4\ell\Delta\zeta_r\right].
\end{aligned}
$$

Eddy flux:

$$\beta_T\tau \cdot \mathcal{E}^k_{i+1/2} = \frac{S}{2}\left[2\ell^2 + 2\Delta\zeta_r^2 - 4\ell\Delta\zeta_r\right].$$

Expected values of $\Delta\zeta_r$ and $\Delta\zeta_r^2$:

$$E_{\ell,3}(\Delta\zeta_r) = \int_{2\ell/3}^{\ell} \zeta \cdot \frac{1}{\ell/3}\, d\zeta = \frac{3}{\ell}\left[\frac{1}{2}\zeta^2\right]_{2\ell/3}^{\ell} = \frac{5}{6}\ell,$$

$$E_{\ell,3}(\Delta\zeta_r^2) = \int_{2\ell/3}^{\ell} \zeta^2 \cdot \frac{1}{\ell/3}\, d\zeta = \frac{3}{\ell}\left[\frac{1}{3}\zeta^3\right]_{2\ell/3}^{2\ell} = \frac{19}{27}\ell^2.$$

This leads to the expected value of the eddy flux

$$E_{\ell,3}(\beta_T\tau \cdot \mathcal{E}^k_{i+1/2}) = \frac{S}{27}\ell^2.$$

Appendix B

Code Documentation

This short documentation addresses new users of the SEC-code as well as those unfamiliar with the features implemented during phase two of the CRC 1029. The first section is dedicated to the basics as the structure of the code and its handling substantially differs in some points from the original SEC-code by Berndt. The following sections shall help users to set up simulations including the features described within this thesis. Configuration variables are explained alongside suggested value ranges. Also minimal working examples are given and additional options are shown.

B.1 General

The minimal working example for a basic simulation with the current SEC-code is indeed very short:

```
% Set-up
nel = 100;
config = ArrheniusKinetics();
config.iv = repmat(config.kineticsif.setTPX(1,1,'F:1'),1,nel);
config.grid.dx = 1/nel;

% Start simulation
rest = RunSimulation(config,[]);
```

Here, we only set `nel` the number of grid cells, decided for a chemical and thermodynamic model by using perfect gas 2-species Arrhenius kinetics, set the initial values as $T = 1$, $p = 1$, $X_{fuel} = 1$ everywhere (please note that flow quantities may be nondimensional depending on the `Kinetics` chosen as pointed out in Chapter 1), calculated the grid width assuming the domain has length one. Finally, we start the simulation for as long as we keep the figure open which will pop up showing us the current state of the flow. All other variables and functions are preset by calling the kinetics function. The default values can be examined in `BaseConfig.m`.

Typical usages of the code will require the change of boundary conditions, storage of intermediate solutions, include axial cross-sectional area variation and a stopping criterion. Such a script could look like this:

```
% Basic set-up
nel = 100;
config = ArrheniusKinetics();
config.iv = repmat(config.kineticsif.setTPX(1,1,'F:1'),1,nel);
config.grid.dx = 1/nel;

% Boundary conditions
config.boundary(1).fn = @ReflectingBoundary;
config.boundary(2).fn = @ExpansionBoundary;
config.boundary(2).outerPressure = 1;

% Cross-sectional area
config.grid.A = linspace(1,4,nel+1);

% Store intermediate solutions at time multiples of 1e-3
global rests
config.plot.heatmaps.enabled = 1;
config.hitZeroTimeMod = 1e-3;

% Start simulation and run until t > 5
rest = RunSimulation(config, [], @(config,rest) rest.t > 5);
```

Boundary conditions are set through an array of structure arrays, defining function handles (1 is left, 2 is right). Continuous boundaries are default but others like reflecting walls and isentropic expansion into an infinite plenum chamber at fixed pressure are provided, too. These are all functions ending with "`Boundary`". Some might require additional settings as the plenum pressure for `ExpansionBoundary`. For own boundary conditions copy the structure of the given functions or use `FnBoundary`.

The cross-sectional area must be given as an array of $A_{i\pm1/2}$ - the area at the interfaces. This differs from the former usage of this variable, where it was defined on the cell midpoints. It is not recommended to work with big jumps in this variable as this could break the equations used for its handling.

Intermediate snapshots of the solution are stored in a global variable called `rests`. To enable these snapshots heatmaps must be activated and will pop up after the simulation. `hitZeroTimeMod` controls the resolution of this time line and modifies the time step Δt^k if needed such that all multiples of the given time are met.

The callback function is a handle using `config` and `rest` and can be given any form while the most commonly used will probably be the one shown here.

More complex demonstration scripts are included in the code package's root directory. Some of them have been used to produce the results discussed in this thesis others originate from the first phase and have been updated to match the new structures.

B.2 Moving Mesh

To include adaptive remeshing in a simulation some more information are needed and some structures change as will be explain by taking the example of the following short script:

```
% Basic set-up
nel = 100;
config = ArrheniusKinetics();
config.iv = repmat(config.kineticsif.setTPX(1,1,'F:1'),1,nel);

% Enable and configure moving mesh
config.grid.moveMesh = true;
config.grid.beta = 0.9;
config.grid.monitor = 1;
config.grid.meshQuantities = [6 7];
config.grid.direction = 2;
config.grid.maxiter = 5;
config.grid.smoothingSteps = 1;
% config.grid.mindx = 0.001;

% Set grid
config.grid.dx = repmat(1/nel,1,nel);
config.grid.x = linspace(0,1,nel+1);

% Start simulation
rest = RunSimulation(config,[]);
```

Actually, the lines $\texttt{config.grid.moveMesh} = \texttt{true}$ and $\texttt{config.grid.dx} = \texttt{repmat}(1/\texttt{nel}, 1, \texttt{nel})$ or $\texttt{config.grid.x} = \texttt{linspace}(0, 1, \texttt{nel} + 1)$ would suffice to start a simulation with moving mesh but it is recommended to set the other variables explicitly. `beta` is the variable controlling the number of points within critical regions (see (3.7)), where close to zero means nearly equidistant grid cell distribution and close to one means nearly all cells cluster in one sharp-gradient region. Values of 0.8 to 0.9 often yield very good results.

`monitor` decides for the monitor function to be utilised. 1 stands for (3.7) on the computational grid whereas 2 uses the physical domain to compute gradients. Both include the changes described in Subsection 3.2.3. When simulating strong shock waves, “2” might break down due to the sharp gradients, while “1” is robust but less effective.

The vector `meshQuantities` contains the indices of the flow quantities which will be taken into account when computing the gradients for the monitor function. All quantities not included here will have no effect of the mesh movement. Usually, pressure and temperature should be tracked, fuel species and velocity could be a suitable addition if the result is not satisfactory.

`smoothingSteps` is the number of low-pass filters (3.8) that are applied to the monitor values after their calculation. If chosen too high, the mesh will not contract fast or strong enough in critical regions, if chosen too low grid points might be stuck in low-gradient regions when a strong gradient dissolves. At least one smoothing step is highly recommended. Two or three might be favourable is some situations.

For the control of the Gauss-Seidel iteration (3.6) `direction` decides for the interface to begin with (1 for left, 2 for both directions and than taking the average, 3 for right). Sometimes it is crucial to choose the direction carefully. “2” is a save but slower choice while the others can be advantageous when strong gradients travel to the right (“1”) or to the left(“3”). The maximum number of GS iterations is given in `maxiter`, therefore, the computational efficiency depends on this number. A single step can be sufficient if time steps are very small or the the fluid is rather quiescent (which is typically not the case for MM simulations). If changes are faster, “5” has proven to be a reliable value balancing accuracy and computation speed.

Setting a lower bound for the MM with `mindx` is fully optional and might seem strange at first. It is included to ensure assumptions for cross-sectional averaging are met as they are the basis of most applications of the SEC-code. If the bound is set, it will be strictly kept otherwise the mesh is computed as seems fit according to the algorithm.

If the cross-sectional area is to vary within this context, we need not only an array for `config.grid.A` but also a function from which `A` will be calculated anew after the computational grid is changed. `config.grid.A` will be initialised as `config.grid.Afn(config.grid.x)` automatically if a function is given but `A` is still a scalar.

For the initial grid, either `x` - the array of cell interfaces - must be given or `dx` must be an array. Instead of defining a mesh manually, the function `InitialiseGrid` may be called to optimise the initial grid automatically already using the moving mesh approach: As long as the grid moves significantly between the iterations and the maximum number of steps is not yet reached, one step of GS iteration in the sense of (3.6) is performed after assigning initial values to the midpoint states. These are generated by user-given functions of T, p, u and mole fraction vector X. For the grid calculation the chosen monitor function is used along with the guiding quantities like later during the actual simulation. For the smoothening steps a different choice can be made. This is sensible because the functions of the initial values might include discontinuities, which cause the mesh points to travel too fast, leaving unwanted gaps. Although it is not necessary, it is strongly recommended to either use this grid initialisation function or supply a well suited grid oneself. Otherwise the simulation loses accuracy in the first few time steps. Albeit, for sharp gradients and jumps this method is quite slow.

B.3 One-Dimensional Turbulence

As the values suitable for a simulation with ODT highly depend on the step-up considered, some parameter have no default values implemented. The required variables must be chosen by hand. A minimal working example, therefore, looks like this

```
% Basic set-up
nel = 100;
config = ArrheniusKinetics_ODT();
config.iv = repmat(config.kineticsif.setTPX(1,1,'F:1'),1,nel);
config.grid.dx = 1/nel;

% Set up ODT variables and parameters
config.odt.enabled = true;
config.odt.A = 1;
config.odt.C = 200;
config.odt.beta = 0.5;

config.odt.Lmin = 1e-5;
config.odt.Lmax = 0.01;
config.odt.Lp = sqrt(config.odt.Lmax * config.odt.Lmin);

config.odt.lambda = config.odt.A * config.odt.C * 1e5;

% Start simulation
rest = RunSimulation(config,[]);
```

At first, a word on `ArrheniusKinetics_ODT`: The original `ArrheniusKinetics` includes a thick flame model where dynamic viscosity, thermal conductivity and diffusion coefficients are raised to smear the flame front. For ODT such a high viscosity is pointless as it leads to a suppression of all eddies. Therefore, a second function was created which uses the same equations except for the much lower viscosity. The other transport coefficients have been adjusted, too. All values were chosen to lie within a range reasonable for gases like hydrogen or methane at high temperatures (cf. [24, 19]) and nondimensionalised.

Now to the actual ODT setting. Surely, `config.odt.enabled` = `true` is the most important line because it switches ODT on. `A`, `C` and `beta` are the model parameter $\mathcal{A}, \mathcal{C}$ and β_T discussed in Subsection 4.4.3. Consult this section for their influence on the calculations and hints for choosing the correct values.

The following three variables `Lmin`, `Lmax` and `Lp` control the shape of the eddy length distribution (4.13) from which eddy sizes are drawn. Naturally, `Lmin`, `Lmax` are the smallest and the biggest size allowed, respectively, and `Lp` is the most probable eddy length. `Lmin` should be chosen such that it is smaller than the size of small eddy suppression which depends on $\mathcal{A}$ as explained in Subsection 4.4.3. `Lmax` will usually be predetermined by the diameter of the SEC-tube. `Lp` is arbitrary but might be chosen as above, following [50].

The real eddy occurrence rate λ depends linearly on $\mathcal{A}$ and $\mathcal{C}$. Hence, they should be included as factors. The above choice is a solid initialisation and should not be worried about to much, for `lambda` is only a first guess and will be automatically adapted according to the choices of `probgoal`, `probmax`, `lambfac` and `nprob_check`, which are set by default. If the acceptance probability of an eddy (4.11) is higher than `probmax` `lambda` is raised. If the average acceptance probability after `nprob_check` eddy trials is lower than `probgoal` then `lambda` is lowered by a factor of `lambfac` at maximum. The default values of these variables are chosen according to the recommendation of the authors of the aODT-code of Heiko Schmidt's group.

B.4 One-Dimensional Network Model

As the combination of multiple one-dimensional domains in a two-dimensional fashion is a rather harsh interference with the general structure of the SEC-code, it needs a more complex set-up than all other features. The following example configures two aligned domains (the left one with raised pressure) as if cutting one big domain in halves:

```
% Set-up two domains
nel = 100;
for i = 1:2
    config(i) = ArrheniusKinetics();
    config(i).grid.dx = repmat(1/nel,1,nel);
    config(i).grid.x = linspace(0,1,nel+1);
end

% Different initial values
config(1).iv = repmat(config(1).kineticsif.setTPX(1,5,'P:1'),1,nel);
config(2).iv = repmat(config(2).kineticsif.setTPX(1,1,'P:1'),1,nel);
```

```
% Non-interacting BC
config(1).boundary(1).fn = @ReflectingBoundary;
config(2).boundary(2).fn = @ReflectingBoundary;

% Interacting boundary conditions
config(1).boundary(2).interacting = true;
config(1).boundary(2).interacting_domain = 2;
config(1).boundary(2).interacting_side = 1;
config(1).boundary(2).interacting_cells = 1;
config(1).boundary(2).fn = @FnBoundary;
config(1).boundary(2).bfn = @OneDInteractionBoundary;

config(2).boundary(1).interacting = true;
config(2).boundary(1).interacting_domain = 1;
config(2).boundary(1).interacting_side = 2;
config(2).boundary(1).interacting_cells = nel;
config(2).boundary(1).fn = @FnBoundary;
config(2).boundary(1).bfn = @OneDInteractionBoundary;

% Start simulation
rest = RunSimulation(config,[]);

function [ border, A, x ]...
  = OneDInteractionBoundary(config, side, rest, dn)

    id = config(dn).boundary(side).interacting_domain;
    ic = config(dn).boundary(side).interacting_cells;
    border = rest(id).data(:,ic);
    A = rest(id).A;
    if side == 1
        x = rest(dn).x(1) - rest(id).dx(ic);
    else
        x = rest(dn).x(end) + rest(id).dx(ic);
    end
end
```

Multiple domains are enabled by default so if `config` is an array of n structure arrays the code will automatically compute solutions for n domains stored in `rest` which is an array then, too. For `rests`, the first index becomes the domain number and the second stands for the time.

As we usually want the domains which are simulated to interact with each other, most of the work is in the boundary conditions as can be seen in the example script. Although the domains are ordered by their index, which determines the order in which solutions are calculated, the arrangement in space is only given through the boundary conditions. In the above example, domain 1 (D1) is left, domain 2 (D2) is right. That is why the left boundary of D1 and the right boundary of D2 are reflecting while the others are interacting. For such boundaries, we need to define the boundary as `interacting` and provide information about the domain number it is interacting with, the side of the other domain which is interactive (1: left, 2: right, 3: top, 4: bottom), the index of the interacting cell of the other domain and of course a boundary function. A simple boundary function for this case is given by `OneDInteractionBoundary`.

Even if MM is not used with multiple domains, `x` and `dx` must be defined as arrays due to the flux correction process. If two-dimensional interacting is to be simulated, it is crucial to choose the domain order such that the one resolving the interacting grid cell interface the finest gets a higher number. This is due to the fact that the flux correction process throws away the first flux calculated over that interface and uses the second one to advance the cell values. In the typical application this means setting plenum chambers as last domains.

This was a one-dimensional example which is only part of the network feature. The most interesting application will be the coupling of domains which are not aligned. The structure is much more complex then the above approach and would go beyond the scope of an example given here. Therefore, any user interested in working with the network feature should look into the helper function `Setup_FullMachine` covering the most important configurations. The function configures the domains as described and used in Section 2.2. It can be used as black box or as guideline to write an own set-up function.

Lists

List of Notations

c_V	specific heat-capacity at constant volume
e	internal energy
f	one-dimensional flux function of Euler equations
f^{rel}	f with velocity relative to moving grid
g	two-dimensional lateral flux function of Euler equations
ℓ	eddy length
ℓ_{min}, ℓ_{max}, ℓ_p	minimum, maximum, most probable eddy length
m	mass
$\vec{n}$	normal vector
p	pressure
p_s	stagnation point pressure
p^*	pressure of numerical flux function
q	continuous solution vector function
q^{rel}	q with velocity relative to moving grid
t	continuous time
t_k	discrete time level
u	axial flow velocity
v	lateral flow velocity
w	Fornberg weight function

x	first (axial) spacial direction
x_i, $x_{i-1/2}$	grid cell midpoints and interface
x_0	eddy's left edge
y	second (lateral) spacial direction
y_0	eddy's bottom edge
A	cross-sectional area of domain
$\mathcal{A}$	ODT model parameter
$\mathtt{A}$	event of accepting eddy
$\mathcal{C}$	ODT model parameter
D_i, D_{ij}	mixture averaged single species and binary mass diffusion coefficients
$\mathcal{D}$	numerical molecular diffusion flux
E	total energy
$\mathcal{E}$	numerical eddy flux
F	numerical one-dimensional flux function of Euler equations
$\mathcal{F}$	corrected numerical flux for Euler equations of moving mesh
G	numerical two-dimensional lateral flux function of Euler equations
$\mathtt{I}$	event of implementing eddy
K	edge of space-time element
L	integral length scale
$\mathcal{L}$	domain length
$\mathfrak{L}$	random variable of eddy length
M	molar mass
$\mathcal{M}$	Mach number
N	number of grid cells
N_{spec}	number of species
Q	discrete solution vector

$\mathbb{Q}$	discrete solution vector in plenum
$\mathcal{Q}$	intermediate solution values in post-eddy cell
$\mathfrak{Q}$	discrete solution vector with reduced flow properties
R	domain radius
R_u	universal gas constant
R_s	specific gas constant
$\mathbb{R}$	real numbers
$\mathcal{R}$	real eddy rate
Re	Reynolds number
S	slope of solution between two grid cells
$\mathcal{S}_{ij}$	mean strain-rate tensor
T	temperature
T_s	stagnation point temperature
$\mathfrak{T}$	random variable of eddy occurrence time
$U(0,1)$	uniform distribution on $[0,1]$
V	volume
$\mathcal{V}$	velocity of grid interfaces
W, $\widetilde{W}$, W_c	diffusion velocity (correction)
X	species mole fractions
$\mathfrak{X}$	random variable of left edge of eddy
Y	species mass fractions
α	floor value for MM
β_M	MM model parameter
β_T	ODT model parameter
γ	isentropic exponent
$\delta_{\Delta t_{eddy}}$, δ_ℓ, δ_{x_0}	distribution functions
ε	dissipation rate

ζ_r, ζ_l	distance between eddy's left edge and right or left cell interface, respectively
η	Kolmogorov length scale
ϑ	angle between combustion chamber and plenum
κ	thermal conductivity
λ	eddy rate distribution
λ^*	majorant of eddy rate
μ	dynamic viscosity
ν	GS iteration step
ξ	spacial variable on uniform computational grid
ϱ	fluid density
τ	eddy turnover time scale
τ_{ij}^R, τ_{ij}^R	Reynolds or residual stress tensor, respectively
φ_a	eddy acceptance probability
χ_i	midpoint of fine resolved interaction interface
ψ	exponent of lateral velocity profile
ω	monitor function
Δ_j	spacial step width in post-eddy cell
Δx	axial width of grid cell
Δy	lateral width of grid cell
Δt^k	width of time step at t_k
Δt_{eddy}	waiting time until next eddy
Δu	velocity difference function for eddy generation
Λ	numerical eddy rate distribution
Ω_c, Ω_p	computational and physical domain
$\cdot_x$ or $\cdot_t$	partial derivative of $\cdot$ w.r.t x or t

List of Acronyms

CFL	Courant–Friedrichs–Lewy
CPC	constant pressure combustion
CRC	collaborative research center
CVC	constant volume combustion
DNS	direct numerical simulation
FHD	fluctuating hydrodynamics
FV(M)	finite volume (method)
GS	Gauss-Seidel
LEM	linear eddy model
LES	large eddy simulation
MM(M)	moving mesh (method)
MUSCL	monotonic upstream-centered scheme for conservation laws
ODT	one-dimensional turbulence
PDC	pulse detonation combustion
PDE	partial differential equation
PDF	probability density function
RANS	Reynolds-averaged Navier-Stokes (equations)
SEC	shockless explosion combustion

List of Figures

List of Tables

Bibliography

[1] W. T. Ashurst and A. R. Kerstein. “One-Dimensional Turbulence: Variable-Density Formulation and Application to Mixing Layers”. In: *Physics of Fluids* 17.2 (Feb. 2005), p. 025107. DOI: 10.1063/1.1847413 (cit. on pp. 60, 93).

[2] P. Attard. *Non-Equilibrium Thermodynamics and Statistical Mechanics: Foundations and Applications*. 1st ed. Oxford: Oxford University Press, 2012. 462 pp. (cit. on p. 72).

[3] E. Aurell, E. Dormy, and P. Frick. “Binary Tree Models of High-Reynolds-Number Turbulence”. In: *Physical Review E* 56.2 (Aug. 1, 1997), pp. 1692–1698. DOI: 10.1103/PhysRevE.56.1692 (cit. on p. 57).

[4] L. Böswirth and S. Bschorer. *Technische Strömungslehre: Lehr- und Übungsbuch*. In collab. with T. Buck. 10., überarb. und erw. Aufl. Lehrbuch. Wiesbaden: Springer Vieweg, 2014. 356 pp. (cit. on pp. 66, 67).

[5] G. Beckett et al. “On The Numerical Solution of One-Dimensional PDEs Using Adaptive Methods Based on Equidistribution”. In: *Journal of Computational Physics* 167.2 (Mar. 2001), pp. 372–392. DOI: 10.1006/jcph.2000.6679 (cit. on pp. 29, 30).

[6] M. Berger and P. Colella. “Local Adaptive Mesh Refinement for Shock Hydrodynamics”. In: *Journal of Computational Physics* 82.1 (May 1989), pp. 64–84. DOI: 10.1016/0021-9991(89)90035-1 (cit. on p. 12).

[7] P. Berndt. “Mathematical Modeling of the Shockless Explosion Combustion”. PhD Thesis. Freie Universität Berlin, 2016 (cit. on pp. 3–5, 9, 37).

[8] P. Berndt and R. Klein. “Modeling the Kinetics of the Shockless Explosion Combustion”. In: *Combustion and Flame* 175 (Jan. 2017), pp. 16–26. DOI: 10.1016/j.combustflame.2016.06.029 (cit. on p. 6).

[9] A. K. Bhattacharjee et al. “Fluctuating Hydrodynamics of Multi-Species Reactive Mixtures”. In: *The Journal of Chemical Physics* 142.22 (June 14, 2015), p. 224107. DOI: 10.1063/1.4922308 (cit. on p. 72).

[10] B. C. Bobusch et al. “Shockless Explosion Combustion: An Innovative Way of Efficient Constant Volume Combustion in Gas Turbines”. In: *Combustion Science and Technology* 186.10-11 (Nov. 2, 2014), pp. 1680–1689. DOI: 10.1080/00102202.2014.935624 (cit. on p. 3).

[11] B. C. Bobusch. “Fluidic Devices for Realizing the Shockless Explosion Combustion Process”. In: (2014), p. 188 (cit. on p. 3).

[12] T. Butler and P. O’Rourke. “A Numerical Method for Two Dimensional Unsteady Reacting Flows”. In: *Symposium (International) on Combustion* 16.1 (Jan. 1977), pp. 1503–1515. DOI: 10.1016/S0082-0784(77)80432-3 (cit. on pp. 8, 86).

[13] S. Cao and T. Echekki. “A Low-Dimensional Stochastic Closure Model for Combustion Large-Eddy Simulation”. In: *Journal of Turbulence* 9 (Jan. 2008), N2. DOI: 10.1080/14685240701790714 (cit. on pp. 50, 62).

[14] T Coffee and J Heimerl. “Transport Algorithms for Premixed, Laminar Steady-State Flames”. In: *Combustion and Flame* 43 (1981), pp. 273–289. DOI: 10.1016/0010-2180(81)90027-4 (cit. on p. 69).

[15] H. Cohen, G. Rogers, and H. Saravanamuttoo. *Gas Turbine Theory*. London: Longman Group Limited, 1996 (cit. on p. 18).

[16] P. Constantin, P. D. Lax, and A. Majda. “A Simple One-Dimensional Model for the Three-Dimensional Vorticity Equation”. In: *Communications on Pure and Applied Mathematics* 38.6 (1985), pp. 715–724. DOI: 10.1002/cpa.3160380605 (cit. on p. 57).

[17] J. Dahl. *Jet Engine*. In: *Wikipedia, The Free Encyclopedia*. Wikimedia Commons, Dec. 14, 2020 (cit. on p. 2).

[18] P. A. Davidson. *Turbulence: An Introduction for Scientists and Engineers*. Second edition. Oxford, United Kingdom ; New York, NY, United States of America: Oxford University Press, 2015. 630 pp. (cit. on pp. 51, 57, 66).

[19] *Diffusion Coefficient of Methane from Dortmund Data Bank*. URL: http://www.ddbst.com/en/EED/PCP/DIF_C1051.php (visited on 06/10/2021) (cit. on pp. 7, 104).

[20] I. Eames and J. B. Flor. “New Developments in Understanding Interfacial Processes in Turbulent Flows”. In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 369.1937 (Feb. 28, 2011), pp. 702–705. DOI: 10.1098/rsta.2010.0332 (cit. on p. 51).

[21] T. Echekki, A. R. Kerstein, and J. C. Sutherland. “One-Dimensional Turbulence Model”. In: *Turbulent Combustion Modeling: Advances, New Trends and Perspectives*. 95. Jan. 2011, pp. 249–276 (cit. on p. 61).

[22] T. Echekki et al. “‘One-Dimensional Turbulence’Simulation of Turbulent Jet Diffusion Flames: Model Formulation and Illustrative Applications”. In: *Combustion and flame* 125.3 (2001), pp. 1083–1105 (cit. on pp. 50, 67).

[23] B. Einfeldt. “On Godunov-Type Methods for Gas Dynamics”. In: *SIAM Journal on Numerical Analysis* 25.2 (Apr. 1988), pp. 294–318. DOI: 10.1137/0725021 (cit. on p. 5).

[24] *Engineering ToolBox.* URL: https://www.engineeringtoolbox.com/ (visited on 06/10/2021) (cit. on pp. 7, 104).

[25] R. Fazio and R. LeVeque. "Moving-Mesh Methods for One-Dimensional Hyperbolic Problems Using CLAWPACK". In: *Computers & Mathematics with Applications* 45.1-3 (Jan. 2003), pp. 273–298. DOI: 10.1016/S0898-1221(03)80019-6 (cit. on p. 32).

[26] *fdcoeffF(k,Xbar,x).* URL: https://faculty.washington.edu/rjl/fdmbook/matlab/fdcoeffF.m (visited on 09/28/2020) (cit. on p. 42).

[27] B. Fornberg. "Generation of Finite Difference Formulas on Arbitrarily Spaced Grids". In: *Mathematics of Computation* 51.184 (1988), p. 8 (cit. on p. 40).

[28] C. Fukushima and J. Westerweel. *False Color Image of the Far Field of a Submerged Turbulent Jet.* In: *Wikipedia, The Free Encyclopedia.* Wikimedia Commons, Nov. 13, 2007 (cit. on p. 51).

[29] A. Harten, P. D. Lax, and B. van Leer. "On Upstream Differencing and Godunov-Type Schemes for Hyperbolic Conservation Laws". In: *SIAM review* 25.1 (1983), pp. 35–61 (cit. on p. 5).

[30] D. Hawken, J. Gottlieb, and J. Hansen. "Review of Some Adaptive Node-Movement Techniques in Finite-Element and Finite-Difference Solutions of Partial Differential Equations". In: *Journal of Computational Physics* 95.2 (Aug. 1991), pp. 254–302. DOI: 10.1016/0021-9991(91)90277-R (cit. on p. 29).

[31] W. H. Heiser and D. T. Pratt. "Thermodynamic Cycle Analysis of Pulse Detonation Engines". In: *Journal of Propulsion and Power* 18.1 (Jan. 2002), pp. 68–76. DOI: 10.2514/2.5899 (cit. on p. 1).

[32] W. Huang and R. D. Russell. *Adaptive Moving Mesh Methods.* Vol. 174. Applied Mathematical Sciences. New York, NY: Springer New York, 2011. DOI: 10.1007/978-1-4419-7916-2 (cit. on p. 29).

[33] Jess. *Nderiv_fornberg(k, xPts, u).* Version 1.0. MATLAB Central File Exchange, 2015 (cit. on p. 42).

[34] P. Junglas. *Strömungslehre 2.* Turbulente Strömung in kreisförmigen Rohren. Apr. 19, 2021. URL: http://www.peter-junglas.de/fh/vorlesungen/stroemungslehre2/html/kap1-4-3.html (visited on 04/19/2021) (cit. on p. 67).

[35] R. J. Kee, M. E. Coltrin, and P. Glarborg. *Chemically Reacting Flow: Theory and Practice.* Hoboken, N.J: Wiley-Interscience, 2003. 848 pp. (cit. on pp. 6, 19, 69).

[36] A. R. Kerstein. "Computational Study of Propagating Fronts in a Lattice-Gas Model". In: *Journal of Statistical Physics* 45.5-6 (1986), pp. 921–931 (cit. on p. 57).

[37] A. R. Kerstein. “A Linear- Eddy Model of Turbulent Scalar Transport and Mixing”. In: *Combustion Science and Technology* 60.4-6 (Aug. 1988), pp. 391–421. DOI: `10.1080/00102208808923995` (cit. on p. 50).

[38] A. R. Kerstein. “Linear-Eddy Modelling of Turbulent Transport. Part 6. Microstructure of Diffusive Scalar Mixing Fields”. In: *Journal of Fluid Mechanics* 231 (Oct. 1991), p. 361. DOI: `10.1017/S0022112091003439` (cit. on p. 59).

[39] A. R. Kerstein. “One-Dimensional Turbulence: Model Formulation and Application to Homogeneous Turbulence, Shear Flows, and Buoyant Stratified Flows”. In: *Journal of Fluid Mechanics* 392 (Aug. 10, 1999), pp. 277–334. DOI: `10.1017/S0022112099005376` (cit. on pp. 50, 58, 60–63, 67, 79, 85).

[40] A. R. Kerstein et al. *High-Resolution Modeling of Multiscale Transient Phenomena in Turbulent Boundary Layers.* 2001 (cit. on pp. 60, 67).

[41] A. R. Kerstein et al. “One-Dimensional Turbulence: Vector Formulation and Application to Free Shear Flows”. In: *Journal of Fluid Mechanics* 447 (Nov. 2001), pp. 85–109. DOI: `10.1017/S0022112001005778` (cit. on pp. 61, 72).

[42] L. Landau and E. Lifshitz. *Fluid Mechanics.* Vol. 6. Course of Theoretical Physics. PergamonPress, Oxford, England, 1959 (cit. on p. 72).

[43] A. M. Law and W. D. Kelton. *Simulation Modeling and Analysis.* 3rd ed. McGraw-Hill Series in Industrial Engineering and Management Science. Boston: McGraw-Hill, 2000. 760 pp. (cit. on p. 60).

[44] R. J. LeVeque. *Finite Volume Methods for Hyperbolic Problems.* Cambridge; New York: Cambridge University Press, 2002 (cit. on p. 4).

[45] D. O. Lignell et al. “Mesh Adaption for Efficient Multiscale Implementation of One-Dimensional Turbulence”. In: *Theoretical and Computational Fluid Dynamics* 27.3-4 (June 2013), pp. 273–295. DOI: `10.1007/s00162-012-0267-9` (cit. on p. 63).

[46] B. d. G. Mars. *6 Régimes Du Cylindre Infini, d’après Lienhard.* In: *Wikipedia, The Free Encyclopedia.* Wikimedia Commons, Jan. 11, 2021 (cit. on p. 52).

[47] R. J. McDermott. “Toward One-Dimensional Turbulence Subgrid Closure for Large-Eddy Simulation”. Department of Chemical Engineering, University of Utah, 2005 (cit. on pp. 59, 63).

[48] H. Pitsch. “Turbulence (CEFRC Combustion Summer School)”. 2014 (cit. on p. 57).

[49] S. B. Pope. *Turbulent Flows.* Cambridge ; New York: Cambridge University Press, 2000. 771 pp. (cit. on p. 57).

[50] N. K. Punati. “An Eulerian One-Dimensional Turbulence Model: Application to Turbulent and Multiphase Reacting Flows”. The University of Utah, 2012 (cit. on pp. 63, 65, 105).

[51] L. F. Richardson. “Weather Prediction by Numerical Process”. In: *Cambridge University Press, Cambridge* (1922) (cit. on p. 53).

[52] A. H. Shapiro. *The Dynamics and Thermodynamics of Compressible Fluid Flow.* 1st ed. Vol. 1. New York: Wiley, 1953. 647 pp. (cit. on p. 37).

[53] G. Strang. “On the Construction and Comparison of Difference Schemes”. In: *SIAM Journal on Numerical Analysis* 5.3 (Sept. 1968), pp. 506–517. DOI: 10.1137/0705041 (cit. on p. 5).

[54] J. C. Sutherland, N. Punati, and A. R. Kerstein. “A Unified Approach to the Various Formulations of the One-Dimensional Turbulence Model”. In: *Institute for Clean and Secure Energy* (2010) (cit. on pp. 61, 72).

[55] H. Tang and T. Tang. “Adaptive Mesh Methods for One- and Two-Dimensional Hyperbolic Conservation Laws”. In: *SIAM Journal on Numerical Analysis* 41.2 (Jan. 2003), pp. 487–515. DOI: 10.1137/S003614290138437X (cit. on pp. 29, 30, 32).

[56] T. Tang. “Moving Mesh Methods for Computational Fluid Dynamics”. In: *Contemporary Mathematics.* Ed. by Z.-C. Shi et al. Vol. 383. Providence, Rhode Island: American Mathematical Society, 2005, pp. 141–173. DOI: 10.1090/conm/383/07162 (cit. on pp. 29, 32).

[57] H. Tennekes and J. L. Lumley. *A First Course in Turbulence.* The MIT Press, 1972. DOI: 10.7551/mitpress/3014.001.0001 (cit. on p. 57).

[58] J. F. Thompson, Z. U. A. Warsi, and C. W. Mastin. *Numerical Grid Generation: Foundations and Applications.* New York: North-Holland : Elsevier Science Pub. Co. [distributor], 1985. 483 pp. (cit. on p. 29).

[59] G. Tornow and R. Klein. “A 1D Multi-Tube Code for the Shockless Explosion Combustion”. In: *Active Flow and Combustion Control 2018.* Ed. by R. King. Vol. 141. Cham: Springer International Publishing, 2019, pp. 321–335. DOI: 10.1007/978-3-319-98177-2_20 (cit. on pp. 7, 9, 12, 13, 17, 21, 22, 27).

[60] E. F. Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction.* 3rd ed. Dordrecht ; New York: Springer, 2009. 724 pp. (cit. on pp. 4, 5).

[61] A. van Dam and P. Zegeling. “A Robust Moving Mesh Finite Volume Method Applied to 1D Hyperbolic Conservation Laws from Magnetohydrodynamics”. In: *Journal of Computational Physics* 216.2 (Aug. 2006), pp. 526–546. DOI: 10.1016/j.jcp.2005.12.014 (cit. on pp. 29, 31, 32, 42).

[62] S. D. Wolff and R. King. “Optimal Control for Firing Synchronization in an Annular Pulsed Detonation Combustor Mockup by Mixed-Integer Programming”. In: *AIAA Scitech 2019 Forum.* AIAA Scitech 2019 Forum. San Diego, California: American Institute of Aeronautics and Astronautics, Jan. 7, 2019. DOI: 10.2514/6.2019-1742 (cit. on p. 27).

[63] L. Zander et al. “Knock Control in Shockless Explosion Combustion by Extension of Excitation Time”. In: *Active Flow and Combustion Control 2018*. Ed. by R. King. Vol. 141. Cham: Springer International Publishing, 2019, pp. 151–166. DOI: 10.1007/978-3-319-98177-2_10 (cit. on p. 28).