Arch Comput Methods Eng (2008) 15: 67–111
DOI 10.1007/s11831-007-9014-6

Development of an Efficient 3–D CFD Software to Simulate and Visualize the Scavenging of a Two-Stroke Engine

Dirk Trescher

Received: 22 February 2007 / Accepted: 22 February 2007 / Published online: 6 December 2007

Abstract In this paper we want to describe in detail how the task of numerically solving the flow through a two-stroke engine with moving parts is solved in an efficient way. The mathematical model behind the scenes is illuminated and the used numerical schemes are specified. First, the computation of the convective flux function is carried out by the AUSMDV Riemann solver, which has been proven to be very efficient in comparison to other schemes. Then the introduction of the temperature dependency of the material properties of the fluid has augmented the realistic setting within the compression and expansion of the hot gas within the cylinder. This temperature dependency of the heat capacity causes a change in the equation of state. The gas is not polytropic any more but calorically imperfect. Thus, the use of a relaxation method is necessary in order to retain our Riemann solver. To account for the complex geometry, it was necessary to realize a special mesh treatment. The computational domain can be assembled by different meshes that are connected in a mass conservative way. Furthermore, the piston and crankshaft motion is obtained by very efficient algorithms. In order to speed up the computation of the numerical solution, different strategies have been followed. Adaptive local time-stepping has been implemented in a time consistent manner. Additionally, a dynamic local mesh adaption with hanging knots is used to reach a better resolution in critical areas. A further reduction in computational time has been obtained by the parallelization of the numerical scheme and the mesh routines. To handle this parallelization of the mesh treatment, an extended partitioning for the dynamic load balancing has been implemented. Finally, a simulation of flow through a real-world geometry of an existing two-stroke engine has been performed, the results have been validated with measured pressure data for this engine, and the flow has been qualitatively and quantitatively studied.

D. Trescher (✉)
Department for Applied Mathematics, University of Freiburg,
Hermann-Herder-Str. 10, 79104 Freiburg, Germany
e-mail: trescher@mathematik.uni-freiburg.de

1 Introduction

The two-stroke engine is widely used in many different devices. This is due to some important advantages as opposed to four-stroke engines. Above all, the lack of valves allows for a smaller size and weight. Additionally, a higher power output is achieved by the firing once every revolution. But the environmentally problematic exhaust characteristic is a big drawback of this type of engine. In the scavenging process, when the fresh charge displaces the exhaust gas in the cylinder, lies the biggest opportunity to improve the exhaust data by geometry optimization. In order to give the engineer the opportunity to analyze the flow field and to compare different engine geometries, we developed a software package to efficiently simulate the three-dimensional, time-dependent, Navier–Stokes equations within the complex, real-world geometry of the two-stroke engine with moving piston and rotating crankshaft.

1.1 Why Computational Simulations?

The measurement of the characteristics of a prototype is an effective research and development tool. But for the early stages of the development of a new engine it is very costly and laborious to build many prototypes. In order to test different possibilities within a short period of time and with an acceptable financial effort it is necessary to employ computational simulations.

1.2 Why 3-D Fluid Dynamics?

The most efficient possibility for reducing the pollutant emission is to minimize the loss of scavenging. By means of the numerical simulation of gas motion in the cylinder, one can determine which changes of the geometry of the cylinder, transfer ducts, and exhaust port can improve these scavenging characteristics. One-dimensional models cannot capture the complexity of the flow pattern within the cylinder and are therefore not suited for geometrical optimization. However, the analysis conducted with these one-dimensional models can provide a good starting point for an in-depth study.

Although three-dimensional computations usually take a lot longer than a one-dimensional calculation, a further reason in favor of the three-dimensional simulation is the possibility of visualizing each detail of the flow at an arbitrary time. A detailed analysis of flow behavior is possible.

1.3 State of the Art

What has been done to quite an extent (see e.g. [1–6]), is the analysis of two-stroke engines with commercial software. The big drawback of this method is the "black-box"–character of these software packages. The source code is obviously not available and detailed description of the used algorithms is sparse. Furthermore, this kind of software is designed for a multitude of applications and special properties of the two-stroke engine cannot be taken into account. A further disadvantage is the error tolerance of these products. Occurring errors are generally ignored and not taken as an indication that something went wrong. The debugging possibilities are therefore very limited and one is left with a calculated flow pattern without any idea of convergence.

Although it has to be admitted that these commercial codes have made quite an impressive progress during the last few years, the user still has to be careful how to interpret the obtained results. Usually, some comparison of characteristics of the predicted flow and measurements is undertaken, but whether the accuracy of the simulation, e.g. the number of degrees of freedom, itself is sufficient is rarely checked. Until now no software is able to obtain an accurate solution for all flows. So it is always vital to analyze the obtained results and study the effects of approximations one had to introduce in order to be able to solve the underlying system of equations. If, for example, one tries to simulate the Navier–Stokes equations and is not using an appropriate mesh with sufficient boundary refinement, one will not obtain the desired flow structure. And if one does not know about the approximations made it is difficult to estimate the accuracy of the result.

> Industrial users of commercial CFD codes should especially be careful, as the optimism of salesmen is legendary. J.H. Ferziger, M. Perić [8]

On the other hand there are lots of very refined methods being developed to analyze fluid flow behavior in very idealistic settings. Many research codes have emerged, and have been thoroughly tested, to solve flow problems very efficiently on simple geometries for model problems. These techniques have also been analyzed on a theoretical mathematical basis. Methods like local adaptive time-stepping, local adaptive grid refinement and parallelization have been used to speed up calculations. Higher order schemes have been constructed to achieve a higher accuracy of the simulation.

The software `engine_flow` which we developed is an approach to apply these new and refined methods of fluid simulation to the complex, real-world problem of the two-stroke engine.

2 The Two-Stroke Engine

2.1 The Different Parts of the Two-Stroke Engine

The two-stroke engine consists of several ports and chambers (see Fig. 1). The ports (inlet, transfer, and exhaust ports) are used to channel the flow from one chamber to the next. They are opened and closed by the motion of the piston, which acts as valves. In the chamber below the piston (the crankcase), the fresh gas is compressed to enable the transfer into the cylinder above the piston, where the combustion and power process takes place.

2.2 Description of the Cycle of Operation

To clarify the processes taking place in a two-stroke engine, we will follow the gas flow all the way through the engine.

The fresh gas coming from the carburator enters the crankcase via the opened inlet port due to the upward motion of the piston, which lowers the pressure inside the crankcase below the atmospheric value (Fig. 2(a)).

Then the fresh charge is compressed in the crankcase as the inlet port is closed by the lower edge of the piston (Fig. 2(b)).

As soon as the piston has moved far enough downwards the transfer ports are opened and the fresh charge enters the cylinder since the pressure in the crankcase exceeds the cylinder pressure (Fig. 2(c)). In the cylinder the fresh gas displaces the burnt exhaust gas, referred to as the *scavenge process*. This is a critical point in the two-stroke process. If the transfer ports are badly directed then a part of the fresh gas can exit from the cylinder directly into the exhaust port (called "short-circuiting"). Therefore, this fresh gas is lost for the next combustion and contributes heavily on the emission rate of unburnt hydrocarbons.

On its upward motion, the piston again closes the transfer and exhaust ports, and the trapped gas is compressed in

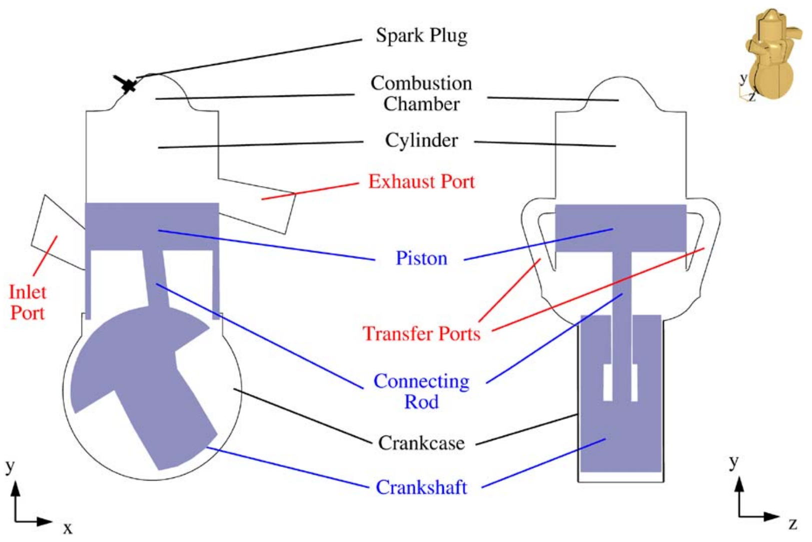


Fig. 1 Components of a two-stroke engine

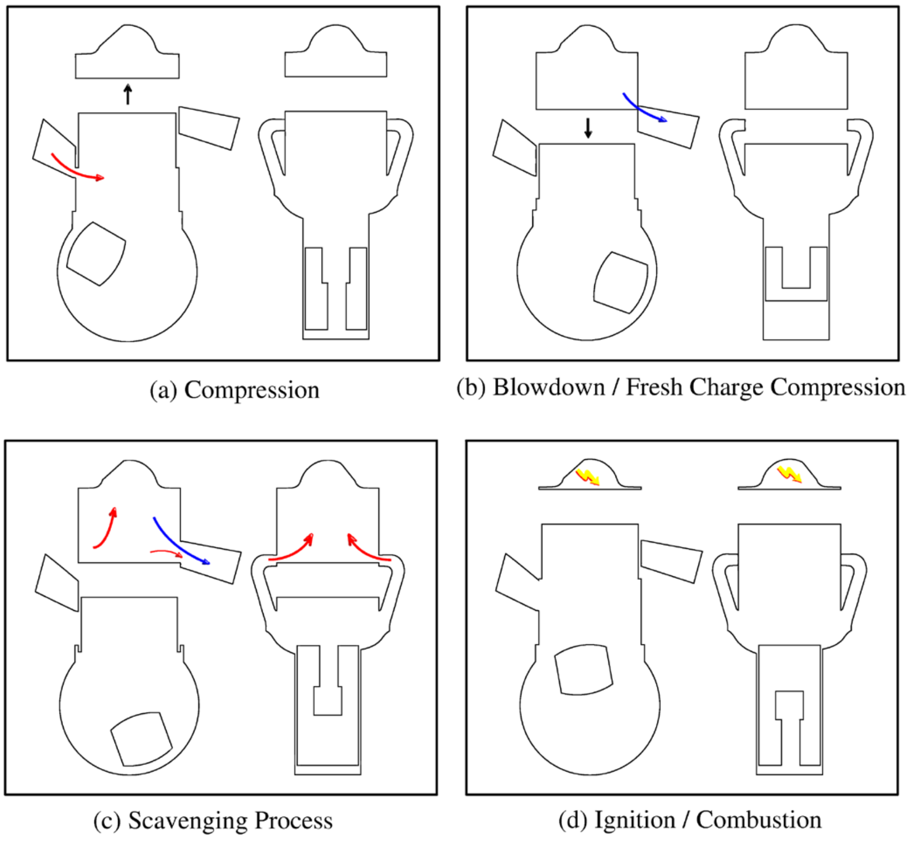

(a) Compression
(b) Blowdown / Fresh Charge Compression
(c) Scavenging Process
(d) Ignition / Combustion

Fig. 2 Different phases in the cycle of a two-stroke engine

the cylinder (*compression stroke*). Shortly before the piston reaches its uppermost position (Fig. 2(d)) the combustion begins with the ignition of the spark plug, producing a rapid rise in temperature and pressure driving the piston down on the *power stroke*.

When the exhaust port is opened, the still over-pressured hot exhaust gas is released into the exhaust port as a pressure shock-type wave and from there into the silencer (Fig. 2(b)).

On opening of the transfer ports the exhaust gas is replaced by the fresh charge (Fig. 2(c)). The reflection of the pressure wave of the initial pulse of exhaust gas at the tapered shape of the exhaust pipe can be used to push the fresh gas, which has already arrived at the exhaust port, back into the cylinder (not shown in Fig. 2). However, this is difficult to achieve when there is not much space available for an extended exhaust pipe.

2.3 Important Characteristics

To be able to measure the scavenge process quantitatively, one needs to introduce some characteristic values.

Definition 2.1 (*SR*) The *scavenge ratio SR* is a function of m_{f}, the mass of fresh air supplied during the scavenge process, and $m_{\mathrm{ev}} := \rho_{\mathrm{at}}(V_{\mathrm{s}} + V_{\mathrm{c}})$, the mass of air to fill the entire cylinder volume (under atmospheric conditions),

$$SR := \frac{m_{\mathrm{f}}}{m_{\mathrm{ev}}}.$$

Definition 2.2 (*TE*) The *trapping efficiency TE* is the ratio of the mass of fresh air that has been captured m_{trf} to that of the air supplied m_{f}:

$$TE := \frac{m_{\mathrm{trf}}}{m_{\mathrm{f}}}.$$

For a comparison of the scavenging process of two engines it is favorable to use the *SR*–*TE* chart. Particularly, the short-circuit (at low *SR* levels) is evident here (see also diagrams derived from the numerical simulation in the results section).

For details concerning this section see also [7].

3 The Mathematical Model

3.1 The Navier–Stokes Equations

Now we can proceed to state the fundamental systems of partial differential equations that describe a compressible fluid.

3.1.1 The Euler Equations

From the conservation of mass, momentum and energy of a compressible inviscid fluid (where we neglect heat conduction) the *Euler equations of gas dynamics* can be derived (see also [8, 9]). In Eulerian coordinates they can be written in the following conservative form:

$$\frac{\partial \mathbf{U}}{\partial t} + \sum_{j=1}^{3} \frac{\partial}{\partial x_j}\mathbf{f}_j(\mathbf{U}) = \mathbf{0}, \qquad \mathbf{x} = (x_1, x_2, x_3) \in \mathbb{R}^3,\ t > 0, \tag{3.1}$$

where the vector of conservative variables is

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho v_1 \\ \rho v_2 \\ \rho v_3 \\ e \end{pmatrix}, \quad e = \rho E,$$

with $\mathbf{U}$ taking values in the set of states

$$\Psi = \left\{(\rho, \rho\mathbf{v} = (\rho v_1, \rho v_2, \rho v_3), e);\ \rho > 0, \mathbf{v} \in \mathbb{R}^3, e - \frac{\rho}{2}|\mathbf{v}|^2 > 0\right\}, \tag{3.2}$$

and the *convective flux functions*

$$\mathbf{f}_1(\mathbf{U}) = \begin{pmatrix} \rho v_1 \\ \rho v_1^2 + p \\ \rho v_1 v_2 \\ \rho v_1 v_3 \\ (e+p)v_1 \end{pmatrix}, \quad \mathbf{f}_2(\mathbf{U}) = \begin{pmatrix} \rho v_2 \\ \rho v_1 v_2 \\ \rho v_2^2 + p \\ \rho v_2 v_3 \\ (e+p)v_2 \end{pmatrix}, \quad \mathbf{f}_3(\mathbf{U}) = \begin{pmatrix} \rho v_3 \\ \rho v_1 v_3 \\ \rho v_2 v_3 \\ \rho v_3^2 + p \\ (e+p)v_3 \end{pmatrix}. \tag{3.3}$$

In order to close the system an additional *equation of state* for the pressure p has to be provided (see Sect. 3.3).

The Euler equations are a hyperbolic symmetrizable nonlinear system of conservation laws (refer to [10]).

3.1.2 The Navier–Stokes Equations

If we also take the viscosity and the thermal conductivity of the fluid into account, the Euler equations (3.1) are replaced by the *Navier–Stokes equations*:

$$\frac{\partial \mathbf{U}}{\partial t} + \sum_{j=1}^{3} \frac{\partial}{\partial x_j}\mathbf{f}_j(\mathbf{U}) - \sum_{j=1}^{3} \frac{\partial}{\partial x_j}\mathbf{h}_j = \mathbf{0}, \qquad \mathbf{x} = (x_1, x_2, x_3) \in \mathbb{R}^3, t > 0, \tag{3.4}$$

with $\mathbf{U}$ and the convective flux functions $\mathbf{f}_j$ as in (3.1), and the *viscous flux functions*

$$\mathbf{h}_j\left(\mathbf{U}, \frac{\partial \mathbf{U}}{\partial x_1}, \frac{\partial \mathbf{U}}{\partial x_2}, \frac{\partial \mathbf{U}}{\partial x_3}\right) = \begin{pmatrix} 0 \\ \tau_{j1} \\ \tau_{j2} \\ \tau_{j3} \\ \sum_{l=1}^{3} \tau_{jl} v_l + \lambda \frac{\partial T}{\partial x_j} \end{pmatrix}, \qquad 1 \le j \le 3,$$

with

$$\tau_{ij} = 2\eta D_{ij}(\mathbf{U}) - \frac{2}{3}\eta\delta_{ij}\operatorname{div}\mathbf{v},$$

$$D_{ij}(\mathbf{U}) = \frac{1}{2}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right),$$

the deformation tensor $D(\mathbf{U}) := (D_{ij}(\mathbf{U}))$, the temperature T, the dynamic shear viscosity η, and the thermal conductivity λ.

To close the system (3.4) it is necessary to add two equations of state for p and T (see Sect. 3.3).

3.2 Initial Conditions and Boundary Conditions

The time interval $\mathcal{T}$ and the region Ω, in which the flow problem is considered, are described next.

3.2.1 Initial Conditions

We want to observe the temporal development of the flow. Therefore, we have to start at an initial instant $T_{\text{start}} := 0$. The physical problem that describes this procedure is given by the following

Definition 3.1 (Initial value problem) Let $\mathbf{U}$, $\mathbf{f}_j$, and $\mathbf{h}_j$ be defined as in Sect. 3.1. Then the following problem is called *initial value problem* or *Cauchy problem*:

Find a *weak entropy solution* $\mathbf{U}(\mathbf{x},t)$ such that for $T_{\text{end}} \in \mathbb{R}^+$

$$\begin{cases} \dfrac{\partial}{\partial t}\mathbf{U} + \displaystyle\sum_{i=1}^{3} \dfrac{\partial}{\partial x_i}\mathbf{f}_i(\mathbf{U}) + \sum_{i=1}^{3} \dfrac{\partial}{\partial x_i}\mathbf{h}_i(\mathbf{U}) = \mathbf{0} \\ \quad \text{in } \mathbb{R}^3 \times (0, T_{\text{end}}), \\ \mathbf{U}(\mathbf{x},0) = \mathbf{U}_0(\mathbf{x}) \quad \text{in } \mathbb{R}^3 \end{cases} \tag{3.5}$$

with initial conditions $\mathbf{U}_0 \in L^\infty(\Omega)$ given.

Remark 3.2 With the help of *characteristics* it can be shown that in general there exists no classical solution of (3.6). The concept of weak solutions has to be introduced. These are unique only if they satisfy an additional property, the entropy condition. For further details refer to [10] and [11].

3.2.2 Boundary Conditions

If we have an open finite region $\Omega \subset \mathbb{R}^3$, which is occupied by the medium under consideration, additional boundary conditions have to be specified.

Definition 3.3 (Initial boundary value problem) Let $\mathbf{U}$, $\mathbf{f}_j$, and $\mathbf{h}_j$ be defined as in Sect. 3.1. Then the *initial boundary value problem* can be stated as:

Find a *weak entropy solution* $\mathbf{U}(\mathbf{x},t)$ such that for open finite $\Omega \subset \mathbb{R}^3$ and $T_{\text{end}} \in \mathbb{R}^+$

$$\begin{cases} \dfrac{\partial}{\partial t}\mathbf{U} + \displaystyle\sum_{i=1}^{3} \dfrac{\partial}{\partial x_i}\mathbf{f}_i(\mathbf{U}) + \sum_{i=1}^{3} \dfrac{\partial}{\partial x_i}\mathbf{h}_i(\mathbf{U}) = \mathbf{0} \\ \quad \text{in } \Omega \times (0, T_{\text{end}}), \\ \mathbf{U}(\mathbf{x},0) = \mathbf{U}_0(\mathbf{x}) \quad \text{in } \Omega, \\ \mathbf{U}(\mathbf{x},t) = \mathbf{g}(\mathbf{x},t) \quad \text{on } \Gamma \subseteq \partial\Omega \times (0, T_{\text{end}}) \end{cases} \tag{3.6}$$

with initial conditions $\mathbf{U}_0 \in L^\infty(\Omega)$ and boundary conditions $\mathbf{g} \in L^\infty(\partial\Omega \times (0, T_{\text{end}}))$ given.

It can be shown that the initial boundary value problem with a boundary condition respecting outgoing characteristics admits a weak solution (cf. [12] or [13]).

3.2.3 The Different Types of Boundary Conditions

It follows a description of the different physical boundary conditions which are encountered in our problem. These boundary conditions arise mainly from physical considerations as no practical rules have emerged so far from mathematical investigations concerning existence theorems of the initial boundary value problem of the Navier–Stokes equations.

But first the notion of *Dirichlet* and *Von Neumann conditions* has to be introduced. The specification of an explicit value at the boundary is said to be a *Dirichlet boundary condition*. For example, if the velocity on a solid wall is imposed: $v = 0$. Whereas with a *Von Neumann boundary condition* a normal derivative is prescribed. For example, the heat flux at an adiabatic wall: $\frac{\partial T}{\partial n} = 0$.

- *Inflow boundary:* This is an artificial boundary condition, since the real fluid extends beyond this boundary. But as the calculation domain has to be finite, this boundary, where the flow enters the numerical domain, is introduced.
  The boundary conditions are generally determined by the physical experiment.
- *Outflow boundary:* The same is true for the outflow boundary condition where the flow leaves the numerical domain.
- *Solid slip wall:* The normal velocity of the fluid at the wall is equal to the velocity of the wall in normal direction: $\mathbf{v} \cdot \mathbf{n} = \mathbf{v}_{\text{wall}} \cdot \mathbf{n}$ with $\mathbf{n}$ the normal of the wall. Therefore, we impose a zero mass flux at this wall.
  Different types of temperature treatment are possible:
  - *adiabatic boundary condition*: There is no heat flux present between wall and fluid: $\frac{\partial T}{\partial n} = 0$.
  - *isothermal boundary condition*: The temperature of the wall is T_{wall}. This temperature is imposed as boundary condition: $T = T_{\text{wall}}$.
- *Solid no-slip wall:* Here also the tangential velocity component of the fluid is determined. Thus, the velocity of the fluid at the wall is equal to the velocity of the wall: $\mathbf{v} = \mathbf{v}_{\text{wall}}$. Also, with this wall condition no mass flux is possible.
  Because of the induced friction at the wall a *boundary layer* appears, where strong gradients of velocity and temperature are present (cf. [14]).
  Concerning the temperature, the same adiabatic or isothermal condition as in the case of the slip wall can be applied.

- *Symmetry boundary:* This is also an artificial boundary condition. It is used mainly for saving computational time by cutting the numerical domain in half and assuming a symmetric flow pattern. Thus, the flow has to be symmetric at this boundary which implies that the normal component of the velocity vanishes: $\mathbf{v} \cdot \mathbf{n} = 0$.
  For all other variables of the flow Von Neumann conditions of zero flux apply.

Remark 3.4 Symmetry boundary conditions are not unproblematic when simulating a viscous flow. Even at perfect symmetric conditions a completely unsymmetrical flow pattern can emerge as e.g. in the case of the Karman vortex street, or, at higher Reynolds numbers, in the attempt to capture the larger turbulent structures of the flow. Thus, careful testing has to be conducted before using this kind of boundary condition.

3.3 The Equations of State

3.3.1 *The Thermal Equation of State*

An ideal gas satisfies the laws of Charles and Gay–Lussac ($\frac{V}{T} = \text{const}$ with n, $p = \text{const}$), Boyle–Mariotte ($pV = \text{const}$ with n, $T = \text{const}$), and the Avogadro principle ($\frac{V}{n} = \text{const}$ with T, $p = \text{const}$), where V is the volume, T the temperature, n the total number of moles of molecules, and p the pressure of the gas. From these laws the *ideal gas law* or *thermal equation of state* $pV = n\mathcal{R}T$ can be deduced. $\mathcal{R}$ is the (molar) universal gas constant $\mathcal{R}$. With the *specific gas constant* $R = \frac{\mathcal{R}}{M}$ in $\frac{\text{J}}{\text{kg K}}$ (M being the *molecular weight* of the gas under consideration) and the density $\rho = \frac{1}{\tau}$ with the *specific volume* $\tau = \frac{V}{nM}$ the thermal equation of state can be written as

$$p = \rho RT. \tag{3.7}$$

One says that the ideal gas is thermally perfect (or perfect).

3.3.2 *The Caloric Equation of State*

The specific heat capacity $c_\text{v}(T)$ is defined by $c_\text{v}(T) = \frac{d\varepsilon}{dT}$ or $d\varepsilon = c_\text{v}(T)dT$ with specific internal energy ε. By integrating the last equation one obtains

$$\varepsilon = \int_{T_0}^{T} c_\text{v}(\tau)d\tau + \varepsilon_0$$

with constants T_0 and ε_0. As these constants are in our setting physically not of interest, one can set $T_0 = 0$ and $\varepsilon_0 = 0$. The resulting *caloric equation of state* for a calorically imperfect gas is thus given by

$$\varepsilon = \int_{0}^{T} c_\text{v}(\tau)d\tau. \tag{3.8}$$

For a calorically perfect gas, also called polytropic gas, c_v is independent of the temperature T, the caloric equation of state can then be stated as

$$\varepsilon = c_\text{v}T. \tag{3.9}$$

An equivalent formulation is given with help of (3.7):

$$p = (\gamma - 1)\rho\varepsilon \tag{3.10}$$

with $\gamma := \frac{R}{c_\text{v}} + 1 = \frac{c_p}{c_\text{v}}$.

3.4 Gas Mixtures

3.4.1 *Notations*

In a multi-component system of N different species the *total number of moles* is given by

$$n_m := \sum_{i=1}^{N} n_i$$

with n_i being the number of moles of each species. Hence, the *mole fractions* are

$$y_i := \frac{n_i}{n_m}, \quad i = 1, 2, \ldots, N \text{ with } \sum_{i=1}^{N} y_i = 1.$$

The *total mass* of the mixture is similarly defined as

$$m_m := \sum_{i=1}^{N} m_i,$$

where $m_i := M_i n_i$ is the mass of species i and M_i the components' molecular weight. Therefore, the *mass fractions* are

$$z_i := \frac{m_i}{m_m}, \quad i = 1, 2, \ldots, N \text{ with } \sum_{i=1}^{N} z_i = 1.$$

3.4.2 *The Equations of State*

In the case of non-reacting mixtures of gases *Dalton's law* states that

$$p = \sum_i p_i,$$

the total pressure is the sum of the partial pressures of each gas in the mixture. With this result the thermal equation of state (3.7) becomes

$$p = \sum_i y_i \rho\mathcal{R}T \tag{3.11}$$

with y_i the mole fraction of component i.

The heat capacity c_{vm} of a mixture of ideal gases follows the exact mixing formula (see Sect. 3.5)

$$c_{vm} = \sum_i z_i c_{vi}$$

with the heat capacity c_{vi} of component i. Therefore, the caloric equation of state (3.8) is transformed to

$$\varepsilon = \int_0^T \sum_i z_i c_{vi}(\tau) d\tau. \tag{3.12}$$

In the setting of the two-stroke engine, we have to deal with air and exhaust gas, which are themselves mixtures of nitrogen N_2, oxygen O_2, hydrogen H_2, carbon monoxide CO, carbon dioxide CO_2 and water steam H_2O.

Remark 3.5 We have stated above that the constant ε_0 is not of physical interest. In our case, where the different gases do not interact, this remains true for gas mixtures. The constant ε_0 would be, e.g., necessary for reacting gases or phase transitions.

3.4.3 Fresh Gas, Exhaust Gas

In our application of the two-stroke engine we use the aforementioned gas mixture formulas for modeling the fresh gas, assumed to have the properties of air, and the exhaust gas. Thus, we have to deal with two different species of gas (cf. also [15]). Let z be the mass fraction of fresh gas then $1-z$ is the mass fraction of the exhaust gas. With σ and τ defined as

$$\sigma := z\rho \quad \text{and} \quad \tau := (1-z)\rho$$

the Navier–Stokes equations (see Sect. 3.1) are extended to

$$\frac{\partial \mathbf{U}}{\partial t} + \sum_{j=1}^{3} \frac{\partial}{\partial x_j}\mathbf{f}_j(\mathbf{U}) - \sum_{j=1}^{3} \frac{\partial}{\partial x_j}\mathbf{h}_j = \mathbf{0},$$

$$\mathbf{x} = (x_1, x_2, x_3) \in \mathbb{R}^3,\ t > 0, \tag{3.13}$$

$$\mathbf{U} = \begin{pmatrix} \sigma \\ \rho \\ \rho v_1 \\ \rho v_2 \\ \rho v_3 \\ e \end{pmatrix},$$

$$\mathbf{f}_1(\mathbf{U}) = \begin{pmatrix} \sigma v_1 \\ \rho v_1 \\ \rho v_1^2 + p \\ \rho v_1 v_2 \\ \rho v_1 v_3 \\ (e+p) v_1 \end{pmatrix}, \qquad \mathbf{f}_2(\mathbf{U}) = \begin{pmatrix} \sigma v_2 \\ \rho v_2 \\ \rho v_1 v_2 \\ \rho v_2^2 + p \\ \rho v_2 v_3 \\ (e+p) v_2 \end{pmatrix},$$

$$\mathbf{f}_3(\mathbf{U}) = \begin{pmatrix} \sigma v_3 \\ \rho v_3 \\ \rho v_1 v_3 \\ \rho v_2 v_3 \\ \rho v_3^2 + p \\ (e+p) v_3 \end{pmatrix},$$

$$\mathbf{h}_j\left(\mathbf{U}, \frac{\partial \mathbf{U}}{\partial x_1}, \frac{\partial \mathbf{U}}{\partial x_2}, \frac{\partial \mathbf{U}}{\partial x_3}\right) = \begin{pmatrix} 0 \\ 0 \\ \tau_{j1} \\ \tau_{j2} \\ \tau_{j3} \\ \sum_{l=1}^{3} \tau_{jl} v_l + \lambda \frac{\partial T}{\partial x_j} \end{pmatrix},$$

$$1 \le j \le 3,$$

with

$$\tau_{ij} = \eta\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) - \frac{2}{3}\eta \delta_{ij} \operatorname{div} \mathbf{v}.$$

3.5 Temperature Dependency of the Material Properties

The material properties, namely the heat capacity at constant volume $c_v(T)$, the heat capacity at constant pressure $c_p(T)$, the adiabatic exponent $\gamma(T) := \frac{c_p(T)}{c_v(T)}$, the dynamic viscosity $\eta(T)$, and the thermal conductivity $\lambda(T)$, are temperature-dependent.

There are two possibilities of calculating the constants: one can interpolate tabulated values or evaluate an explicit formula.

For the *heat capacity* c_p and c_v it is difficult to find a formula which is accurate enough, on the other hand tabulated data is readily available (see e.g. [16]). So in this case interpolations of tabulated data for c_p is used. For mixtures of ideal gases the exact mixture formula

$$c_{pm} := \sum_i z_i c_{pi}$$

with mass fraction z_i is applied. The heat capacity at constant volume c_v can be deduced with the above cited equation.

For the *viscosity* η and the *thermal conductivity* λ empiric formulas recommended in [16] are used.

4 The Numerical Scheme

The Euler equations (3.1) and Navier–Stokes equations (3.4) are analytically solvable for only a limited number of special cases. For more complex applications the solutions have to be approximated by the use of a *discretization method*. Usually, for the space discretization a mesh is fitted to the geometry and on each cell of the mesh an algebraic system of equations for the values of the unknowns at the mesh points is solved. Also, in time-dependent problems, the time interval of interest is discretized into several small time-steps.

Furthermore, the equations themselves have to be discretized. The most important methods to approach the differential equations by a system of algebraic equations are the *finite difference* (*FD*), *finite element* (*FE*), and *finite volume* (*FV*) methods. As FD methods are especially suited for structured meshes they are ruled out for our application, which consists of a complex geometry that cannot be handled by a structured mesh. The FV methods are known to be very well suited for the treatment of conservation laws. This is founded in the fact that they ensure the conservation property, which is the underlying principle of the system of equations (3.1), at the discrete level by discretizing the integral form of the conservation laws. Furthermore, they take full advantage of arbitrary meshes. Therefore, the FV approach is used here.

4.1 The Mesh Structure

Here we treat the three-dimensional case.

Remark 4.1 The case of curvilinear element faces with non-constant outer normals $\boldsymbol{\nu}_{jl}$ and $\mathbf{n}_{jl}$ is covered in Sect. 5.3.

Definition 4.2 (Unstructured conformal mesh) Let $\mathcal{I} \subset \mathbb{N}$ be a finite set of indices. The set

$$\mathcal{T} := \{T_i | T_i \text{ is a compact polyhedron for } i \in \mathcal{I}\},$$

is called an *unstructured conformal mesh* of $\Omega \subset \mathbb{R}^3$, if

1. $\Omega = \bigcup_{i \in I} T_i$,
2. $\forall T_i, T_j \ (i \neq j)$ we have
   - $T_i \cap T_j = \emptyset$ or
   - $T_i \cap T_j$ is a common vertex, edge, or face of T_i and T_j.[1]

Remark 4.3 In a conformal mesh no hanging knots, edges, or faces are possible. With local refinement of a grid consisting of hexahedrons, such hanging faces occur. Therefore, we need the following definition.

Definition 4.4 (Unstructured non-conformal mesh) In an *unstructured non-conformal mesh* constraint (2) in Definition 4.2 reduces to

$2'$. $\forall T_i, T_j \ (i \neq j)$ we have $\overset{\circ}{T}_i \cap \overset{\circ}{T}_j = \emptyset$.

[1]With vertex, edge, and face defined in the obvious way (cf. also [17]).

4.2 The Finite Volume Scheme

We want to derive the finite volume scheme of first order for the initial boundary value problem (3.6).

Remark 4.5 The scheme will be deduced for the Navier–Stokes equations (3.4). An extension to the equation system governing the gas mixture of fresh-gas and exhaust-gas (3.13) is straightforward.

For the motivation of the finite volume scheme we assume that $\mathbf{U}$ and the functions $\mathbf{f}$ and $\mathbf{h}$ of (3.6) are sufficiently smooth. We start our motivation with the integral form of (3.6)

$$\frac{d}{dt}\int_{T_j} \mathbf{U} + \int_{\partial T_j} \mathbf{f}(\mathbf{U}) \cdot \mathbf{n} - \int_{\partial T_j} \mathbf{h}(\mathbf{U}) \cdot \mathbf{n} = \mathbf{0} \tag{4.1}$$

which one obtains from the system of Navier–Stokes equations (3.4) by integration over a control volume T_j and then applying the integral theorem of Gauss.

Now, for a cell-centered[2] first order scheme we define for each element $T_j \, (j \in \mathcal{I})$ and for each time t the average values

$$\mathbf{U}_j(t) := \frac{1}{|T_j|}\int_{T_j} \mathbf{U}(\cdot, t)$$

and a grid function

$$\mathbf{U}_h(\mathbf{x}, t) := \mathbf{U}_j(t) \quad \text{for } \mathbf{x} \in T_j.$$

This we use to approximate (4.1) with

$$\begin{aligned}\frac{d}{dt}\mathbf{U}_j(t) &= -\frac{1}{|T_j|}\left(\int_{\partial T_j} \mathbf{f}(\mathbf{U}_h(\cdot, t)) \cdot \mathbf{n}\right.\\ &\qquad \left. - \int_{\partial T_j} \mathbf{h}(\mathbf{U}_h(\cdot, t)) \cdot \mathbf{n}\right)\\ &= -\frac{1}{|T_j|}\left(\sum_{l=1}^{k_j}\int_{S_{jl}} \mathbf{f}(\mathbf{U}_h(\cdot, t)) \cdot \mathbf{n}_{jl}\right.\\ &\qquad \left. - \sum_{l=1}^{k_j}\int_{S_{jl}} \mathbf{h}(\mathbf{U}_h(\cdot, t)) \cdot \mathbf{n}_{jl}\right).\end{aligned}$$

The grid function $\mathbf{U}_h$ is not necessarily continuous over the cell boundaries, therefore, the flux functions $\mathbf{f}(\mathbf{U}_h)$ and $\mathbf{h}(\mathbf{U}_h)$ are not well defined on S_{jl} and have to be approximated by *numerical flux functions* $\mathbf{g}_{jl}$ and $\mathbf{G}_{jl}$ respectively (see Sect. 4.2.2, and Sects. 4.3 and 4.4).

[2]The data is located in the cell center and not on the vertices.

4.2.1 The Time Derivative

The *time derivative* is an ordinary differential equation and can be approximated by a first order explicit Euler scheme (see [11]).

Let $\mathbf{U}_j^n$ be the approximation of the exact solution $\mathbf{U}(\mathbf{x}, t)$ at time level t^n with $n \in \mathbb{N}_0$ such that

$$\mathbf{U}_j^0 := \frac{1}{|T_j|} \int_{T_j} \mathbf{U}_0(\cdot),$$
$$\mathbf{U}_j^{n+1} := \mathbf{U}_j(t) \quad \text{for } t^n < t \leq t^{n+1}.$$

It follows for the grid function $\mathbf{U}_h$:

$$\mathbf{U}_h(\mathbf{x}, t) = \mathbf{U}_j^{n+1} \quad \text{for } \mathbf{x} \in T_j,\ t^n < t \leq t^{n+1}.$$

The explicit Euler scheme is then given by

$$\mathbf{U}_j^{n+1} = \mathbf{U}_j^n - \frac{\Delta t^n}{|T_j|} \left(\sum_{l=1}^{k_j} \int_{S_{jl}} \mathbf{g}_{jl}\big(\mathbf{U}_j^n, \mathbf{U}_{jl}^n\big) \cdot \mathbf{n}_{jl} - \sum_{l=1}^{k_j} \int_{S_{jl}} \mathbf{G}_{jl}\big(\mathbf{U}_j^n, \mathbf{U}_{jl}^n\big) \cdot \mathbf{n}_{jl} \right)$$

with $t^{n+1} = t^n + \Delta t^n$ and $\mathbf{g}_{jl}$, $\mathbf{G}_{jl}$ defined in Sect. 4.2.2.

4.2.2 The Numerical Fluxes

The numerical flux functions $\mathbf{g}_{jl}$ and $\mathbf{G}_{jl}$ are approximations of the integral of the flux functions $\mathbf{f}(\mathbf{U}_h)$ and $\mathbf{h}(\mathbf{U}_h)$ over the face S_{jl}. These fluxes are influenced by the values on both sides of the face S_{jl}, therefore, the numerical fluxes $\mathbf{g}_{jl}$ and $\mathbf{G}_{jl}$ depend on the approximate solution $\mathbf{U}_j$ and $\mathbf{U}_{jl}$:

$$\mathbf{g}_{jl}\big(\mathbf{U}_j^n, \mathbf{U}_{jl}^n\big) \approx |S_{jl}| \mathbf{f}(\mathbf{U}_h(\mathbf{z}_{jl}, t^n)) \cdot \mathbf{n}_{jl} \approx \int_{S_{jl}} \mathbf{f}(\mathbf{U}_h(\cdot, t^n)) \cdot \mathbf{n}_{jl},$$
$$\mathbf{G}_{jl}\big(\mathbf{U}_j^n, \mathbf{U}_{jl}^n\big) \approx |S_{jl}| \mathbf{h}\big(\mathbf{U}_h(\mathbf{z}_{jl}, t^n)\big) \cdot \mathbf{n}_{jl} \approx \int_{S_{jl}} \mathbf{h}(\mathbf{U}_h(\cdot, t^n)) \cdot \mathbf{n}_{jl},$$

where the integral over the face S_{jl} is approximated by the value at its center $\mathbf{z}_{jl}$.

For details concerning the numerical flux functions see Sects. 4.3 and 4.4.

Assumption 4.6 If $\mathbf{U}_h$ is continuous over the face S_{jl} the flux functions $\mathbf{f}(\mathbf{U}_h)$ and $\mathbf{h}(\mathbf{U}_h)$ can be evaluated and the numerical fluxes should be identical to the analytical ones. Therefore, *consistency* is required for the numerical fluxes:

$$\mathbf{g}_{jl}(\mathbf{U}, \mathbf{U}) = |S_{jl}| \mathbf{f}(\mathbf{U}) \cdot \mathbf{n}_{jl}, \qquad \mathbf{G}_{jl}(\mathbf{U}, \mathbf{U}) = |S_{jl}| \mathbf{h}(\mathbf{U}) \cdot \mathbf{n}_{jl}. \tag{4.2}$$

In order to get the *conservation property* on the discrete level, the fluxes of two neighboring cells T_i and T_j calculated from either side of the face S_{ij} need to be identical:

$$\mathbf{g}_{ij}(\mathbf{U}, \mathbf{V}) = -\mathbf{g}_{kl}(\mathbf{V}, \mathbf{U}), \qquad \mathbf{G}_{ij}(\mathbf{U}, \mathbf{V}) = -\mathbf{G}_{kl}(\mathbf{V}, \mathbf{U}) \tag{4.3}$$

with k being the index of the neighboring element, and l the local number of the corresponding edge.

Furthermore, we assume that a third property, the *local Lipschitz condition*, is satisfied by the numerical fluxes:

$$|\mathbf{g}_{jl}(\mathbf{U}, \mathbf{V}) - \mathbf{g}_{jl}(\mathbf{U}', \mathbf{V}')| \leq c_1(R) h(|\mathbf{U} - \mathbf{U}'| + |\mathbf{V} - \mathbf{V}'|),$$
$$|\mathbf{G}_{jl}(\mathbf{U}, \mathbf{V}) - \mathbf{G}_{jl}(\mathbf{U}', \mathbf{V}')| \leq c_2(R) h(|\mathbf{U} - \mathbf{U}'| + |\mathbf{V} - \mathbf{V}'|), \tag{4.4}$$

where $R \in \mathbb{R}^+$, $\mathbf{U}$, $\mathbf{V}$, $\mathbf{U}'$, $\mathbf{V}' \in B_R(0)$, $c_1(R)$ and $c_2(R)$ are constants that depend only on R, and h the mesh-size.

Now we are ready for the following

Definition 4.7 (Finite volume scheme of first order) For given initial values $\mathbf{U}_0 \in L^\infty(\mathbb{R}^3)$ let $\mathbf{U}_j^n$ be given by:

$$\mathbf{U}_j^0 := \frac{1}{|T_j|} \int_{T_j} \mathbf{U}_0(\cdot),$$
$$\mathbf{U}_j^{n+1} := \mathbf{U}_j^n - \frac{\Delta t^n}{|T_j|} \left(\sum_{l=1}^{k_j} \mathbf{g}_{jl}\big(\mathbf{U}_j^n, \mathbf{U}_{jl}^n\big) - \sum_{l=1}^{k_j} \mathbf{G}_{jl}\big(\mathbf{U}_j^n, \mathbf{U}_{jl}^n\big) \right)$$

with the numerical flux functions satisfying properties (4.2), (4.3) and (4.4).

Remark 4.8 If the face S_{jl} is a boundary of the domain a special boundary treatment, as described in Sect. 4.5, is necessary.

Remark 4.9 Up to now there are no general existence results for systems in three space dimensions. Therefore, no convergence results for numerical schemes exist in this case. But if the numerical scheme in Definition 4.7 defines a convergent sequence the limit is a weak solution of the conservation law.

4.3 The Convective Flux

For the calculation of the convective flux function $\mathbf{g} = \mathbf{g}(\mathbf{U}, \mathbf{V})$ in three space dimensions we use a one-dimensional approach.[3] The approximate solution $\mathbf{U}_j$ and $\mathbf{U}_{jl}$ of the elements T_j and T_{jl} respectively are rotated by an orthogonal matrix $R(\mathbf{n}_{jl})$ such that the outer normal $\mathbf{n}_{jl}$ is transformed into the unit vector $\mathbf{e}_1 = (1, 0, 0)^T$. These rotations $R(\mathbf{n}_{jl})$ affect only the velocity vector $\mathbf{v}$ in $\mathbf{U}_j$ and $\mathbf{U}_{jl}$, the scalar quantities density ρ and total energy e are not concerned. Now it is possible to apply a one-dimensional scheme to the rotated values

$$\hat{\mathbf{U}}_j = R(\mathbf{n}_{jl})\mathbf{U}_j, \qquad \hat{\mathbf{U}}_{jl} = R(\mathbf{n}_{jl})\mathbf{U}_{jl}$$

to calculate the numerical flux function $\mathbf{g}(\hat{\mathbf{U}}_j, \hat{\mathbf{U}}_{jl})$. This flux is then rotated back into its original orientation. Therefore, we have

$$\mathbf{g}_{jl}(\mathbf{U}_j, \mathbf{U}_{jl}) = |S_{jl}| R^{-1}(\mathbf{n}_{jl})\mathbf{g}(\hat{\mathbf{U}}_j, \hat{\mathbf{U}}_{jl}).$$

To calculate the numerical fluxes for the one-dimensional scheme mentioned above, we use the AUSMDV scheme proposed by Liou and Steffen in [18] and enhanced by Wada and Liou in [19] (see also [20, 21]). We extended this scheme to the handling of the fresh-gas component σ that we present now. As mentioned in Sect. 4.2.2, the numerical convective flux function should approximate

$$\mathbf{g}(\hat{\mathbf{U}}_j, \hat{\mathbf{U}}_{jl}) \approx \mathbf{f}(\hat{\mathbf{U}}_h) = \begin{pmatrix} \sigma u \\ \rho u \\ \rho u^2 + p \\ (e+p)u \end{pmatrix},$$

with $\mathbf{f}$ analogous to the function defined in Sect. 3.4.3 for the rotated discrete solution $\hat{\mathbf{U}}_h$, and u the rotated velocity vector $\mathbf{v}$ as described above. Now, the numerical flux function is defined by the AUSMDV scheme

$$\mathbf{g}(\hat{\mathbf{U}}_j, \hat{\mathbf{U}}_{jl}) := \begin{pmatrix} (\sigma u)_{\frac{1}{2}} \\ (\rho u)_{\frac{1}{2}} \\ (\rho u^2)^{\mathrm{AUSMDV}}_{\frac{1}{2}} + p_{\frac{1}{2}} \\ (\rho u)_{\frac{1}{2}} (\frac{e+p}{\rho})_{\frac{1}{2}} \end{pmatrix},$$

with the fresh-gas flux $(\sigma u)_{\frac{1}{2}}$ and the mass flux $(\rho u)_{\frac{1}{2}}$ given by

$$(\sigma u)_{\frac{1}{2}} := u_j^+ \sigma_j + u_{jl}^- \sigma_{jl},$$

$$(\rho u)_{\frac{1}{2}} := u_j^+ \rho_j + u_{jl}^- \rho_{jl},$$

[3]This is possible due to the rotational invariance of the Euler and Navier–Stokes equations.

where

$$u_j^+ := \begin{cases} \alpha_j \frac{(u_j + c_m)^2}{4c_m} + (1-\alpha_j)\frac{u_j + |u_j|}{2} & \text{if } |u_j| \le c_m, \\ \frac{u_j + |u_j|}{2} & \text{otherwise,} \end{cases}$$

$$u_{jl}^- := \begin{cases} -\alpha_{jl} \frac{(u_{jl} - c_m)^2}{4c_m} + (1-\alpha_{jl})\frac{u_{jl} - |u_{jl}|}{2} & \text{if } |u_{jl}| \le c_m, \\ \frac{u_{jl} + |u_{jl}|}{2} & \text{otherwise,} \end{cases}$$

$$\alpha_j := \frac{2p_j/\rho_j}{p_j/\rho_j + p_{jl}/\rho_{jl}}, \qquad \alpha_{jl} := \frac{2p_{jl}/\rho_{jl}}{p_j/\rho_j + p_{jl}/\rho_{jl}},$$

$$c_m := \max(c_j, c_{jl}).$$

$(\rho u^2)^{\mathrm{AUSMDV}}_{\frac{1}{2}}$ is defined as

$$(\rho u^2)^{\mathrm{AUSMDV}}_{\frac{1}{2}} := \frac{1}{2}(1+s)(\rho u^2)^{\mathrm{AUSMV}}_{\frac{1}{2}} + \frac{1}{2}(1-s)(\rho u^2)^{\mathrm{AUSMD}}_{\frac{1}{2}},$$

with

$$(\rho u^2)^{\mathrm{AUSMV}}_{\frac{1}{2}} := u_j^+ \rho_j u_j + u_{jl}^- \rho_{jl} u_{jl},$$

$$(\rho u^2)^{\mathrm{AUSMD}}_{\frac{1}{2}} := \begin{cases} (\rho u)_{\frac{1}{2}} u_j & \text{if } (\rho u)_{\frac{1}{2}} > 0, \\ (\rho u)_{\frac{1}{2}} u_{jl} & \text{otherwise,} \end{cases}$$

$$s := \min\left\{1, K\frac{|p_j - p_{jl}|}{\min\{p_j, p_{jl}\}}\right\}, \qquad K := 10.$$

For the pressure flux $p_{\frac{1}{2}}$ we have

$$p_{\frac{1}{2}} := p_j^+ + p_{jl}^-,$$

$$p_j^+ := \begin{cases} p_j \frac{(u_j + c_m)^2}{4c_m^2}\left(2 - \frac{u_j}{c_m}\right) & \text{if } |u_j| \le c_m, \\ p_j \frac{u_j + |u_j|}{2u_j} & \text{otherwise,} \end{cases}$$

$$p_{jl}^- := \begin{cases} p_{jl} \frac{(u_{jl} - c_m)^2}{4c_m^2}\left(2 + \frac{u_{jl}}{c_m}\right) & \text{if } |u_{jl}| \le c_m, \\ p_{jl} \frac{u_{jl} + |u_{jl}|}{2u_{jl}} & \text{otherwise.} \end{cases}$$

And finally $(\frac{e+p}{\rho})_{\frac{1}{2}}$ is defined by

$$\left(\frac{e+p}{\rho}\right)_{\frac{1}{2}} := \begin{cases} \frac{e_j + p_j}{\rho_j} & \text{if } (\rho u)_{\frac{1}{2}} > 0, \\ \frac{e_{jl} + p_{jl}}{\rho_{jl}} & \text{otherwise.} \end{cases}$$

This AUSMDV flux has the above required properties (4.2), (4.3) and (4.4), as can be seen by a direct calculation. It is a hybrid flux vector–flux difference–splitting scheme and has good shock capturing capabilities with little numerical viscosity. Therefore, it is especially adapted for the treatment of the convective flux in a simulation of the Navier–Stokes equations. For details of the scheme and numerous numerical experiments confer [19, 22].

4.4 The Viscous Flux

In order to be able to calculate the viscous flux $\mathbf{G} = \mathbf{G}(\mathbf{U}, \mathbf{V})$ we need an approximation of the derivatives of $\mathbf{U}_h$ at the face S_{jl}. Therefore, a gradient has to be extracted from the element-wise constant $\mathbf{U}_h$. This is done with a central discretization since shear stress and heat conduction correspond to diffusive effects. The advantage of a central discretization is furthermore that it meets property (4.3) and therefore is conservative.

An easy approach to calculate the discrete gradient of the numerical solution between elements T_j and T_{jl} would be to use (u being the quantity in question, either the temperature T or a component of the velocity $\mathbf{v}$)

$$\frac{\partial u}{\partial x_i} := \Delta x_i \frac{\Delta u}{|\Delta \mathbf{x}|^2}$$

with the numerical approximation of the gradient $\Delta u := (u)_j - (u)_{jl}$ and, analogously Δx_i and $\Delta \mathbf{x}$ (cf. also [23]).

This would yield a good approximation for an orthogonal mesh, i.e. $\mathbf{w}_j - \mathbf{w}_{jl} = c\mathbf{n}_{jl}, c \in \mathbb{R}$. However, in unstructured meshes for complex geometries this condition is usually violated. Therefore, it is better (cf. the simulation of the flow over a flat plate in [24, 25]) to calculate the gradient by using a hyperplane that satisfies

$$\frac{\partial u}{\partial \mathbf{t}_{jl}} = 0$$

with the tangential vector $\mathbf{t}_{jl} \cdot \mathbf{n}_{jl} = 0$.

In order to construct this hyperplane we take four linearly independent points $\mathbf{P}_{im} \in \mathbb{R}^4$ as supporting points. Let Points $\tilde{\mathbf{P}}_{11} = (x_{11}, y_{11}, z_{11})$ and $\tilde{\mathbf{P}}_{21} = (x_{21}, y_{21}, z_{21})$ be the centers of gravity of T_j and T_{jl} and two further points $\tilde{\mathbf{P}}_{i1} = (x_{i1}, y_{i1}, z_{i1}), i \in \{1, 2\}$ satisfying $(\tilde{\mathbf{P}}_{i1} - \tilde{\mathbf{P}}_{i2}) \cdot \mathbf{n}_{jl} = 0$, such that these four points do not belong to a plane $E \in \mathbb{R}^3$. Let furthermore $\mathbf{p}_{im} = (x_{im}, y_{im}, z_{im}, u_i), i, m \in \{1, 2\}$ with $u_1 = (u)_j$ and $u_2 = (u)_{jl}$. Then, the gradient $\mathbf{D}_{jl}(u)$ of the so defined hyperplane is given by

$$\mathbf{D}_{jl}(u) = \frac{1}{\det K_{jl}} \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix} \tag{4.5}$$

with

$$K_{jl} = \begin{pmatrix} x_{12} & y_{12} & z_{12} & 1 \\ x_{11} & y_{11} & z_{11} & 1 \\ x_{22} & y_{22} & z_{22} & 1 \\ x_{21} & y_{21} & z_{21} & 1 \end{pmatrix},$$

$$d_1 = \det \begin{pmatrix} (u)_j & y_{12} & z_{12} & 1 \\ (u)_j & y_{11} & z_{11} & 1 \\ (u)_{jl} & y_{22} & z_{22} & 1 \\ (u)_{jl} & y_{21} & z_{21} & 1 \end{pmatrix},$$

$$d_2 = \det \begin{pmatrix} x_{12} & (u)_j & z_{12} & 1 \\ x_{11} & (u)_j & z_{11} & 1 \\ x_{22} & (u)_{jl} & z_{22} & 1 \\ x_{21} & (u)_{jl} & z_{21} & 1 \end{pmatrix},$$

$$d_3 = \det \begin{pmatrix} x_{12} & y_{12} & (u)_j & 1 \\ x_{11} & y_{11} & (u)_j & 1 \\ x_{22} & y_{22} & (u)_{jl} & 1 \\ x_{21} & y_{21} & (u)_{jl} & 1 \end{pmatrix},$$

as can be verified by direct calculation.

With this gradient $\mathbf{D}_{jl}$ the numerical viscous flux $\mathbf{G}_{jl}$ can be constructed as an approximation of the analytical viscous flux defined in (3.4):

$$\mathbf{G}_{jl}(\mathbf{U}_j, \mathbf{U}_{jl}) := |S_{jl}| \sum_{k=1}^{3} \mathbf{H}_{jl}^k(\mathbf{U}_j, \mathbf{U}_{jl}) \cdot \mathbf{n}_{jl}$$
$$\approx \int_{S_{jl}} \mathbf{h}(\mathbf{U}_h(\cdot, t^n)) \cdot \mathbf{n}_{jl}$$

with

$$\mathbf{H}_{jl}^k(\mathbf{U}_j, \mathbf{U}_{jl}) := \begin{pmatrix} 0 \\ \tau_{k1} \\ \tau_{k2} \\ \tau_{k3} \\ \sum_{l=1}^{3} \tau_{kl}\bar{v}_l + \lambda \mathbf{D}_{jl}^k(T) \end{pmatrix},$$

$$\tau_{rs} := \eta(\mathbf{D}_{jl}^r(v_s) + \mathbf{D}_{jl}^s(v_r)) - \frac{2}{3}\eta\delta_{rs} \sum_{i=1}^{3} \mathbf{D}_{jl}^i(v_i),$$

$$r, s \in \{1, 2, 3\},$$

$\mathbf{D}_{jl}^k(T)$	gradient of temperature T over face S_{jl} in direction x_k,
$\mathbf{D}_{jl}^k(v_i)$	gradient of velocity component v_i over face S_{jl} in direction x_k,

$$\bar{v}_i := \frac{1}{2}((v_i)_j + (v_i)_{jl}).$$

Remark 4.10 Comparisons with a *reconstruction technique*, similar to the one used in the MUSCL approach for higher order schemes (cf. [11]) to obtain a linear solution function $\mathbf{U}_j$ on each element as described in [26–28], did not show improved results. To compute the gradient with this reconstruction technique is computationally more complex, thus we use the above described, more efficient, approach (4.5).

4.4.1 The Time-Dependent Integral Form of the Navier–Stokes Equations

As in Sect. 4.2 we now want to derive equations for the numerical flux in time-dependent geometries. For this we use the identity (as in [29])

$$\frac{d}{dt}\int_{V(t)}\phi(t,\mathbf{x})d\mathbf{x} = \int_{V(t)}\frac{\partial}{\partial t}\phi(t,\mathbf{x})d\mathbf{x} + \int_{\partial V(t)}\phi(t,\mathbf{x})\mathbf{w}(t,\mathbf{x})\cdot\mathbf{n} \tag{4.6}$$

with an arbitrary time-dependent volume $V(t)$, a scalar function $\phi(t,\mathbf{x})$, $\mathbf{w}(t,\mathbf{x})$ the velocity of $\partial V(t)$, and $\mathbf{n}$ the outer normal of $\partial V(t)$.

As in the motivation of the numerical scheme (cf. Sect. 4.2), we use the integral form of the system of Navier–Stokes equations (3.4), with the numerical flux functions $\mathbf{f}_1$, $\mathbf{f}_2$, and $\mathbf{f}_3$ as defined in (3.3), that is obtained by integration over a control volume T_j and then by applying the integral theorem of Gauss. But in the case of time-dependent geometries, the additional boundary term $\int_{\partial V(t)}\phi(t,\mathbf{x})\mathbf{w}(t,\mathbf{x})\cdot\mathbf{n}$ from (4.6) has to be taken into account. This boundary term can be combined with the numerical flux functions $\mathbf{f}_1$, $\mathbf{f}_2$, and $\mathbf{f}_3$, resulting in the new flux functions $\tilde{\mathbf{f}}_1$, $\tilde{\mathbf{f}}_2$, and $\tilde{\mathbf{f}}_3$. Thus, we have the integral form of the conservation laws for a time-dependent domain $V(t)$:

$$\frac{d}{dt}\int_{T_j}\mathbf{U} + \int_{\partial T_j}\tilde{\mathbf{f}}(\mathbf{U})\cdot\mathbf{n} - \int_{\partial T_j}\mathbf{h}(\mathbf{U})\cdot\mathbf{n} = \mathbf{0} \tag{4.7}$$

with

$$\tilde{\mathbf{f}}_1(\mathbf{U}) = \begin{pmatrix} \rho(v_1 - w_1) \\ \rho(v_1^2 - w_1) + p \\ \rho(v_1 v_2 - w_1) \\ \rho(v_1 v_3 - w_1) \\ (e+p)(v_1 - w_1) \end{pmatrix},$$

$$\tilde{\mathbf{f}}_2(\mathbf{U}) = \begin{pmatrix} \rho(v_2 - w_2) \\ \rho(v_1 v_2 - w_2) \\ \rho(v_2^2 - w_2) + p \\ \rho(v_2 v_3 - w_2) \\ (e+p)(v_2 - w_2) \end{pmatrix},$$

$$\tilde{\mathbf{f}}_3(\mathbf{U}) = \begin{pmatrix} \rho(v_3 - w_3) \\ \rho(v_1 v_3 - w_3) \\ \rho(v_2 v_3 - w_3) \\ \rho(v_3^2 - w_3) + p \\ (e+p)(v_3 - w_3) \end{pmatrix},$$

and $\mathbf{h}$ as defined in (3.4).

4.5 The Boundary Conditions

For the calculation of the approximate solution $\mathbf{U}_j^{n+1}$ of the next time step in element T_j the finite volume scheme relies on the numerical fluxes. These are evaluated on the basis of the values of $\mathbf{U}_j^n$ and $\mathbf{U}_{jl}^n$ on the neighboring elements. If the element T_j is situated on a boundary of the computational domain Ω there are faces S_{jl} where T_j has no neighbor. In this case we use the *method of ghost cells*. A ghost cell $\check{T}_{jl}$ of T_j is a mirror image of T_j over the face S_{jl}. Its values $\check{\mathbf{U}}_{jl}^n$ depend on the type of boundary and are either specified by a physical boundary condition or obtained by extrapolation from the element T_j itself. This ghost cell $\check{T}_{jl}$ can then be used to calculate the numerical flux over the face S_{jl} with the interior finite volume scheme as described above. Therefore, this method closes the system of discrete equations on the calculation domain. It also enables the influence of the boundary condition on the flow by the upwind flux of the numerical flux function.

As seen in Sect. 3.2 there are *actual boundary conditions* like fixed or moving walls, and *artificial boundary conditions*, like inflow or outflow conditions, where the physical domain is unbounded and which has to be truncated in order to get a finite domain. In both types of boundaries *Dirichlet* (e.g. $p = p_0$, prescribed pressure) and *Von Neumann boundary conditions* (e.g. $\frac{\partial T}{\partial n} = 0$, no heat flux) (or a mixture of these) can be prescribed:

- The numerical realization of Dirichlet boundary conditions is accomplished by an extrapolation of the boundary data into the ghost cell. For example, the pressure $\check{p}$ on the ghost cell $\check{T}_{jl}$ is calculated as

  $$\check{p} = 2p_0 - (p)_j$$

  with a prescribed pressure p_0 and the pressure $(p)_j$ from within T_j.
- Von Neumann boundary conditions are implemented as an extrapolation of the data from the element T_j. In the above example of $\frac{\partial T}{\partial n} = 0$ the temperature $\check{T}$ of the ghost cell would be given by

  $$\check{T} = (T)_j$$

  with the temperature $(T)_j$ of element T_j.

The treatment of the boundary conditions is normally not a trivial task. As stated in Sect. 3.2.2, it is usually not possible to specify conditions for all variables. Depending on the type of boundary (e.g. subsonic inflow, subsonic outflow, no-slip wall condition) only a certain number of *physical boundary conditions* can be given. The remaining variables have to be supported by *numerical boundary conditions:*

- For numerical boundary conditions for a first order scheme the values are extrapolated constantly from within the domain.

With these facts in mind we can now describe the implemented boundary conditions in detail:

4.5.1 *Inflow*

As inflow boundary condition we use a mass-flow condition. Thus, density ρ_0 and velocity $\mathbf{v}_0$ are prescribed at this boundary. As we are in the case of a subsonic flow at the inlet the last variable has to be taken from inside the domain. The temperature T is therefore treated as a numerical boundary condition. With these quantities given, p can then be calculated via the thermal equation of state and the total energy e by the caloric equation of state:

$$\check{\mathbf{U}}_{jl} = \begin{pmatrix} \rho_0 \\ \rho_0 \mathbf{v}_0 \\ e((T)_j) \end{pmatrix}.$$

4.5.2 *Outflow*

At the outflow boundary the pressure p_0 is imposed. Density ρ and velocity $\mathbf{v}$ are taken from the inside:

$$\check{\mathbf{U}}_{jl} = \begin{pmatrix} (\rho)_j \\ (\rho \mathbf{v})_j \\ e(T((\rho)_j, p_0)) \end{pmatrix}.$$

4.5.3 *Slip (Fixed)*

As this is a solid wall condition, no mass flux should be present and the normal components of the velocity have to vanish on the boundary. Depending on the thermal properties of the wall (adiabatic or isothermal) the condition for the temperature T is either a Von Neumann or a Dirichlet condition:

$$\text{adiabatic:} \quad \check{\mathbf{U}}_{jl} = \begin{pmatrix} (\rho)_j \\ (\rho)_j((\mathbf{v})_j - 2((\mathbf{v}_j) \cdot \mathbf{n}_{jl})\mathbf{n}_{jl}) \\ e((T)_j) \end{pmatrix}$$

$$\text{isothermal:} \quad \check{\mathbf{U}}_{jl} = \begin{pmatrix} (\rho)_j \\ (\rho)_j((\mathbf{v})_j - 2((\mathbf{v}_j) \cdot \mathbf{n}_{jl})\mathbf{n}_{jl}) \\ e(T_0) \end{pmatrix}$$

4.5.4 *No-Slip (Fixed)*

The difference to the slip boundary condition is that here also the tangential component of the velocity has to vanish on the boundary:

$$\text{adiabatic:} \quad \check{\mathbf{U}}_{jl} = \begin{pmatrix} (\rho)_j \\ -(\rho \mathbf{v})_j \\ e((T)_j) \end{pmatrix}$$

$$\text{isothermal:} \quad \check{\mathbf{U}}_{jl} = \begin{pmatrix} (\rho)_j \\ -(\rho \mathbf{v})_j \\ e(T_0) \end{pmatrix}$$

4.5.5 *No-Slip (Moving)*

The Moving Wall Boundary Condition can be deduced from the descriptions given in Sect. 4.4.1. As the velocity of the fluid at a no-slip boundary is given by $\mathbf{v} = \mathbf{v}_{\text{wall}}$ the equation (4.6) for the convective flux over face S_{jl} of element T_j results in

$$\int_{S_{jl}} \check{f}(\mathbf{U}_j)\mathbf{n}_{jl} = -|S_{jl}| \begin{pmatrix} 0 \\ (p)_j(\mathbf{n}_{jl})_1 \\ (p)_j(\mathbf{n}_{jl})_2 \\ (p)_j(\mathbf{n}_{jl})_3 \\ (p)_j \mathbf{n}_{jl} \cdot \mathbf{v}_{\text{wall}} \end{pmatrix} =: \mathbf{g}_{jl}(\mathbf{U}_j, \mathbf{U}_{jl})$$

with $(\mathbf{n}_{jl})_i$ the ith component of the outer normal of face S_{jl} and $\check{f}$ defined in (4.6) (cf. [27–30]).

For the viscous flux the ghost cell approach, similar to the fixed no-slip condition, is used:

$$\text{adiabatic:} \quad \check{\mathbf{U}}_{jl} = \begin{pmatrix} (\rho)_j \\ (\rho)_j(2\mathbf{v}_{\text{wall}} - (\mathbf{v})_j) \\ e((T)_j) \end{pmatrix}$$

$$\text{isothermal:} \quad \check{\mathbf{U}}_{jl} = \begin{pmatrix} (\rho)_j \\ (\rho)_j(2\mathbf{v}_{\text{wall}} - (\mathbf{v})_j) \\ e(T_0) \end{pmatrix}$$

Remark 4.11 The choice of the prescription of boundary conditions is by no means trivial, and one has to be careful not to get an ill-posed problem. Especially the interaction of inflow and outflow boundary has to be kept in mind. In the special case of subsonic inflow and outflow conditions one can impose two out of three independent variables at the inflow and, at the outflow boundary the remaining third one should be given (cf. [31, 32]).

4.6 The CFL-Condition

For the discretization of the time derivative we use an explicit first order Euler scheme.

Remark 4.12 One of the reasons why we apply an explicit scheme, is that we need a high temporal resolution for our time-dependent problem. Thus, an implicit scheme which allows us to use big time-steps would not be advantageous. Additionally, it has been shown (see e.g. [11, 33]) that for time exact simulations an explicit time discretization is more efficient than an implicit one. The reason for this is that the error accumulates at large time-steps with the implicit scheme and hence, the mesh size has to be smaller in order to attain the same numerical error as with an explicit scheme.

However, stability and convergence conditions impose restrictions on the maximum admissible time-step for an explicit scheme. This can be motivated as follows (cf. [11, 34]).

4.6.1 *The Convective Part*

Remark 4.13 An explicit finite difference scheme for a scalar conservation law in one space dimension is stable if the time-step Δt^n is restricted by the, so-called, Courant–Friedrichs–Levi condition (or CFL condition) (cf. e.g. [35]):

$$\Delta t^n < \mathrm{CFL}\frac{h}{\max_{x\in\Omega} f'(u^n(x))}$$

with minimum mesh size h and a constant CFL < 1.

Similarly, an explicit finite volume scheme for a scalar conservation law on an unstructured triangulation $\mathcal{T}$ is monotone if

$$\Delta t^n < \mathrm{CFL}\min_{T_j\in\mathcal{T}}\left\{\frac{|T_j|}{\sum_{l=1}^{3}\max\{\nu_{jl}f'(u_j^n),0\}}\right\},$$

with a constant CFL < 1 (see [36]).

Analogously, we define the time-step restriction for the convective part of our finite volume scheme (cf. [29, 33, 37]).

Definition 4.14 (Finite volume time-step restriction for the convective part) For a given constant CFL < 1 let

$$\Delta t^n_{\mathrm{conv}} := \mathrm{CFL}\min_{T_j\in\mathcal{T}}\{\Delta t^n_{j,\mathrm{conv}}\},$$

$$\Delta t^n_{j,\mathrm{conv}} := \frac{|T_j|}{\max_{l=1,\dots,k_j}\{|S_{jl}||\lambda_{jl}(\mathbf{U}_j^n)|\}},$$

with λ_{jl} $(l=1,\dots,k_j)$ defined (motivated by the eigenvalues of the system of Euler equations (3.1) (see [11])) by

$$\lambda_{jl}(\mathbf{U}_j^n) := |\mathbf{v}_j^n\cdot\mathbf{n}_{jl}| + c_j^n,$$

where $\mathbf{v}_j^n$ is the velocity of element T_j at time-step t^n and c_j^n its local speed of sound.

Remark 4.15 In other words, the time-step has to be small enough so that information can not travel further than one cell during this time-step.

4.6.2 *The Viscous Part*

As the flow in our two-stroke simulation is strongly convection dominant, the viscous part of the finite volume scheme is mainly dominated by the convective one. Thus, a CFL-like condition for the viscous discretization is not as relevant. We base our choice on the partly empirical results obtained in [38] for dual meshes in two dimensions. Applied to our situation, we get

$$\Delta t^n_{\mathrm{visc}} := \mathrm{CFL}\min_{T_j\in\mathcal{T}}\{\Delta t^n_{j,\mathrm{visc}}\},$$

$$\Delta t^n_{j,\mathrm{visc}} := \chi\frac{|T_j|^2\rho_j}{\alpha_{\mathrm{part}}\eta\sum_{l=1}^{k_j}|S_{jl}|}\quad\text{with}$$

$$\alpha_{\mathrm{part}} := \frac{\gamma_{\mathrm{part}}^{\frac{3}{2}}\kappa_{\mathrm{part}}}{c_{\mathrm{part}}\rho_{\mathrm{part}}L_{\mathrm{part}}C_{p,\mathrm{part}}},$$

where $(\cdot)_{\mathrm{part}}$ is the average of the respective value in the part of the geometry (inlet with crankcase and transfer ports, cylinder, outlet with silencer) where element T_j is situated, L_{part} is a characteristic length scale of this part of the geometry, and $\chi = \frac{1}{4}$ specifies the relative weight of the viscous to the convective part of the time-step Δt^n.

4.6.3 *The Overall Time-Step*

The time-step Δt^n is composed of the above-defined convective and viscous parts (see [26, 28]):

$$\Delta t^n := \mathrm{CFL}\min_{T_j\in\mathcal{T}}\left\{\frac{\Delta t^n_{j,\mathrm{conv}}\Delta t^n_{j,\mathrm{visc}}}{\Delta t^n_{j,\mathrm{conv}}+\Delta t^n_{j,\mathrm{visc}}}\right\}$$

with $\Delta t^n_{j,\mathrm{conv}}$ and $\Delta t^n_{j,\mathrm{visc}}$ as defined in Sects. 4.6.1 and 4.6.2 respectively.

Remark 4.16 For a more efficient, local handling of this time-step restriction see Sect. 6.1.

4.7 The General Equation of State

In the case of a (thermally and calorically, i.e. polytropic) perfect gas the Euler equations, and therefore the convective part of the Navier–Stokes equations, can be discretized with the AUSMDV Riemann solver, which was developed for this specific problem. In the case of a calorically imperfect gas (that results from taking into account the temperature-dependent material constants (cf. Sect. 3.5)), the caloric equation of state is given by (3.8) instead of (3.10). Thus, the prerequisites for the usage of the AUSMDV solver are not valid any more. This problem can be handled by the *energy relaxation scheme* first described in [39]. The idea of this scheme is based on the splitting of the internal energy $\varepsilon = \varepsilon_1 + \varepsilon_2$ in a part ε_1 that defines the polytropic equation of state $p_1 = (\gamma-1)\rho\varepsilon_1$ for a polytropic gas analogous to (3.10), and a part ε_2, which contains the disturbing nonlinearities of the caloric equation of state of the imperfect gas. The flux determined by the polytropic part of the internal energy ε_1 can be handled by the standard Riemann solver (AUSMDV). The second part ε_2 is only advected in this first step. In a second step, the relaxation step, the influence of the nonlinearities in ε_2 are taken into account.

4.7.1 The Relaxation System

The following *relaxation system* for $\lambda \in \mathbb{R}^+$ is studied

$$\begin{cases} \frac{\partial}{\partial t}\rho^\lambda + \nabla\cdot(\rho^\lambda \mathbf{v}^\lambda) = 0, \\ \frac{\partial}{\partial t}(\rho^\lambda \mathbf{v}^\lambda) + \nabla\cdot(\rho^\lambda \mathbf{v}^\lambda (\mathbf{v}^\lambda)^T + p_1^\lambda I) = 0, \\ \frac{\partial}{\partial t} e_1^\lambda + \nabla\cdot((e_1^\lambda + p_1^\lambda)\mathbf{v}^\lambda) = \lambda\rho^\lambda(\varepsilon_2^\lambda - \Phi(\rho^\lambda, \varepsilon_1^\lambda)), \\ \frac{\partial}{\partial t}(\rho^\lambda \varepsilon_2^\lambda) + \nabla\cdot(\rho\varepsilon_2^\lambda \mathbf{v}^\lambda) = -\lambda\rho^\lambda(\varepsilon_2^\lambda - \Phi(\rho^\lambda, \varepsilon_1^\lambda)), \end{cases} \tag{4.8}$$

with the total energy density

$$e_1^\lambda = \rho\left(\varepsilon_1^\lambda + \frac{|\mathbf{v}^\lambda|^2}{2}\right), \tag{4.9}$$

the polytropic equation of state $p_1^\lambda = (\gamma_1 - 1)\rho\varepsilon_1^\lambda$ with a constant $\gamma_1 > 1$ and the energy function $\Phi(\rho, \varepsilon)$. It can be shown that in the equilibrium limit $\lambda \to \infty$ the original Euler equations are obtained:

Theorem 4.17 *Consider a family of classical solutions*

$$(\rho^\lambda, \rho^\lambda \mathbf{v}^\lambda, e_1^\lambda, \rho^\lambda \varepsilon_2^\lambda)_{\lambda>0}$$

of the relaxation system (4.8), *with* $(\rho^\lambda, \rho^\lambda \mathbf{v}^\lambda, e_1^\lambda)^T \in \Psi$ *as defined in* (3.2), *that is uniformly bounded with respect to* λ. *Assume that the equilibrium limit*

$$\mathbf{U}(\mathbf{x}, t) := \lim_{\lambda\to\infty} \begin{pmatrix} \rho^\lambda \\ \rho^\lambda \mathbf{v}^\lambda \\ e_1^\lambda + \rho^\lambda \varepsilon_2^\lambda \end{pmatrix} (\mathbf{x}, t)$$

exists. Then **U** *is a solution of the Euler equations* (3.1) *if we choose*

$$\Phi(\rho, \varepsilon_1) = \varepsilon(\rho, p_1(\rho, \varepsilon_1)) - \varepsilon_1, \tag{4.10}$$

where $\varepsilon(\rho, \cdot)$ *is the inverse of* $p(\rho, \cdot)$: $p(\rho, \varepsilon(\rho, \bar{p})) = \bar{p}$ *for fixed* ρ.

For the proof of this theorem cf. [40].

In order to construct the numerical scheme for the relaxation step we need the following

Theorem 4.18 *Let the energy* Φ *be given by* (4.10) *and assume that* Φ *is monotone increasing in* ε_1. *Consider the system of ODEs*

$$\begin{cases} \frac{\partial}{\partial t}\rho^\lambda = 0, \\ \frac{\partial}{\partial t}(\rho^\lambda \mathbf{v}^\lambda) = 0, \\ \frac{\partial}{\partial t} e_1^\lambda = \lambda\rho^\lambda(\varepsilon_2^\lambda - \Phi(\rho^\lambda, \varepsilon_1^\lambda)), \\ \frac{\partial}{\partial t}(\rho^\lambda \varepsilon_2^\lambda) = -\lambda\rho^\lambda(\varepsilon_2^\lambda - \Phi(\rho^\lambda, \varepsilon_1^\lambda)), \end{cases} \tag{4.11}$$

with the initial conditions given by

$$(\rho_0, (\rho\mathbf{v})_0, (e_1)_0, (\rho\varepsilon_2)_0).$$

Denote with $(\varepsilon_1)_0$ *the internal energy of the initial data defined through the relation* (4.9). *Then the solution of* (4.11) *for* $\lambda \to \infty$ *is*

$$\left(\rho_0, (\rho\mathbf{v})_0, \rho_0\left(\varepsilon_1^* + \frac{1}{2}|\mathbf{v}_0|^2\right), \rho_0\varepsilon_2^*\right).$$

The constants ε_1^* *and* ε_2^* *are defined by the algebraic relations*

$$p(\rho_0, (\varepsilon_1)_0 + (\varepsilon_2)_0) = p_1(\rho_0, \varepsilon_1^*),$$
$$\varepsilon_1^* + \varepsilon_2^* = (\varepsilon_1)_0 + (\varepsilon_2)_0.$$

If $p_1(\rho, \varepsilon_1) = (\gamma_1 - 1)\rho\varepsilon_1$ *then* ε_1^* *and* ε_2^* *are given by the explicit relations*

$$\varepsilon_1^* := \frac{p(\rho_0, \varepsilon_0)}{(\gamma_1 - 1)\rho_0},$$
$$\varepsilon_2^* := \varepsilon_0 - \varepsilon_1^*$$

with $\varepsilon_0 := (\varepsilon_1)_0 + (\varepsilon_2)_0$.

The proof of this theorem can also be found in [40].

4.7.2 The Numerical Scheme

Now in order to use our initial AUSMDV Riemann solver the following scheme is constructed. For given left and right-hand states $\mathbf{U}_{l/r} = (\rho_{l/r}, (\rho\mathbf{v})_{l/r}, e_{l/r})^T$ we construct the numerical flux $\mathbf{g}_{jl}(\mathbf{U}_l, \mathbf{U}_r)$ consisting of a relaxation step, the employment of the standard Euler flux $\mathbf{g}_{jl}^{\text{perf}}(\tilde{\mathbf{U}}_l, \tilde{\mathbf{U}}_r)$, followed by a pure advection of the remaining internal energy ε_2. The solution of the relaxation step can be analytically defined by using Theorem 4.18

$$\varepsilon_{2,l/r} := \Theta_2(\mathbf{U}_{l/r}) := \varepsilon_{l/r} - \frac{p_{l/r}(\rho_{l/r}, \varepsilon_{l/r})}{\rho_{l/r}(\gamma_{1,l/r} - 1)},$$
$$\tilde{\mathbf{U}}_{l/r} := \boldsymbol{\Theta}_1(\mathbf{U}_{l/r}) := (\rho_{l/r}, (\rho\mathbf{v})_{l/r}, e_{l/r} - \rho_{l/r}\varepsilon_{2,l/r})^T.$$

The constant $\gamma_{1,l/r}$ has to fulfill the additional restriction

$$\gamma_{1,l/r} > \max\{\gamma(\rho_l,\varepsilon_l), \gamma(\rho_r,\varepsilon_r), \Gamma(\rho_l,\varepsilon_l), \Gamma(\rho_r,\varepsilon_r)\}$$

with $\gamma(\rho,\varepsilon) := \frac{\rho c^2(\rho,\varepsilon)}{p(\rho,\varepsilon)}$, $\Gamma(\rho,\varepsilon) := 1 + \frac{1}{\rho}\partial_\varepsilon p(\rho,\varepsilon)$ (as shown in [40]). However, in our case the definition of $\Gamma(\rho,\varepsilon)$ reduces to

$$\Gamma(\rho,\varepsilon) = 1 + \frac{R}{c_{\mathrm{v}}(T)} = \gamma(T) = \gamma(\rho,\varepsilon)$$

resulting in the attenuated restriction

$$\gamma_{1,l/r} > \max\{\gamma(\rho_l,\varepsilon_l), \gamma(\rho_r,\varepsilon_r)\}.$$

Now the convective numerical flux function $\mathbf{g}_{jl}^{\text{perf}}(\tilde{\mathbf{U}}_l, \tilde{\mathbf{U}}_r)$, i.e. the AUSMDV Riemann solver, can be used to compute the fluxes for $\tilde{\mathbf{U}} = (\rho, \rho\mathbf{v}, e_1)^T$.

Then the advection is calculated with the help of the mass flux approximation $(\mathbf{g}_{jl}^{\text{perf}})_1 \approx \rho\mathbf{v}$ on face S_{jl}. In the case that it is positive ε_2 should be advected to the left otherwise to the right:

$$\mathbf{g}_{jl}^{\varepsilon_2}(\mathbf{U}_l,\mathbf{U}_r) := \begin{cases} \mathbf{g}_{jl}^{\text{perf}}(\tilde{\mathbf{U}}_l, \tilde{\mathbf{U}}_r)\varepsilon_{2,l} & \text{for } \mathbf{g}_{jl}^{\text{perf}}(\tilde{\mathbf{U}}_l, \tilde{\mathbf{U}}_r) \geq 0, \\ \mathbf{g}_{jl}^{\text{perf}}(\tilde{\mathbf{U}}_l, \tilde{\mathbf{U}}_r)\varepsilon_{2,r} & \text{for } \mathbf{g}_{jl}^{\text{perf}}(\tilde{\mathbf{U}}_l, \tilde{\mathbf{U}}_r) < 0. \end{cases}$$

Thus, the convective numerical flux according to the energy relaxation scheme is given by

$$\mathbf{g}_{jl}(\mathbf{U}_l,\mathbf{U}_r) := \mathbf{g}_{jl}^{\text{perf}}(\tilde{\mathbf{U}}_l, \tilde{\mathbf{U}}_r) + (0, \mathbf{0}, \mathbf{g}_{jl}^{\varepsilon_2}(\mathbf{U}_l,\mathbf{U}_r)).$$

For a detailed description and analysis of this method see [40]. It is also shown that a higher efficiency is attained by using this technique as opposed to the standard FVS.

The application of this relaxation scheme to the extended finite volume scheme that accounts for the two gas mixture species fresh-gas and exhaust-gas (see Sect. 3.4.3) is straightforward.

4.7.3 *Tabularized Equation of State*

To calculate the pressure p from the vector $\mathbf{U}$ of conservative variables, the temperature T has to be evaluated from the calorical equation of state (3.8). As this temperature is only given implicitly, a tabularized version of this equation is computed, thus making $T(\varepsilon)$ available. As the heat capacity at constant volume $c_{\mathrm{v}}(T)$ is assumed to be constant for $n \leq T < n+1$, for $n \in \mathbb{Z}$, this table is easy to compute. The step size is chosen as 1 000 J. This corresponds approximately to the one degree-step size of the table $\varepsilon(T)$. These step sizes have been chosen according to the slope of the functions under consideration.

With this temperature T and the given density ρ one can then easily evaluate the thermal equation of state (3.7) in order to obtain the pressure p.

5 Special Mesh Treatment

5.1 Grid Merging

As meshes consisting of hexahedrons are better suited to resolve boundary layers we use meshes this kind of meshes. Additionally, hexahedron meshes are needed for the *Snapper* algorithm (introduced in Sect. 5.2), that is responsible for the movement of the piston. But it is more difficult to generate hexahedron meshes than tetrahedral ones, especially for complex geometries in three space dimensions. Therefore, it is usually a great benefit to assemble a mesh from different parts and "glue together" these composite meshes on their interfaces. The advantages are the facilitation of the mesh generation and the better quality of the resulting mesh due to less topological constraints on the interfaces of the different building blocks and the absence of restrictions to the shape of the ports that connect to the cylinder and the crankcase (for details refer to Sect. 9.1).

5.1.1 *The Overlapping Grid Scheme*

There are two main ideas to approach the problem of assembling grids. One is the *overlapping* (or *overset*) *grid scheme*. We have in this case two or more grids that overlap. On each grid the system of partial differential equations is solved. The values on boundary points are obtained by an interpolation procedure from the data on inner points of the other grid (see [41, 42]). Sometimes holes have to be cut out of a mesh in order to exclude these elements from the flow field calculation as described in [43]. The big drawback of this method of overlapping grids is its non-conservativeness if no special precautions are undertaken. The conservation property, as set forth in the numerical scheme (see Sect. 4.2.2), is very important, e.g. in order to capture shock waves passing through the boundary. For maintaining the conservation it is necessary to solve an additional system of equations to balance the interpolation coefficients (cf. [44]). This computational effort is avoided by using the second approach described now.

5.1.2 *The Patched Grid Scheme*

Here, the different parts of the mesh are constructed so that they share common boundary interfaces. But still the single element faces usually join discontinuously. In this situation it is possible to avoid the need for interpolation of data from the different grids. The interfaces are split into several *facets* and the flux over each facet is calculated and accounted for on either side of the facet. Therefore, we preserve the conservation property over the mesh interface without the need for any additional computation. The only effort has to be put into the calculation of the facets between two elements of the two adjacent meshes. This is described now in further detail.

5.1.3 *Matching Interfaces with the Patched Grid Scheme*

As mentioned before, the faces of the boundary elements of the two meshes under consideration generally do not match each other. Thus, a direct calculation of the fluxes is not possible. Our approach now consists in splitting up the face S_{jl} of element T_j into several facets S_{jlk} such that each of these facets is shared by two neighboring elements T_j and T_{jlk} for $0 \leq k \leq k_{jl}$ with k_{jl} being the number of neighbors of element T_j over the boundary face S_{jl}. With this method an element therefore not only has one neighbor T_{jl} per face S_{jl} anymore but k_{jl} neighbors. In order to evaluate the corresponding numerical convective and viscous fluxes g_{jlk} and G_{jlk} of these two elements T_j and T_{jlk} through their common facet S_{jlk}, the only additional information we need to calculate is the area of the common facet S_{jlk}. However, apart from this modification the standard scheme as described in Sect. 4 can be applied. As the neighbors T_{jlk} and the area of the corresponding common facets S_{jlk} can be calculated beforehand and saved in a special structure (see the next Sect. 5.1.4), this method needs very little computational time.

5.1.4 *The Structure*

When the calculation of the numerical fluxes reaches a face on an inter-mesh interface a special treatment of this face is necessary. Thus, we need a new boundary condition as an indicator of such an interface.

Then, in order to calculate these fluxes, the area of each facet S_{jlk}, its outer normal $\mathbf{n}_{jlk}$, and the pointer to the corresponding neighbor T_{jlk} are used. Therefore, this information has to be stored in the grid merging structure in the initialization phase and can then be used at each iteration step. As the number of neighbors k_{jl} of element T_j on such an interface is not bounded, a linked list is employed.

5.1.5 *Initialization*

In the initialization phase the neighbors of each inter-mesh boundary element T_j have to be detected. For the first neighbor T_{jl0} a heuristic approach is used. It is assumed that the connecting faces have a similar size. Then we want to find a boundary element T_m from a different part of the mesh that satisfies

$$|\mathbf{z}_{jl} - \mathbf{z}_{mn}|^2 < 2|S_{jl}|,$$

where $\mathbf{z}_{jl}$ is the center of gravity of S_{jl}. If such an element is found it is used as a starting point for an iterative depth search of all the neighboring elements of T_j. Depth level $d = 0$ is the starting element itself, elements of depth level $d = 1$ are the neighbors of the starting element, elements of level $d = 2$ are the neighbors of these neighbors, and so on. The common area A_{jlk} of the boundary faces of these elements with face S_{jl} is calculated. As this area calculation is a 2D problem, it can be handled by fast 2D algorithms to decide if a point is inside a face, to calculate the intersection point of two line segments, to sort vectors, and finally, to calculate the area A_{jlk} of the common facet. If $A_{jlk} > 0$ a facet S_{jlk} is created and the grid merge structure is filled with this information. The recursive search is stopped if there are no new facets S_{jlk} when advancing from depth level d to depth level $d + 1$. If

$$\sum_{k=0}^{k_{jl}} |S_{jlk}| = |S_{jl}|$$

then element T_j is an inner element and all neighbors over boundary face S_{jl} have been found. Otherwise, element T_j is situated at a corner of the composite mesh and a part of the face S_{jl} is a boundary.

5.1.6 *Time Iteration*

On each face S_{jl} of each element T_j the fluxes are calculated as described in Sect. 4. If a boundary to another part of the mesh is encountered, each facet S_{jlk} of face S_{jl} is handled in turn in the same manner as a normal face. Only $|S_{jl}|$ is replaced by $|S_{jlk}|$, $\mathbf{n}_{jl}$ by $\mathbf{n}_{jlk}$, and T_{jl} by T_{jlk}. If

$$A_{\text{bnd}} := |S_{jl}| - \sum_{k=0}^{k_{jl}} |S_{jlk}| > 0, \tag{5.1}$$

then the residual face of area A_{bnd} is calculated as a wall boundary condition with convective numerical flux

$$g_{jlk} = \begin{pmatrix} 0 \\ p\mathbf{n}_{jl} \\ 0 \end{pmatrix}$$

and viscous numerical flux G_{jl} as described in Sects. 4.4 and 4.5.

5.2 The Piston Motion

The vertical position of the piston is determined by an explicit formula derived from geometrical properties of the engine (namely *stroke* and *connecting rod length*). In order to realize this position the mesh has to be changed dynamically after each iteration step. This mesh changing affects also the connection to other parts of the grid, namely the transfer ports, the inlet and the outlet duct. This dynamic changing has to be applied in the cylinder as well as in the crankcase. In the crankcase we have the additional problem that the piston bottom is not a flat surface but has extensions that are necessary for a correct timing of the inlet duct opening.

5.2.1 *Different Techniques*

For this dynamic piston motion several methods of mesh changing have been proposed. Some of these are described here.

5.2.2 *Grid Compression*

A first approach is to compress and stretch the whole grid as the piston moves up and down, called ICED-ALE ([45]), as applied e.g. in [46] for unstructured meshes. But, as the compression ratio in two-stroke engines is usually very high, this would lead to many very small cells as the piston reaches tdc, decreasing the time step dramatically.

5.2.3 *Re-Meshing*

To solve this problem one could re-mesh the area under consideration with bigger cells. The data on these new cells would be obtained by interpolation of the old mesh (see e.g. [45, 46]). But still the whole grid must be moved in each step, which is computationally costly, and an additional interpolation from the old to the new mesh during the re-meshing process can be very complicated and time-consuming and introduces a further interpolation error. Furthermore, the generation of a new grid might not be that easy for the irregular lower surface of the piston.

5.2.4 *The Collected Cell Algorithm*

With this approach, a cutting technique (cf. [29]) intersects all elements with the piston crown. Only the part of the element above the piston is retained. If an element happens to be too small (and the CFL condition would imply too small a time step) it is merged with other neighboring cells by the *collected cell algorithm*. This method results in complex case differentiation and many different types of elements, depending on the orientation of the element that has been cut by the piston, even more so if the piston is not flat anymore. However, the advantage is that an arbitrary tetrahedral mesh can be used to discretize the cylinder and crankcase. In addition, the calculation of less and less elements in the cylinder is necessary as the piston moves its way upwards, reducing computational time for the flow simulation.

5.2.5 *The Snapper Algorithm*

The requirement of the *snapper algorithm* (see e.g. [47]) is an equal distribution of hexahedrons or prisms oriented parallel to the piston surface. If the piston moves, e.g. upwards, the lowest layer of elements in the cylinder (above the piston) is reduced in size as long as this does not result in too small elements. If too small elements occur the lowest layer of elements is merged with the next upper one, which now becomes the lowest layer. This process is reversed in the case of the downwards motion of the piston. The elements in the crankcase (below the piston) are treated in an analogous manner. The advantage of this algorithm is its easy application and therefore its low cost with respect to computational time. And as with the Collected Cell Algorithm there are always many layers of deactivated cells, which can therefore be neglected for the flow calculation.

5.2.6 *Detailed Description of the Snapper Algorithm*

Because we decided to use a hexahedral mesh, the snapper algorithm is a natural choice. It is not difficult to assure that we have a prescribed parallel orthogonal layering within the lower part of the cylinder and the upper part of the crankcase.

In the initialization step we first build a list of all layers in the cylinder and crankshaft to grant a fast and easy access to all elements of the lowest and second lowest layer at all times. Within the time iteration the exact position $h_{\text{new}} = h_p(\Theta)$ of the piston above bdc is calculated via an explicit formula. This new position h_{new} is compared to the old position h_{old}. For the following description we assume an upward motion of the piston, i.e. $h_{\text{new}} > h_{\text{old}}$. Therefore, above the piston the elements of the lowest layer have to shrink. Their lower face rises to position h_{new} and identifies the new boundary to the piston crown (see the upper part of Fig. 3) if their new relative height l_{rel} is not smaller than a certain prescribed threshold t, where l_{rel} is defined as

$$l_{\text{rel}} := \frac{l_{\text{act}}}{l_{\text{orig}}} := \frac{h_{\text{u}} - h_{\text{l,act}}}{h_{\text{u}} - h_{\text{l,orig}}},$$

with relative height l_{rel} of the lowest layer, actual height l_{act}, original height l_{orig}, h_{u} position of the upper face, $h_{\text{l,act}}$ the actual position of the lower face, and $h_{\text{l,orig}}$ the original position of the lower face. The data in these cells is then adapted according to the conservation of mass, impulse and energy:

$$u_{\text{new}} = \frac{V_{\text{old}}}{V_{\text{new}}} u_{\text{old}},$$

where u is the conserved variable ρ, $\rho\mathbf{v}$, and e.

If $l_{\text{rel}} < t$, i.e. the new cells would be too small, the lowest layer “snaps” back to its original position, is deactivated, and the lower face of the elements of the former second lowest layer is set to position h_{new}. The boundary condition of these faces is changed in order to identify the piston boundary (Fig. 3 (middle)). Now the data in the cells has to be matched to fulfill the conservation properties:

$$u_{\text{new}} = \frac{V_{\text{l,old}}}{V_{\text{new}}} u_{\text{l,old}} + \frac{V_{\text{s,old}}}{V_{\text{new}}} u_{\text{s,old}}$$

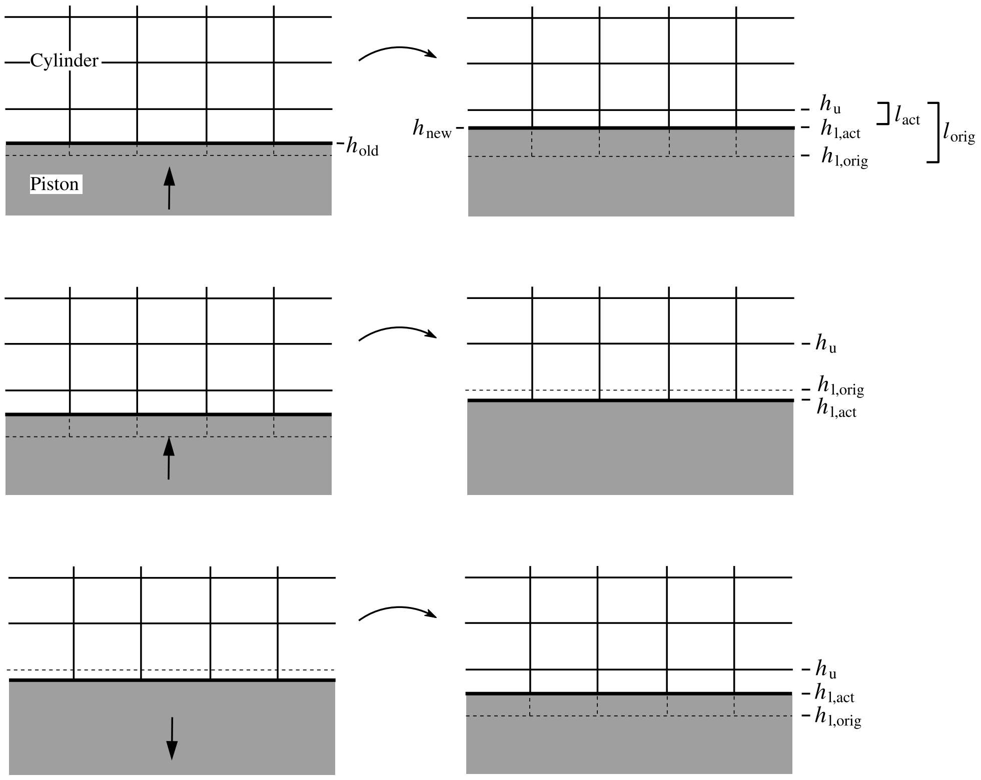


Fig. 3 The snapper algorithm, moving (*top*), merging (*middle*), and splitting (*bottom*)

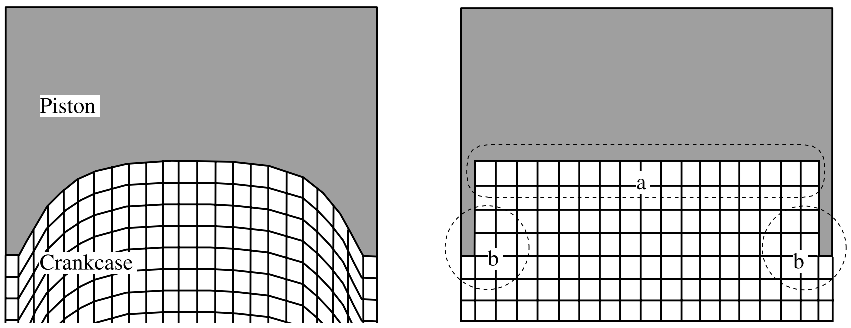


Fig. 4 Two different pistons: layering possible (*left*) and impossible (*right*)

with $(\cdot)_{\mathrm{l,old}}$ and $(\cdot)_{\mathrm{s,old}}$ the former value of the element in the lowest and second lowest layer, respectively.

In the case of downward motion of the piston the faces of the lowest elements are lowered to the new position h_{new}. The data on the elements is adjusted in the same manner as in the compression of the cells. The elements are split into two layers if the height of the new layer is larger than the threshold percentage of its original size. In this case the elements of the new layer are activated, the boundary condition is shifted down one layer, the now second lowest layer is restored to its original size, and the lower face of the just activated elements is moved to represent the piston position h_{new} (as displayed in Fig. 3 (bottom)). Also the data in the cells is split into the two new layers:

$$u_{\mathrm{l,new}} = u_{\mathrm{s,new}} = \frac{V_{\mathrm{old}}}{V_{\mathrm{l,new}} + V_{\mathrm{s,new}}} u_{\mathrm{old}}$$

5.2.7 *The Extended Snapper Algorithm*

With the above described approach the piston surface does not need to be flat as long as a layering of the elements along the piston surface is possible. However, the bottom side of the piston does not allow for this layering as one can see in Fig. 4. Therefore, the snapper algorithm has to be extended to this case. On the upper part of the mesh (area *a* on Fig. 4) the standard algorithm can be applied. Just on the lower part of the mesh (area *b* on Fig. 4) special care has to be taken of the neighboring cells, which are affected by the motion of the upper faces of the elements at the piston boundary (as seen in Fig. 5).

We have to deal with three cases to handle the data in these neighbors:

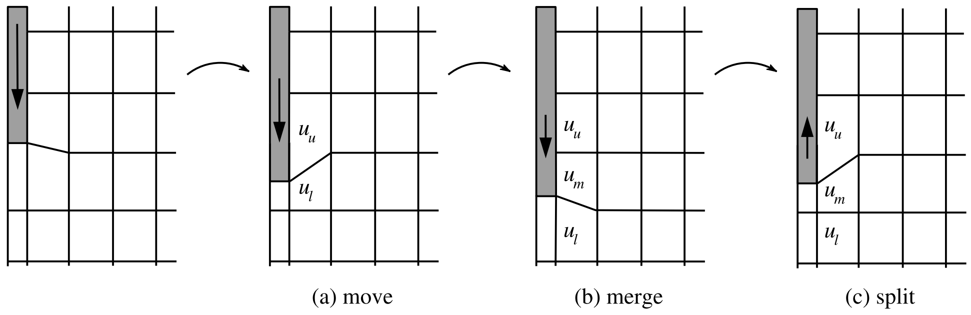


Fig. 5 The neighboring cells get affected by the piston motion

- The lowest layer elements are only changed in size, no merging or splitting occurs (case *a* in Fig. 5):

$$u_{\mathrm{u,new}} = \frac{V_{\mathrm{u,old}}}{V_{\mathrm{u,new}}} u_{\mathrm{u,old}} + \frac{V_{\mathrm{u,new}} - V_{\mathrm{u,old}}}{V_{\mathrm{u,new}}} u_{\mathrm{l,old}},$$
$$u_{\mathrm{l,new}} = u_{\mathrm{l,old}}$$

  for $u_{\mathrm{u},\cdot}$ being the conserved variable of the upper element and $u_{\mathrm{l},\cdot}$ the one for the lower element.[4] These equations, as well as the ones stated below, result from the transport flux from the lower element to the upper element to compensate for the grid motion.
- The lowest and second lowest layers are merged while the piston moves downwards (case *b* in Fig. 5):

$$u_{\mathrm{u,new}} = u_{\mathrm{u,old}},$$
$$u_{\mathrm{m,new}} = \frac{V_{\mathrm{m,old}} + V_{\mathrm{l,old}} - V_{\mathrm{l,new}}}{V_{\mathrm{m,new}}} u_{\mathrm{m,old}} + \frac{V_{\mathrm{u,old}} - V_{\mathrm{u,new}}}{V_{\mathrm{m,new}}} u_{\mathrm{u,old}},$$
$$u_{\mathrm{l,new}} = \frac{V_{\mathrm{l,old}}}{V_{\mathrm{l,new}}} u_{\mathrm{l,old}} + \frac{V_{\mathrm{l,old}} - V_{\mathrm{l,new}}}{V_{\mathrm{l,new}}} u_{\mathrm{m,old}}$$

  with $u_{\mathrm{u},\cdot}$ being the conserved variable of the upper element, $u_{\mathrm{m},\cdot}$ the one for the middle element, and $u_{\mathrm{l},\cdot}$ for the lower element.
- The lowest level is split into two layers on the upward motion of the piston (case *c* in Fig. 5):

$$u_{\mathrm{u,new}} = \frac{V_{\mathrm{u,old}}}{V_{\mathrm{u,new}}} u_{\mathrm{u,old}} + \frac{V_{\mathrm{u,new}} - V_{\mathrm{u,old}}}{V_{\mathrm{u,new}}} u_{\mathrm{m,old}},$$
$$u_{\mathrm{m,new}} = \frac{V_{\mathrm{u,old}} + V_{\mathrm{m,old}} - V_{\mathrm{u,new}}}{V_{\mathrm{m,new}}} u_{\mathrm{m,old}} + \frac{V_{\mathrm{l,old}} - V_{\mathrm{l,new}}}{V_{\mathrm{m,new}}} u_{\mathrm{l,old}},$$
$$u_{\mathrm{l,new}} = u_{\mathrm{l,old}}.$$

[4]Here we stated the equations governing the downward motion of the piston. The upward motion is handled in the same manner.

5.2.8 *The Update of the Window Area to the Ducts*

As the piston acts as a valve for the inflow, transfer, and outflow ports by covering them at certain positions, this also affects the grid merging algorithm, described in Sect. 5.1. This is due to the feature that all ports connecting to the cylinder and crankcase can be realized as self contained meshes that are connected to the cylinder and crankcase by the grid merging method. It has been stated there that the connectivity information for each element T_j on an inter-mesh boundary can be calculated at the initialization phase before the time iteration starts. Now, if T_j is a cylinder or crankcase element that changes its size by the snapper algorithm this has to be accounted for in the grid merging structure. Thus, the area of a boundary element T_j with all its neighbors T_{jlk} $(1 \le k \le k_{jl})$ over the boundary face S_{jl} has to be updated dynamically if its size was changed. But as there do not exist many boundary elements whose size is changed during a time step, and no new neighbors have to be found, this dynamic area update is not too computationally costly.

5.3 The Crankshaft Motion

The crankshaft, which rotates in the crankcase, possesses a complex three-dimensional structure. For the investigation of its influence on the flow behavior a three-dimensional simulation is therefore indispensable.

We described different merging methods for a composite mesh in Sect. 5.1. The same possibilities are given for the rotating mesh of the crankshaft within the outer mesh of the crankcase. Because of conservation issues we also decided here to use a patched grid approach. But due to the rotating motion of the crankshaft and the cylindrical shape of the mesh the straightforward implementation failed. The exhaustive search for new neighbors was too computationally expensive, and the approximation of the cylinder by planar cell faces produced too big errors because of the violation of the conservation property. Therefore, a new approach had to be found.

5.3.1 The Curved Interface Method

This method treats the interface of the two meshes as a true cylinder $\mathcal{Z} := S^1 \times [z_1, z_2]$. Thus, the faces of the boundary elements are not planar any longer but curved in one direction. This also ensures the conservation property and a constant changing of face locations is overcome. As a further advantage the outer face of the crankshaft itself is simulated with its real round shape.

To avoid the thin outer layer of elements around the crankshaft, it is admitted that the outer layer of the inner grid has holes. With this method this does not pose a problem since no points need to be projected to the faces of the inner elements. In order to handle elements with curved faces some modifications to standard elements have to be considered. We describe now the treatment for an element *in the outer mesh with inwards curved face*. The elements in the inner mesh with outwards curved faces are handled in an analogous manner.

Thus, all elements at the crankcase-crankshaft interface are adapted at initialization, as follows. The area of the curved face S_{jl_0} increases whereas the area of the front and rear face, S_{jl_f} and S_{jl_r} respectively, and the volume V_{T_j} decrease. In our case (cf. Fig. 6) these changes can be calculated easily:

$$\begin{aligned}|S_{jl_0}| &= \alpha_0 R_{cs} l_z,\\ |S_{jl_f,\text{new}}| &= |S_{jl_f,\text{old}}| - \left(\frac{\alpha_0}{2}R_{cs}^2 - \frac{d}{2}R_{cs}\cos\left(\frac{\alpha_0}{2}\right)\right),\\ |S_{jl_r,\text{new}}| &= |S_{jl_f,\text{new}}|,\\ V_{T_j} &= |S_{jl_f,\text{new}}| l_z,\end{aligned}$$

where $\alpha_0 := \alpha_{P_2} - \alpha_{P_1}$, with $\alpha_{P_{1/2}}$ as in Fig. 6 in degrees, R_{cs} the radius of the crankshaft, l_z the length of the cell in z-direction, and $d := |\mathbf{p}_2 - \mathbf{p}_1| = 2R_{cs}\sin(\frac{\alpha_0}{2})$ the distance between points P_1 and P_2. The normal $\mathbf{n}_{jl_0}$ of face S_{jl_0} is now not constant anymore but depends, in our case, i.e. the

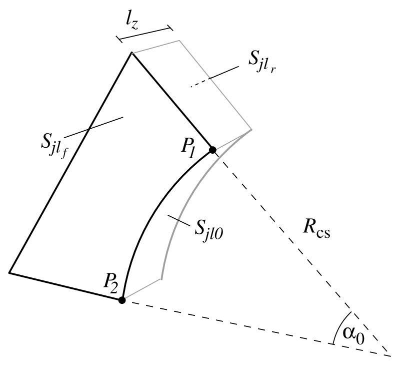


Fig. 6 Round element faces in the crankcase and crankshaft meshes

center of rotation is the origin and the axis of rotation is the z-axis, on $\alpha(\mathbf{x})$, $\mathbf{x} \in S_{jl_0}$:

$$\mathbf{n}_{jl_0}(\alpha(\mathbf{x})) = \begin{pmatrix} -\cos\alpha(\mathbf{x}) \\ -\sin\alpha(\mathbf{x}) \\ 0 \end{pmatrix}.$$

In the finite volume scheme given in Sect. 4.2 $|S_{jl}|\mathbf{n}_{jl}$ is assumed to be an approximation of $\int_{S_{jl}} \mathbf{n}_{jl}(\cdot)$ which is not correct in the case of an evaluation of the normal $\mathbf{n}_{jl_0}(\mathbf{x})$ in a single point. But because of the explicit representation of the normal $\mathbf{n}_{jl_0}(\mathbf{x})$ and the rectangular shape of S_{jl_0} an exact integration of $\mathbf{n}_{jl_0}(\mathbf{x})$ on S_{jl_0} is possible:

$$\begin{aligned}\int_{S_{jl_0}} \mathbf{n}_{jl_0}(\mathbf{x})d\mathbf{x} &= l_z R_{cs}\int_{\alpha_{P_1}}^{\alpha_{P_2}} \mathbf{n}_{jl_0}(\alpha)d\alpha\\ &= l_z R_{cs}\begin{pmatrix} -\sin\alpha_{P_2} + \sin\alpha_{P_1} \\ \cos\alpha_{P_2} - \cos\alpha_{P_1} \\ 0\end{pmatrix}.\end{aligned}$$

Now, if in the finite volume scheme we replace $\mathbf{n}_{jl_0}$ by

$$\begin{aligned}\bar{\mathbf{n}}_{jl_0} &:= \frac{1}{|S_{jl}|}\int_{S_{jl}} \mathbf{n}_{jl_0}(\mathbf{x})d\mathbf{x}\\ &= \frac{1}{\alpha_0}\begin{pmatrix} -\sin\alpha_{P_2} + \sin\alpha_{P_1} \\ \cos\alpha_{P_2} - \cos\alpha_{P_1} \\ 0\end{pmatrix}\end{aligned} \tag{5.2}$$

the standard algorithm can be applied to treat such elements with curved faces.

Lemma 5.1 *For the curved face S_{jl_0} let $\bar{\mathbf{n}}_{jl_0}$ be given by* (5.2) *and $\bar{\mathbf{n}}_{jl} = \mathbf{n}_{jl}$ for $l \neq l_0$. Then*

$$\sum_{l=1}^{k_j} |S_{jl}|\bar{\mathbf{n}}_{jl} = \mathbf{0}. \tag{5.3}$$

Proof It is sufficient to show that $|S_{jl_0}|\bar{\mathbf{n}}_{jl_0} = |\tilde{S}_{jl_0}|\tilde{\mathbf{n}}_{jl_0}$ with $\tilde{S}_{jl_0}$ being the planar face and $\tilde{\mathbf{n}}_{jl_0}$ its constant outer normal vector. Then the proposition follows from the according property for elements with planar faces.

The extension in z-direction of S_{jl_0} and $\tilde{S}_{jl_0}$ is identical. Therefore, we only have to show that

$$l_R\bar{\mathbf{n}}_{jl_0} = d\tilde{\mathbf{n}}_{jl_0} \tag{5.4}$$

with $l_R := \frac{2\pi}{360}\alpha_0 R_{cs}$ the arc length from P_1 to P_2 and $d := |\mathbf{p}_2 - \mathbf{p}_1| = 2R_{cs}\sin(\frac{\alpha_0}{2})$ the distance between P_1 and P_2. Equation (5.4) is true since

$$R_{cs}\begin{pmatrix}-\sin\alpha_{P_2}+\sin\alpha_{P_1}\\ \cos\alpha_{P_2}-\cos\alpha_{P_1}\\ 0\end{pmatrix}$$
$$=2R_{cs}\sin\left(\frac{\alpha_{P_2}-\alpha_{P_1}}{2}\right)\begin{pmatrix}-\cos(\frac{\alpha_{P_2}+\alpha_{P_1}}{2})\\ -\sin(\frac{\alpha_{P_2}+\alpha_{P_1}}{2})\\ 0\end{pmatrix},$$

which can be seen by applying the addition theorems for sine and cosine. □

Remark 5.2 The property (5.3) is a necessary requirement for the convergence of the finite volume method. The following example illustrates this fact.

Assuming constant data $\mathbf{U}$ for element T_j and all neighbors T_{jl} $(1 \le l \le k_j)$, one would expect the data to stay constant: $\mathbf{U}_j^{n+1} = \mathbf{U}_j^n$. However, in the finite volume scheme we obtain

$$\begin{aligned}\mathbf{U}_j^{n+1} &= \mathbf{U}_j^n - \frac{\Delta t^n}{|T_j|}\sum_{l=1}^{k_j}\mathbf{g}_{jl}(\mathbf{U},\mathbf{U})\\ &= \mathbf{U}_j^n - \frac{\Delta t^n}{|T_j|}\sum_{l=1}^{k_j}|S_{jl}|\mathbf{f}(\mathbf{U})\cdot\mathbf{n}_{jl}\\ &= \mathbf{U}_j^n - \frac{\Delta t^n}{|T_j|}\mathbf{f}(\mathbf{U})\cdot\sum_{l=1}^{k_j}|S_{jl}|\mathbf{n}_{jl},\end{aligned}$$

thus if $\mathbf{f}(\mathbf{U})$ does not vanish (i.e. pressure $p > 0$) and (5.3) does not hold then $\mathbf{U}_j^{n+1} \neq \mathbf{U}_j^n$.

Now we have described the changes to the boundary elements to handle the curved surface at the initialization phase. On the inner mesh the orientation of the normals has to be updated after every movement of the cells during the time iteration but the sizes of the faces and the volume of the elements stays the same as well as all properties on the outer mesh. What we also have to consider is the possible changing of neighbors while the inner mesh rotates.

5.3.2 *Exploitation of the Quasi 2D Structure*

In order to speed up the neighbor search, a new structure was created that exploits the quasi 2D structure of the crankshaft-crankcase interface. A pointer to each element of the boundary is stored such that an element on the cylinder $\mathcal{Z} := S^1 \times [z_1, z_2]$ can be accessed by its position on the circle S^1 and its layer n_z. A search for an element is then performed by first looking for an element whose range on the S^1 is admissible, starting with the first element to the left of a former neighbor.[5] If this is found the z-range is checked for the elements in the correct layers. The worst-case complexity of this algorithm is $O(N)$ as opposed to $O(N^2)$ of the naive neighbor search. Because the inner mesh can have holes (the crankshaft itself) not every element in the outer mesh has a full set of neighbors. If a neighbor T_{jlk} of element T_j is found the area of the common facet S_{jlk} has to be determined. This is straightforwardly solved as

$$\begin{aligned}|S_{jlk}| &= l_z(T_j,T_{jlk})l_\alpha(T_j,T_{jlk}),\\ l_z(T_j,T_{jlk}) &:= \min\left\{\max_{i\in\mathcal{K}_j}\{(P_i)_z\},\ \max_{i\in\mathcal{K}_{jlk}}\{(P_i)_z\}\right\}\\ &\quad - \max\left\{\min_{i\in\mathcal{K}_j}\{(P_i)_z\},\ \min_{i\in\mathcal{K}_{jlk}}\{(P_i)_z\}\right\},\\ l_\alpha(T_j,T_{jlk}) &:= \min\left\{\max_{i\in\mathcal{K}_j}\{(P_i)_\alpha\},\ \max_{i\in\mathcal{K}_{jlk}}\{(P_i)_\alpha\}\right\}\\ &\quad - \max\left\{\min_{i\in\mathcal{K}_j}\{(P_i)_\alpha\},\ \min_{i\in\mathcal{K}_{jlk}}\{(P_i)_\alpha\}\right\},\end{aligned}$$

with l_z being the interval of the overlap of T_j and T_{jlk} in z direction, and l_α the interval of the overlap on the circle S^1, where $\mathcal{K}_j$ is an index set such that P_i $(i \in \mathcal{K}_i)$ is a vertex of element T_j, and $(P_i)_z$ is the z coordinate of the point P_i, and $(P_i)_\alpha$ is its α coordinate with respect to the S^1 used above. Now, all entities of the grid merging structure are filled and the finite volume scheme can be calculated as usual.

5.3.3 *Compensation of the Artificial Transport*

Because of the rotation of the elements in the crankshaft mesh, also the data on these cells is rotated. However, this is unphysical as the rotation of a mesh without obstacles should leave the flow unchanged. Therefore, this artificial transport has to be compensated for. This is done by calculating a transport of all data in the opposite direction of rotation (motivated by the time-dependent formulation of the integral form of the Navier–Stokes equations in Sect. 4.4.1).

Thus, starting from the standard conservation law of transport

$$\int_{T_j}\frac{\partial}{\partial t}\mathbf{U} - \int_{\partial T_j}\mathbf{v}_{\text{grid}}\cdot\mathbf{n}\,\mathbf{U} = \mathbf{0},$$

$$\mathbf{v}_{\text{grid}} = 2\pi\frac{\text{rpm}}{60}\begin{pmatrix}x_2\\ -x_1\\ 0\end{pmatrix},$$

with rpm being the rotations per minute of the crankshaft mesh, we get for the discrete explicit upwind finite volume scheme of first order (cf. Sect. 4.2)

$$\mathbf{U}_j^{n+1} = \mathbf{U}_j^n - \frac{\Delta t_n}{|T_j|}\sum_{l=1}^{k_j}\tilde{\mathbf{g}}_{jl}(\mathbf{U}_j^n,\mathbf{U}_{jl}^n)$$

[5] The inner mesh rotates counter clockwise, therefore, the first element to the left has the highest probability of being the sought element.

with

$$\tilde{\mathbf{g}}_{jl}(\mathbf{U}_j^n, \mathbf{U}_{jl}^n) = \begin{cases} |S_{jl}|\mathbf{v}_{\text{grid}} \cdot \mathbf{n}_{jl}\mathbf{U}_{jl}^n & \text{if } \mathbf{v}_{\text{grid}} \cdot \mathbf{n}_{jl} \geq 0, \\ |S_{jl}|\mathbf{v}_{\text{grid}} \cdot \mathbf{n}_{jl}\mathbf{U}_j^n & \text{if } \mathbf{v}_{\text{grid}} \cdot \mathbf{n}_{jl} < 0, \\ \mathbf{0} & \text{if } S_{jl} \text{ is wall boundary.} \end{cases}$$

Therefore, we can write

$$\mathbf{U}_j^{n+1} = \mathbf{U}_j^n - \sum_{l=1}^{k_j} \frac{\mathbf{v}_{\text{grid}} \cdot \mathbf{n}_{jl}\Delta t^n |S_{jl}|}{|T_j|} \times \begin{cases} \mathbf{U}_{jl}^n & \text{if } \mathbf{v}_{\text{grid}} \cdot \mathbf{n}_{jl} \geq 0, \\ \mathbf{U}_j^n & \text{if } \mathbf{v}_{\text{grid}} \cdot \mathbf{n}_{jl} < 0, \\ \mathbf{0} & \text{if } S_{jl} \text{ is wall boundary.} \end{cases}$$

Now, $V_{\text{swept}} := \mathbf{v}_{\text{grid}} \cdot \mathbf{n}_{jl}\Delta t^n |S_{jl}|$ is the volume swept by the face S_{jl} from time-step n to time-step $n+1$ as shown in Fig. 7. This volume V_{swept} can be calculated easily if one stores the coordinates of the mesh points from the last time-step. As we assumed a layered mesh, the front and back faces of the elements are parallel to the x–y axis, therefore, $\mathbf{v}_{\text{grid}} \cdot \mathbf{n}_{jl} = 0$ for such faces S_{jl}. They do not account for any transport terms. Summarizing, our resulting compensation scheme, combined with the finite volume scheme for the discretization of the Navier–Stokes equations, can be calculated as

$$\mathbf{U}_j^{n+1} = \mathbf{U}_j^n - \frac{\Delta t^n}{|T_j|}\left(\sum_{l=1}^{k_j} \mathbf{g}_{jl}(\mathbf{U}_j^n, \mathbf{U}_{jl}^n) - \sum_{l=1}^{k_j} \mathbf{G}_{jl}(\mathbf{U}_j^n, \mathbf{U}_{jl}^n)\right) - \sum_{\substack{l=1 \\ \mathbf{v}_{\text{grid}}\cdot\mathbf{n}_{jl}\neq 0}}^{k_j} \frac{V_{\text{swept}}}{|T_j|} \begin{cases} \mathbf{U}_{jl}^n & \text{if } \mathbf{v}_{\text{grid}} \cdot \mathbf{n}_{jl} > 0, \\ \mathbf{U}_j^n & \text{if } \mathbf{v}_{\text{grid}} \cdot \mathbf{n}_{jl} < 0, \\ \mathbf{0} & \text{if } S_{jl} \text{ is wall boundary,} \end{cases}$$

or equivalently

$$\bar{\mathbf{U}}_j^n = \mathbf{U}_j^n - \sum_{\substack{l=1 \\ \mathbf{v}_{\text{grid}}\cdot\mathbf{n}_{jl}\neq 0}}^{k_j} \frac{V_{\text{swept}}}{|T_j|} \begin{cases} \mathbf{U}_{jl}^n & \text{if } \mathbf{v}_{\text{grid}} \cdot \mathbf{n}_{jl} > 0, \\ \mathbf{U}_j^n & \text{if } \mathbf{v}_{\text{grid}} \cdot \mathbf{n}_{jl} < 0, \\ \mathbf{0} & \text{if } S_{jl} \text{ is wall boundary,} \end{cases}$$

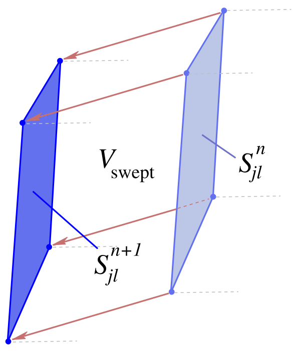


Fig. 7 Swept volume of a face S_{jl} at crankshaft rotation

$$\mathbf{U}_j^{n+1} = \bar{\mathbf{U}}_j^n - \frac{\Delta t^n}{|T_j|}\left(\sum_{l=1}^{k_j} \mathbf{g}_{jl}(\mathbf{U}_j^n, \mathbf{U}_{jl}^n) - \sum_{l=1}^{k_j} \mathbf{G}_{jl}(\mathbf{U}_j^n, \mathbf{U}_{jl}^n)\right),$$

with $\mathbf{g}_{jl}$ and $\mathbf{G}_{jl}$ being the convective and viscous numerical fluxes of the Navier–Stokes discretization as defined in Sects. 4.3 and 4.4. (For the treatment of moving grids compare also [8].)

5.3.4 The Crankshaft-Transfer Port Interface

Another problem that arises, is the interface between the rotating crankshaft mesh and the transfer port mesh. So far, we just discussed the search and area computation for the crankshaft-crankcase interface. As there is no special grid structure to profit from, the standard grid merging algorithm, as described in Sect. 5.1, is used. It is just extended by the ability to update the facet area of active neighbors, to look for new neighbors starting from the neighbor on the leading edge of the rotating mesh, and to respect the adapted area of the round faces. Especially the latter extension needs further consideration. As the frontal face S_{jl_f} of an element with rounded face S_{jl_0} is bigger (for a cell in the inner mesh) or smaller (for a cell in the outer one), this has to be kept in mind at the calculation of the common area of the facet $S_{jl_f k}$ (see Lemma 5.1 and Remark 5.2). Thus, the arc segment of the frontal face S_{jl_f} has to be calculated and intersected with the element faces of the neighbors T_{jl_f} in the transfer port mesh. This calculation is of basic geometrical nature and is straightforward.

6 Reducing Computational Time

In computational fluid dynamics computational time is always a topic. A mesh size that would be small enough to capture even the smallest feature of the flow field is still far beyond reach. In three dimensions, this problem is even intensified as the number of mesh cells rises with the order of N^3 with N being the number of cells in one dimension. As one is also interested in an accurate solution with respect to the time scale, this also implies a restriction on the time step, which results in an increased number of time steps for a fixed time interval. This number also increases with the order N, resulting in a total complexity of the simulation of $O(N^4)$.

With the techniques described in the following, it is possible to reduce this cost in the temporal as well as in the spacial respect and keep the high accuracy of the fine mesh and small time-step.

6.1 Temporally Consistent Adaptive Local Time-Stepping

Especially in dealing with complex geometries it is sometimes inevitable to have a mesh consisting of elements of largely differing size. The time-step Δt^n from the finite volume scheme as given in Definition 4.7 depends, for stability reasons as stated in Sect. 4.6, on the data $\mathbf{U}_j$ and element sizes of T_j. To assure the temporal consistency of the elements among themselves the smallest time-step $\Delta t^n := \min_{j\in\mathcal{I}}\{\Delta t_j^n\}$ of all elements is chosen. Thus, small elements in the mesh force the time-step to be very small.

The idea of the *adaptive local time-stepping* is now to guarantee this temporal consistency by a different mechanism (see also [48, 49]).

6.1.1 The Time-Step Level

Therefore, the time iteration is split up into time cycles of fixed duration. At the beginning of each cycle, at time t^{n_0}, the elements are sorted into a time level table according to their actual $\Delta t_j^{n_0}$ as follows. The element T_j is in time level l_j if its time-step $\Delta t_j^{n_0}$ is in the slice

$$2^{l_l}\Delta t_{\min} \le \Delta t_j^{n_0} < 2^{l_j+1}\Delta t_{\min}$$

with $0 \le l_j \le l_{\max}$ and $\Delta t_{\min} := \min_{j\in\mathcal{T}}\{\Delta t_j^{n_0}\}$. Thus, at the beginning of the time cycle the values for $\Delta t_{\min}$ and $l_{\max}$ are calculated. Within the cycle, all elements belonging to time level $l_{\text{timestep}} = 0$ are calculated each time-step with $\Delta t = \Delta t_{\min}$, the elements of level $l_{\text{timestep}} = 1$ every second time-step with $2\Delta t_{\min}$, the elements of level $l_{\text{timestep}} = 2$ every fourth one with $4\Delta t_{\min}$, and so on. During this time stepping process the minimum time-step $\Delta t_{\min}$ is fixed, but the classification of the Δt_j^n of element T_j to its time level l_j is checked after each update of element T_j. I.e. if after the calculation of T_j the new Δt_j^{n+1} is found to have decreased below the bound $2^{l_j}\Delta t_{\min}$ its time level l_j is reduced by one, and on the other hand if $\Delta t_j^{n+1} \ge 2^{l_j+1}\Delta t_{\min}$ the time level l_j is increased by one. If the time level l_j of element T_j has to be decreased below level $l_{\text{timestep}} = 0$ a new level $l_{\text{timestep}} = -1$ is created. Thus, the next time-step is conducted for the level $l_{\text{timestep}} = -1$ with $\Delta t = \frac{1}{2}\Delta t_{\min}$ and the restriction on l_j has to be replaced by $l_{\min} \le l_j \le l_{\max}$. The time level l_j can only be raised by one when the elements of time level $l_{\text{timestep}} = l_j + 1$ have been updated in this step, because only then is element T_j consistent with the other elements T_i being in this higher time level.

By this method it is assured that every element is updated by the finite volume scheme with a Δt that is smaller than or equal to the Δt required by the CFL-condition. Furthermore, the temporal link between the elements is respected, since after the time cycle all elements advance to

$$t^n = 2^{l_{\max}}\Delta t_{\min} + t^{n_0} = (n - n_0)\Delta t_{\min} + t^{n_0},$$

due to the duration of a complete time cycle of $2^{l_{\max}}\Delta t_{\min}$.

Remark 6.1 It has to be noted that, in order to save computational time, the flux between two neighboring elements T_j and T_{jl} is only computed once, since it is required to be identical for conservation reasons. In addition to the speedup of the calculation, this is a crucial property for the local time stepping method to work. The reason being that after an update of the element T_j with time level $l_j < l_{jl}$ the data $\mathbf{U}_j^{n-1}$ of time-step $n-1$ would be lost for the computation of the flux of element T_{jl} with T_j. Thus a complete history of data $\mathbf{U}^{\tilde{n}}$ for $n_0 \le \tilde{n} \le n$ would need to be stored, which is impossible in the case of large grids. Due to the one sided flux calculation, only data of not yet updated elements is needed which implies that only $\mathbf{U}^{n-1}$ and $\mathbf{U}^n$ have to be stored.

6.1.2 Forced Update of an Element

It can occur that an element, or all elements, have to be updated. For instance, if an element has to be refined or coarsened (see the next Sect. 6.2) or if the data on all elements is written into a file. Then, the element under consideration has to be calculated up to the actual time t^n. Therefore, it has to be determined when the element T_j was last updated, which is basic calculus knowing the time level l_j of the element, the minimum level $l_{\min}$, the starting point of the time cycle n_0, and the actual step n. This enables us to find the number of steps n_j and the corresponding $\Delta t_j = n_j\Delta t_{\min}$ to update the element using the standard finite volume scheme. Now the data $\mathbf{U}_j$ of the element can be accessed. The new time level l_j of the element is set to $l_j = l_{\min}$ and is adaptively raised during the next time-steps.

6.1.3 Time Level Restrictions

Not all elements can be in every time level. If the data of an element is changed by a mesh movement the element has to be updated at this moment. Therefore, if the geometry is adapted in each time level l_{geom} to the actual situation, then this would also be the maximum level to be in for the elements affected by the geometry movement, i.e. all elements involved in the extended snapper algorithm of the piston (described in Sect. 5.2) and all elements on the connecting crankcase-crankshaft interface. This restriction has to be verified each time an element tries to increase its time level due to a large enough Δt_j^n.

6.1.4 Elements in the Rotating Crankshaft Mesh

In order to not restrict the time level for all elements situated in the rotating crankshaft mesh, and therefore reducing the efficiency of the local time-stepping algorithm, a special treatment of the normals of these elements is necessary. As said in Sect. 5.3, the normals of every element in the crankshaft mesh are updated according to the rotational position of the crankshaft. Thus, if an element is calculated only every n_jth time-step with corresponding time level $l_j > l_{\text{geom,crank}}$ ($l_{\text{geom,crank}}$ being the update level of the crankshaft mesh rotation) it has to use average normals

$$\bar{\mathbf{n}}_{jl} := \frac{\mathbf{n}_{jl}^{n_{\text{old}}} + \mathbf{n}_{jl}^{n}}{2}$$

with the normal n_{old} of the last update of element T_j. Because of the one-sided update of the flux with the neighbors T_{jl} of T_j, as noted in Sect. 6.1.1, this procedure does not violate the conservation property, and also Lemma 5.1 is satisfied.

6.2 Dynamic Local Mesh Adaption

The idea behind the local mesh adaption is to approximate the solution of the considered problem with an equal distribution of the numerical error to reach a given accuracy with minimum computational cost. Thus, in areas where the approximation error is large the mesh should be refined, and where the error is small we can calculate on a coarse mesh to save resources. In the case of a solution evolving in time this adaption has to be dynamic.

Also, in the case of the dynamic mesh adaption it was necessary to extend the standard method for the handling of the moving meshes and the complex geometry.

6.2.1 Different Approaches

In the literature, it is distinguished between adaption techniques that result in conformal meshes, i.e. meshes without hanging edges (cf. Definitions 4.2 and 4.4) and nonconformal meshes.

6.2.2 Conformal Adaption

One possibility is a re-meshing of the whole domain (as done in [50]) with a prescribed refinement in the areas indicated by the adaption criterion (see Sect. 6.2.7). This approach is ruled out for complex three-dimensional hexahedral meshes, since an automatic mesh generation is not feasible in this case.

Another technique consists in moving the points of the discretization (cf. [51]) according to the adaption criterion, resulting in small elements where more accuracy is needed and large ones in areas of already good approximation. The number of elements stays constant in this approach.

The difficulty in these two approaches consists in the interpolation of the data $\mathbf{U}_j^n$ for the changed elements. As with the re-meshing and compression algorithms to realize the piston motion (cf. Sect. 5.2.1) this can be very complex and time consuming.

6.2.3 Non-Conformal Adaption

Therefore, a local adaption procedure is used. A local refinement of a hexahedral mesh always implies the presence of nonconforming (hanging) edges, as opposed to a tetrahedral one (see [52, 53]). If indicated by the adaption criterion an element of the mesh is refined by splitting it up into several smaller child elements. For coarsening an element this process is reversed. Thus, a local interpolation of the data is possible (see Sect. 6.2.6).

6.2.4 The Mesh Structure

A structure has to be devised which is able to handle the difficulty of non-conformal meshes. Therefore, the local mesh adaption was realized with an underlying hierarchical mesh structure (compare [54–57]). The macro grid consists of the initially generated mesh. Each macro element can now be refined into child elements, which themselves can be further refined. Also, a coarsening is possible which is necessary for time-dependent problems.

6.2.5 Geometrical Procedure

On refining, in our case, a hexahedron, it is divided into eight child hexahedrons, where the new vertices of the children are generated by the arithmetic mean of the vertices of the parent hexahedron, i.e. the new vertex P_{12} on the edges between vertices P_1 and P_2 of the parent element is given by $\mathbf{p}_{12} = \frac{\mathbf{p}_1+\mathbf{p}_2}{2}$, the new vertex on the face by $\mathbf{p}_{1234} = \frac{1}{4}\sum_{i=1}^{4} \mathbf{p}_i$, and the volume vertex by $\mathbf{p}_{1\dots8} = \frac{1}{8}\sum_{i=1}^{8} \mathbf{p}_i$. Each element is situated in a refinement level l^{ref} depending on the depth in the hierarchical mesh. Macro elements are in refinement level $l^{\text{ref}} = 0$, their children in level $l^{\text{ref}} = 1$, and so forth. Elements of level $l^{\text{ref}} > 0$ can be coarsened again if they do not have any child elements. The refinement level difference of neighboring cells is restricted to one. Therefore, in the case of hexahedrons the maximum number of neighbors T_{jlk} of element T_j per side S_{jl} is four: $k \in \{1, \dots, 4\}$. This is the case if the neighboring element is on refinement level $l_{jl} = l_j + 1$. That implies that if an element has to be refined all its neighbor elements have to be on the same or a higher level: $l_{jl} \geq l_j$ for $1 \leq l \leq k_j$, k_j being the number of faces of element T_j. If this precondition is not fulfilled then all neighboring elements with lower level have to be refined themselves. This has to be recursively iterated until an admissible state is reached. Inversely an element can only be coarsened if none of its neighbors has a higher refinement level: $l_{jl} \leq l_j$ $(1 \leq l \leq k_j)$.

6.2.6 Prolongation and Restriction of the Data

At refinement the data $\mathbf{U}_j^{\text{parent}}$ of a parent element is just constantly prolonged to the child elements

$$\mathbf{U}_j^{\text{child } i} := \mathbf{U}_j^{\text{parent}} \quad (1 \le i \le 8).$$

This is obviously conservative since $|T_j| = \sum_{i=1}^{8} |T_j^{\text{child } i}|$.

When coarsening an element the data of the parent is obtained by a volume weighted averaging of the child data, i.e.

$$\mathbf{U}_j^{\text{parent}} := \frac{1}{|T_j|} \sum_{i=1}^{8} |T_j^{\text{child } i}| \mathbf{U}_j^{\text{child } i}.$$

Also, with this procedure the conservation property holds, since

$$\int_{T_j} \mathbf{U}_j^{\text{parent}} = |T_j| \mathbf{U}_j^{\text{parent}} = \sum_{i=1}^{8} |T_j^{\text{child } i}| \mathbf{U}_j^{\text{child } i}$$
$$= \sum_{i=1}^{8} \int_{T_j^{\text{child } i}} \mathbf{U}_j^{\text{child } i}.$$

6.2.7 Mesh Adaption Criteria

Now we have to decide which elements of the grid are to be refined and where a coarsened mesh is sufficient. Therefore, we would need to know the numerical error on each cell at each time-step (cf. [58]).

6.2.8 Error Estimator

For linear elliptic or parabolic partial differential equations rigorous a-posteriori error estimates of the form

$$\|u - u_h\| \le C h^\alpha \eta(u_h)$$

with an adequate norm $\|\cdot\|$, the exact and approximated solution u and u_h respectively, a constant C, α positive and a term $\eta(u_h)$ depending on the problem, are well-established. This implies that the numerical error can be reduced by decreasing the mesh size.

For the considered complex flow problem, i.e. the time-dependent convection dominant Navier–Stokes equations on a bounded domain in three space dimensions, no such a-posteriori error estimations are available up to now. Although for a model problem

$$\begin{cases} u_t + \nabla \cdot f(u) - \varepsilon \triangle u = 0 & \text{in } \mathbb{R}^2 \times (0, T_{\text{end}}), \\ u(x, y, 0) = u_0(x, y) & \text{in } \mathbb{R}^2 \end{cases}$$

with $\varepsilon \ge 0$ as a diffusion parameter, a-posteriori error estimates have been established (see [59, 60]). Furthermore, in [61] an a-posteriori error estimate has been given for the boundary value problem in a multi dimensional setting for the scalar case for a general conservation law. Thus, this is a research field of special interest.

6.2.9 Error Indicator

Several approaches can be exploited to derive an error indicator. First of all, the residual of the numerical approximation gives a first hint of the location of the error (see e.g. [29]). For finite difference schemes for a conservation law in one space dimension a rigorous error estimator has been derived by Tadmor et al. in [62]. However, the used norms cannot be computationally evaluated.

Another method is to calculate the approximate solution on two meshes, one of them having a smaller mesh size. Then, in areas of large difference between the two solutions a further refinement would be indicated.

A similar technique uses the difference of a numerical solution of higher order to one of first order (as done e.g. in [63]). Also, here a big difference is a sign for mesh refinement. As the latter two ideas are very time consuming and complex, heuristic criteria are used most often.

6.2.10 Heuristic Criteria

Heuristic indicators are usually based on local gradients. A large part of the numerical error occurs normally at discontinuities, i.e. the shocks and contact discontinuities, which are badly resolved on a discrete mesh and induce numerical diffusion. These structures can be detected by the use of gradients of key variables. This procedure is founded on theoretical results ([60, 64]) and numerous numerical experiments (see e.g. [33, 37, 65–67]).

6.2.11 The Marking Strategy

For marking an element to be refined or coarsened, we use a heuristic strategy which is easy and fast to compute as it employs only readily available quantities (compare Sect. 6.2.7 and [33]).

Definition 6.2 (Local weighted flow quantity) Let $u_j^n := x(\mathbf{U}_j^n)_k$, $k \in \{1, \dots, 5\}$ being a scalar component of the vector of conservative variables of element T_j at time t^n and $\mathbf{w}_j$ being the center of gravity of T_j. Then (for unrefined neighbors)

$$\eta_j^n := \left(|T_j| \left(\left(\frac{u_j^n - u_j^{n-1}}{\Delta t^n} \right)^2 + \sum_{l=1}^{k_j} \left(\frac{u_j^n - u_{jl}^n}{|\mathbf{w}_j - \mathbf{w}_{jl}|} \right)^2 \right) \right)^{\frac{1}{2}}$$

defines the local weighted flow quantity. In the case of a refined neighbor T_{jl} the following definition

$$\eta_j^n := \left(|T_j| \left(\left(\frac{u_j^n - u_j^{n-1}}{\Delta t^n} \right)^2 + \sum_{l=1}^{k_j} \left(\frac{1}{k_{jl}} \sum_{k=0}^{k_{jl}} \frac{u_j^n - u_{jlk}^n}{|\mathbf{w}_j - \mathbf{w}_{jlk}|} \right)^2 \right) \right)^{\frac{1}{2}}$$

with k_{jl} the number of facing Elements T_{jlk} on face S_{jl} with values u_{jlk}^n and center of gravity $\mathbf{w}_{jlk}$ is used.

The mesh adaption criterion is then the relative value, which corresponds to an evenly distributed approximation error.

Definition 6.3 (Local relative mesh adaption quantity) Let

$$\bar{\eta}^n := \frac{1}{\sharp\mathcal{T}} \sum_{\substack{j \\ T_j \in \mathcal{T}}} \eta_j^n,$$

be the average of η_j^n on the mesh $\mathcal{T}$, with $\sharp\mathcal{T}$ being the number of elements in $\mathcal{T}$. Then

$$\zeta_j^n := \frac{\eta_j^n}{\bar{\eta}^n}$$

is the local relative mesh adaption quantity on element T_j at time t^n.

Now the mesh adaption criterion, when to refine or to coarsen an element, can be stated.

Definition 6.4 (Local mesh adaption criterion) Let ζ_j^n be the local relative mesh adaption quantity of element T_j at time t^n from Definition 6.3. Then the element T_j is marked for refinement or coarsening according to:

$$\begin{aligned} &T_j \text{ marked for refinement} && \text{if } \zeta_j^n > C_{\text{refine}}, \\ &T_j \text{ marked for coarsening} && \text{if } \zeta_j^n < C_{\text{corsen}} \end{aligned}$$

with positive constants $C_{\text{refine}} > C_{\text{coarsen}}$.

Remark 6.5 (Choice of $C_{\text{refine}}/C_{\text{coarsen}}$) As described in [33], the constants to control the mesh adaption $C_{\text{refine}}/C_{\text{coarsen}}$ are chosen depending on the desired adaption accuracy. They can lie between $C_{\text{refine}}/C_{\text{coarsen}} = 0.5/0.1$ for a very sensitive mesh refinement up to $C_{\text{refine}}/C_{\text{coarsen}} = 1.3/0.7$ if the mesh should only be refined at intense structures in the numerical solution.

Remark 6.6 (Used components for marking) As indicators for the marking strategy we use the density ρ, the fresh-gas concentration σ, and the pressure p.

6.2.12 Special Mesh Considerations

This standard algorithm now needs further extensions in order to work with the above described mesh treatment for complex geometries with moving parts.

6.2.13 Boundary Fitting

As the boundary of our domain is not planar but curved in most parts, the new boundary vertices of refined elements do not lie exactly on the boundary. These vertices can be projected onto the exact geometrical boundary if the surface of this boundary is known. Thus, for a complete treatment of all boundaries it would be necessary to use the CAD data of the geometry. Since this data is a very complex structure, and as an initial mesh of fine enough resolution approximates even complex geometries quite well, such a projection is only needed in critical areas. Especially at the inter-mesh interfaces where the ports connect to the cylinder and crankcase, and at the crankshaft-crankcase interface, a high accuracy is desired. But the exact geometry of the lower part of the cylinder and the upper part of the crankcase is also known without extraction from the CAD data. They are just cylindrical bodies with given radius. Therefore, a projection onto these surfaces is a straightforward trigonometrical calculation.

6.2.14 Moving Mesh Interaction

Elements in the rotating crankshaft mesh are constantly changed. Also, the newly created vertices of refined elements have to be included in the rotational movement and the update of their outer normals $\mathbf{n}_{jl}$. An adaptive treatment of the crankshaft-crankcase interface has been avoided by initially refining this part of the mesh up to the maximum refinement level and fixing it for the time iteration. This has been done for efficiency reasons. It is too time consuming to update the concerned structures with every refinement. Also, a high accuracy is needed on this interface, thus justifying the computation on the maximum refinement level.

At the moving boundary just above the piston crown, and similarly on the uppermost layer below the piston in the crankcase, the mesh is likewise refined to maximum depth. The reason also being time considerations within the moving piston algorithm and the desired accuracy in this region of high activity. But unlike in the case of the rotating crankshaft mesh, this refinement needs constant updating due to the snapper algorithm described in Sect. 5.2. If, during the upwards motion of the piston, a merging of two layers above the piston occurs then a further refinement of the cells in the second layer above the piston might be necessary. Inversely, at the downward motion a refinement, induced by the piston motion, is not needed anymore after the splitting of two layers. Therefore, a coarsening can be conducted if the local mesh adaption criterion indicates this. Furthermore, special care has to be taken at the neighbors of moved elements in the crankcase mesh below the piston whose data is updated during the snapper algorithm as described in Sect. 5.2.7. These elements have to be refined to maximum level as well in order to treat them with the procedure described in Sect. 5.2.7.

Remark 6.7 (Modified marking strategy) Due to the strong influence of the mesh movement on the temporal part of the marking strategy, this part is omitted in the two-stroke engine simulation.

6.2.15 Merged Meshes

For the purpose of achieving a high accuracy on the inter-mesh interfaces, and to avoid an excessive calculation of the area of common facets of neighboring elements across the contact boundary, we also decided in this case to refine these boundaries up to the maximum refinement level. The update of the merging grid structure therefore only has to be done at initialization time and not dynamically during the time stepping iteration.

7 Parallelization

Most applications have an unbounded appetite for memory and processor time. The efficient use of computing resources is, therefore, always a concern in scientific computing. E.F. van de Velde [68]

This citation emphasizes the desire of every CFD simulation to be as accurate as possible and to calculate problems with ever-increasing complexity within a reasonable time. By the use of parallel computers, the runtime of a problem can be reduced considerably. Also, the available memory in parallel computers is many times bigger.

7.1 Efficiency of a Parallel Algorithm

The reason for parallel computing is founded in the need to execute an identical problem in less time (speed-up) or a larger problem in the same time (scalability). The following definitions are used to measure these quantities for a given algorithm.

Definition 7.1 (Speed-up) The *speed-up* $Sp(n)$ is defined as the ratio of the sequential total execution time $T(1)$ to the time needed with n processors $T(n)$:

$$Sp(n) := \frac{T(1)}{T(n)}.$$

Definition 7.2 (Efficiency) The *efficiency* E describes the ratio of actual speed-up $Sp(n)$ to ideal speed-up $Sp_{\text{opt}}(n) = n$ (see Remark 7.4):

$$E(n) := \frac{Sp(n)}{n}.$$

Definition 7.3 (Scalability) Let an n-scaled problem nP for a problem P be given by $T_{nP}(1) = nT_P(1)$, where $T_P(1)$ is the serial execution time for P. Then the scalability $Sc(n)$ is defined by:

$$Sc(n) := \frac{T_P(1)}{T_{nP}(n)}.$$

Remark 7.4 The theorem on the limitation of speed-up states that $Sp(n) \leq n$ (see, e.g. [70]).[6] Therefore, it follows for the efficiency and scalability $E(n) \leq 1$ and $Sc(n) \leq 1$.

The speed-up and scalability of an algorithm are determined by the portion of the work load T_{par} that can be distributed between the processors and the work load T_{ser} that needs to be executed by only one processor. An ideal speed-up $Sp(n) = n$ is obtained if $T_{\text{ser}} = 0$, i.e. the algorithm is fully parallelized and the whole load can be distributed between the n processors (see also Remark 7.6). This is, e.g., the case in Monte-Carlo simulations, where the single tasks are completely independent of each other. But, in general, $T_{\text{ser}} \neq 0$, thus it follows for the speed-up $Sp(n) = T_{\text{ser}} + \frac{T_{\text{par}}}{n}$. This is, if T_{ser} and n are small, unproblematic, but it is critical if either T_{ser} or n are bigger (Amdahl's Law).

Example 7.5 In the case of $T_{\text{ser}} = 0.2$ and $n = 8$ the total execution time reduces to $T(8) = 0.2 + \frac{0.8}{8} = 0.3$ resulting in a speed-up of $Sp(8) = 3.\bar{3}$ which is far from the expected ideal speed-up of $Sp_{\text{opt}}(8) = 8$.

Remark 7.6 In addition to the serial part T_{ser} of the algorithm the parallel overhead, such as synchronization and data exchange, slows down the parallel execution of the program.

But a gleam of hope is spotted in the fact that the non-parallelizable part T_{ser} generally decreases with increasing size of the problem (Gustafson's Law). Thus, even with bad performance with respect to speed-up $Sp(n)$ the algorithm can still possess a good scalability $Sc(n)$.

For further details cf. [68–71].

7.2 Parallelization Concepts

In recent years two different types of supercomputers have become widely available, namely the shared memory architecture and the distributed memory architecture. As these two types of architecture also influence the efficiency of the parallelization strategy, they are now described in further detail.

[6] A possible exception to this rule is the improved cache performance for smaller problems.

7.2.1 Distributed Memory Architectures

In this approach the parallel computer consists of many individual processors, each possessing its own memory. One processor cannot access the memory of another. An executed program and its processed data is split into n parts and is distributed to the n processors. An exchange of information is possible via a communication network connecting the processors with each other (see e.g. [72]). A fast network, with short length of path from one processor to any other one, is essential for the performance of such a machine. This is also the limiting factor of the size of such supercomputers which can, nevertheless, consist of up to thousands of processors.

For the information transfer on such architectures there exists a widespread standard, the *message passing interface MPI* (see [73]).

7.2.2 Shared Memory Architectures

This type of parallel computer also consists of individual processors but each of them has access to the same memory. An exchange of information is therefore easily realized by the common memory, an explicit, time consuming, message passing is not necessary. This advantage is diminished by the risk of the, so-called, *race condition*, i.e. two processors modifying the same storage position. The outcome of such a situation is non-deterministic and has to be avoided by the programmer. Also, after changing a common variable it is necessary to update the cache of every processor keeping this variable. For further details confer Sect. 7.3.

For hardware considerations this architecture is more difficult to realize than the distributed memory approach. The requirements for the fast memory access via the memory switch limit the size of such an architecture (nowadays) to some hundred processors.

The accepted programming standard for the usage of shared memory computers is the OpenMP standard, as described in [74] and briefly in Sect. 7.3.

7.2.3 Hybrid Architectures

These are parallel computers sharing the characteristics of both of the above described architectures. The largest supercomputers these days are of this type. They consist of many nodes connected to each other by a communication network, whereas each node is composed of processors accessing one memory block within their node.

In the programming model both types of directives, MPI as well as OpenMP, are used.

7.3 The OpenMP Parallelization

7.3.1 The Finite Volume Scheme

The parallelization of the main numerical finite volume scheme is quite straightforward.

First, the mesh is divided into n_{part} disjoint domains $\mathcal{D}_i$ ($1 \leq i \leq n_{\text{part}}$), called partitioning. The objective for the creation of these domains is an equal distribution of elements. This is necessary for the even distribution of the load between the processors (further clarified in Sect. 7.5). Furthermore, small boundaries to other domains are desirable, due to the fact that the data $\mathbf{U}_j$ on each element situated on a domain-boundary can be modified by two processors.[7] Thus, in order to avoid the above-mentioned race condition, this element has to be locked before changing the content of $\mathbf{U}_j$ and unlocked afterwards. As this locking/unlocking needs time for itself, and might block the other processor, this should be avoided as much as possible. The partitioning is performed by the *recursive coordinate bisection algorithm*. It is fast, easy to implement, and efficient regarding the above required attributes (cf. [70, 75]). In order to handle the diverse add-ons to the main numerical scheme, this recursive coordinate bisection had to be extended, which is described in Sect. 7.4.

Then the OpenMP `#pragma omp parallel`-directive is applied to create n different threads (each being run on one processor) which execute the same code from this point onwards. Thus, each thread i calculates the same scheme on all elements $T_j \in \mathcal{D}_i$ of the domain $\mathcal{D}_i$ that has been appointed to it.

For synchronization of the threads the directive `#pragma omp barrier` is used. The threads wait at this point for each other. This is, e.g., necessary at the end of one time-step, before the threads use the data $\mathbf{U}_{jl}^{n-1}$ of the neighboring elements from the time-step before. It has to be assured that this data is completely calculated.

For the update of global variables it is crucial that only one thread at a time accesses this variable to avoid the race condition. This is done by the `#pragma omp critical`-directive. E.g. for the new minimum time-step $\Delta t_{\min}$ each thread i calculates its own local minimum time-step $\Delta t_{i,\min}$ on domain $\mathcal{D}_i$ and compares it with the global $\Delta t_{\min}$. If $\Delta t_{i,\min} < \Delta t_{\min}$ then $\Delta t_{\min} := \Delta t_{i,\min}$, using this directive.

A similar case is given for instructions that only one thread needs to execute. They are indicated by the `#pragma omp single`.

And finally, the locking of the data $\mathbf{U}_j^n$ that can be modified by more than one thread (if the element T_j is situated

[7]For the reason of the modification of the data $\mathbf{U}_{jl}^n$ of neighbor T_{jl} see Remark 6.1.

at a domain boundary) is done by the OpenMP library functions `omp_set_lock()` and `omp_unset_lock()`.

The changes that need to be applied to the serial program are minor, such that the same code can be used for serial and parallel execution. The dedication is set at compilation time by the compiler flag `-OPENMP`.

7.3.2 The Mesh Routines

The parallelization of the main scheme resulted in a code which had a rather big serial part T_{ser}, most importantly due to the routines for the mesh movement and the dynamic local mesh refinement. Therefore, the speed-up and efficiency of this parallel code was not very impressive. To enhance the performance two strategies were applied.

On the one hand, the absolute execution time of the mesh related routines was optimized. This was achieved by profiling and tuning the concerned program parts, and by changing the mesh not after each step but only after $2^{l_{\text{geom}}}$ steps, with l_{geom} as a given parameter (cf. Sect. 6.1).

On the other hand, these routines used for mesh treatment were parallelized to the greatest possible extent. For this, we used not just one partitioning of the mesh for the finite volume scheme, as described in Sect. 7.3.1, but in parallel also several others for the mesh related routines (see Sect. 7.4).

7.4 The Extended Partitioning Algorithm

The load of a single processor during the execution of the finite volume scheme is determined by the number of elements being updated at the current time-step t^n. Because of the adaptive local time-stepping (cf. Sect. 6.1) this number is different for each time-step level l_{timestep}. An element T_{j_1} belonging to time-step level $l_{j_1} = l_{\min}$ is calculated every time-step, one element T_{j_2} belonging to the next higher level $l_{j_2} = l_{\min} + 1$ every second step, and so on. Thus, for a time-step in which only elements of level $l_{\text{timestep}} = l_{\min}$ are updated, only these elements contribute to the load in this step. In a time-step for level $l_{\text{timestep}} = l_{\min} + 1$ all elements of level $l_j \le l_{\min} + 1$ are to be considered, and so forth. Therefore, we need for each level $l_{\min} \le l_{\text{timestep}} \le l_{\max}$ a separate partitioning in order to obtain a correct load balancing for every single time-step. That means that the domains $\mathcal{D}_i^{l_k}$ $(k = 1, 2)$ can differ for $l_1 \ne l_2$.

Additionally, the mesh routines need to work in parallel, too. For the snapper algorithm, e.g., most of the time only one layer of elements in the cylinder and the crankcase are changed (see Sect. 5.2). Thus, the elements in this layer have to be partitioned for the n_{part} processors on the fly. Similarly, for the treatment of the rotating crankshaft mesh we need a balanced partitioning of this part of the mesh. The dynamic local mesh adaption algorithm (described in Sect. 6.2) is operating on every cell, thus a further partitioning of the whole mesh is necessary. As the number of elements on domain boundaries is not important in the case of the mesh routines (no data on neighboring elements needs to be changed, thus no locking is necessary) a simple partitioning by element number is carried out. Summarizing this, we obtain a whole family of partitions $\mathcal{P}^r := \{\mathcal{D}_i^r | 1 \le i \le n_{\text{part}}\}$ with $1 \le r \le l_{\max} - l_{\min} + 4$ with each partitioning having a particular role within the load balancing procedure.

This approach therefore uses the capabilities of the shared memory architecture to its full extent, since a new partitioning does not need any communication or data exchange as it would be the case in distributed memory machines. On this distributed memory architecture the above described approach of using more than one partitioning would be impossible.

7.5 Dynamic Load Balancing

The load balancing in our case is a very important issue. As the mesh is changed by several different mechanisms, the size of the partitioning domains $\mathcal{D}_i^r$ is also affected. However, if these domains are not evenly balanced the performance of the parallelization suffers, since some processors have a lower work load, waiting for the others to finish (cf. [72]). Therefore, a dynamic load balancing is essential.

For this, a concept of work load has to be derived (see also [70]). It is assumed that the work load is proportional to the number of elements that a processor has to compute in a certain time-step level. It follows for the cost per level for each domain $\mathcal{D}_i$:

$$W(l,i) := m_{li}\, 2^{-l}$$

with m_{li} being the number of elements T_j in domain $\mathcal{D}_i$ with $l_j \le l$. The total cost for a complete level cycle is thus:

$$W := \sum_{l_{\min} \le l \le l_{\max}} \max_i (W(l,i)).$$

Every $2^{l_{\text{part}}}$ steps, with $l_{\text{part}} \in \mathbb{N}$ a given constant, the even distribution of the domains is verified by calculating the ratio of the number of elements of the biggest to the smallest domain:

$$\zeta_{\text{loadbal}}^n := \frac{\sum_{l_{\min} \le l \le l_{\max}} \max_i W(l,i) \frac{\max_i (m_{li})}{\min_i (m_{li})}}{W}.$$

In the case of exceeding a given threshold $C_{\text{loadbal}} > 1$ a new partitioning, i.e. a re-distribution of the mesh to the processors, is created by the extended recursive coordinate bisection, as described in Sect. 7.4.

8 Validation of the Software

For simple model problems, the convergence properties of a finite volume scheme can be studied analytically (see e.g. [11]). However, in our case of the three-dimensional time-dependent Navier–Stokes equations, the numerical scheme has to be evaluated by numerical experiments. The results of these calculations can then be compared to known exact solutions, calculations with other schemes found in literature, or physical experiments. All of these methods were conducted and can be seen in detail in [76].

9 Results

Our final goal is the simulation of the flow through a two-stroke engine. Apart from the general study of the flow structure in an existing two-stroke engine, we treat in this chapter two problems of particular interest for the engineer. The first one is the question of how much does the crankshaft motion influence the flow situation in the cylinder and the exhaust characteristics. Is it therefore really necessary to simulate the moving parts in the crankcase. Finally, we want to study the mechanism and the paths of the short-circuit flow from transfer port to exhaust port with help of the enhanced fresh-gas tracking.

9.1 Meshes

9.1.1 Generation

The quality of the underlying meshes is of vital importance for the obtained solution. Bad quality meshes may result in a very small time-step (in the case of elements with very small volume) or in an oscillating solution (arising from strongly skewed elements). It was pointed out that our meshes consist of hexahedrons due to better performance. As we are in the case of a complex real-world mesh based on CAD data, a commercial mesh generator is necessary with this type of mesh (as e.g. [77]).

The geometry of a two-stroke engine has a complex topology, thus it is inevitable to use a composition of a family of meshes assembled by the grid merging method to form the whole geometry (as shown in Fig. 8). It would not be possible to construct one single mesh meeting all the geometrical restraints. With the grid merging technique, O-grids can be used to mesh the inlet, transfer port and outlet duct, thus accounting for their cylindrical shape and the possible boundary refinement. Also, the upper edge of the transfer ports connecting to the cylinder does not need to be horizontal, as would be the case without grid merging due to the

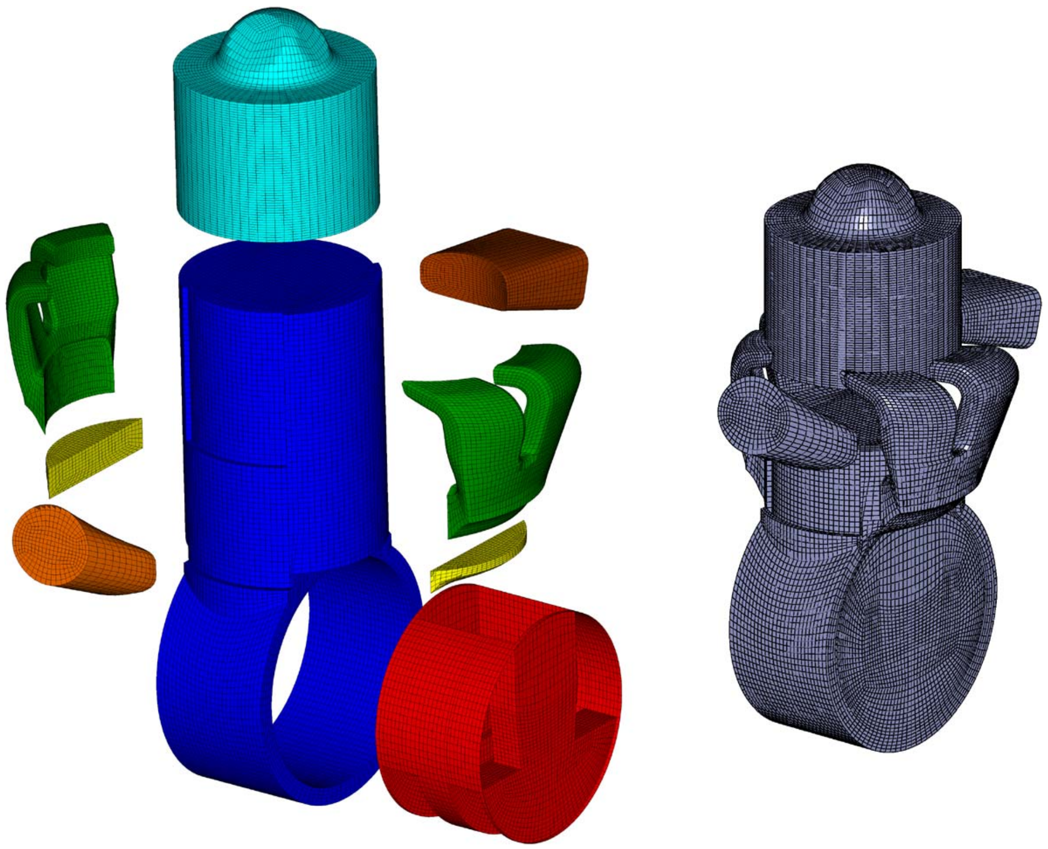


Fig. 8 Meshes: the complete mesh is assembled by several different parts (*left*), on the *right* the 250 000 element mesh is displayed

Table 1 The element number of the different mesh families for the two-stroke engine simulation

Mesh family	15,000	30,000	60,000	120,000	250,000
Inlet duct	270	432	826	1,680	3,456
Crankcase	3,952	7,595	14,444	31,572	65,838
Crankshaft	1,920	3,420	7,916	14,313	26,848
Lower transfer ports	236	430	866	1,744	3,440
Transfer ports	2,144	4,272	8,032	17,522	34,048
Cylinder	5,271	10,206	20,922	41,300	82,548
Outlet duct	512	1,298	2,632	5,032	10,384
Silencer	1,160	2,592	5,190	10,507	20,736

horizontal layering of the cylinder elements, as needed by the snapper algorithm (cf. Sect. 5.2).

9.1.2 *Different Size Meshes*

In order to check for the mesh dependency of our solution, we need a whole series of families of meshes. Seven families of meshes have been generated. The element numbers can be seen in Table 1. Further meshes with higher element numbers are created by global or adaptive refinement of the existing ones. Therefore, the number of elements of a family of meshes is only limited by the computational time of the simulation on the resulting mesh.

9.1.3 *The Silencer—An Alternative to a Transparent Boundary Condition*

The handling of the outflow boundary is not easy. The straightforward characteristics based approach reflects waves back into the interior of the domain (if we are in the subsonic regime, which is the case in most of the outflow duct).

One method to cover this problem is the usage of a transparent or artificial boundary condition (ABC) at the end of the outflow duct. These boundary conditions approximate the flow that would continue in an infinitely long outflow duct. They are an improvement to the characteristics based boundary treatment in that they do not influence the interior flow by perturbations created at the boundary. Usually, this kind of boundary condition is found in the simulation of outer flow problems, e.g. airfoil calculations (cf. [78]). In our case of the viscous three-dimensional time-dependent duct flow with boundary layer, the transparent boundary condition approach is not as well studied as the far field boundary in an outer flow problem. But, more importantly, in the two-stroke engine context the pressure wave, resulting from exhaust port opening, is reflected at the transition of outflow duct to silencer. Therefore, in reality we have a strong interaction between outflow boundary and interior flow. This can only be simulated by adding the silencer to our family of meshes and thus calculating the flow through this part of the engine as well. The disturbing boundary is now far away from the cylinder at the outlet of the silencer. As can be seen from Table 1, only about 8% of the total number of elements are needed for the mesh of the silencer due to its basic geometry.

Table 2 Geometrical data of the test engine

Variable	Value
Bore d_b	47.0 mm
Stroke l_s	34.0 mm
Connecting rod length l_cr	58.0 mm
Trapped stroke l_ts	23.3 mm
Swept volume V_s	59.0 cm^3
Trapped swept volume V_ts	40.4 cm^3
Rotations per minute rpm	$9000\ \frac{1}{\mathrm{min}}$
Crankcase compression ratio CR_cc	1.35
Geometric compression ratio CR_g	10.59
Trapped compression ratio CR_t	7.57

9.2 Configuration

9.2.1 *Geometrical Data*

The geometrical data of the two-stroke engine with which we test our software are given in Table 2 (for the definitions of these entities see Sect. 2.3). The values are extracted from the CAD data of a real engine employed in a chain saw. The port areas as functions of time of the different ports are shown in Fig. 9. The timing for the opening and closing is specified in Table 3. The piston motion is determined by explicit formulae for h_p and v_p. The rotation of the crankshaft is given by the value for the rotations per minute. Thus, in our case we have 9000 rpm, i.e. the crankshaft rotates 360° per $\frac{1}{150\ \mathrm{s}}$.

9.2.2 *Temperature-dependent Material Properties*

In order to calculate the material properties of the fresh gas and the exhaust gas, the formulae as stated in Sect. 3.5 with the appropriate values for fresh air and the exhaust gas mixture are used. (For further details see [76].)

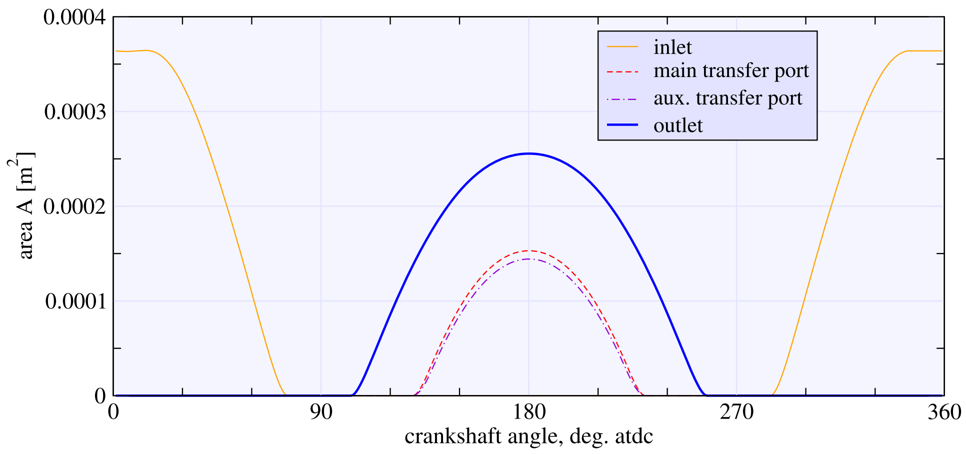


Fig. 9 Port areas by piston control of the different ports

Table 3 Port timing for the test engine

Port	Opens ° crankshaft atdc	Closes ° crankshaft atdc
Inlet port	285	75
Main transfer port	130	230
Auxiliary transfer port	130	230
Outlet port	104	256

9.2.3 *Combustion Data*

As the combustion process itself is not simulated, we need the actual state of the exhaust gas within the cylinder after the combustion has terminated. This data is extracted from an experimental measurement:

$$p(\mathbf{x}, t) := 1092360 \text{ Pa}$$

$$\text{for } t = 50^\circ \text{ crankshaft angle atdc}, \mathbf{x} \in \text{cylinder}.$$

The density does not change during combustion, therefore, the temperature can be calculated by the thermal equation of state (3.7).

9.3 Initial and Boundary Data

9.3.1 *Initial Conditions*

The initial data for the coarsest mesh is set to ambient standard temperature and pressure for the inlet and crankcase:

$$\rho_0 := 1.2039 \, \frac{\text{kg}}{\text{m}^3},$$

$$p_0 := 101325 \text{ Pa}$$

and for the outlet and silencer:

$$\rho_0 := 0.4100 \, \frac{\text{kg}}{\text{m}^3},$$

$$p_0 := 101325 \text{ Pa}.$$

Whereas in the cylinder the trapped compression ratio is applied to these values based on a polytropic compression:

$$\rho_0^{\text{cyl}} := \rho_0 C R_\text{t} = 9.1135 \, \frac{\text{kg}}{\text{m}^3},$$

$$p_0^{\text{cyl}} := p_0 (C R_\text{t})^{1.4} = 1723657 \text{ Pa}.$$

The velocity field is assumed to vanish in the whole geometry. The initial data does not influence the asymptotic solution. The boundary conditions determine the converged periodic solution. The initial values just influence the time needed for the solution to reach a converged state.

For the subsequent meshes, the converged solution of the previous mesh is used as initial data in order to speed up the convergence.

9.3.2 *Boundary Conditions*

The numerical treatment of the boundary conditions is described in Sect. 4.5. The outflow boundary condition has also been discussed in Sect. 9.1.3. The inflow boundary is the only one that needs further consideration. Two possible approaches are feasible to cover this boundary. If the massflow data from a physical experiment is available, this data can be used to impose a massflow condition at the inflow boundary. Otherwise, a the ambient standard pressure $p_\text{in} := p_0$ and density $\rho_\text{in} := \rho_0$ can be prescribed.

The advantage of the massflow condition is a better comparability of two different engine geometries. Because the massflow through the engine is always the same (due to the setting at the inflow boundary), the delivery ratio DR is kept constant, and therefore also the scavenge ratio SR (with the assumption of identical cylinder volumes). Therefore, the trapping efficiency TE as main criterion is only influenced by the geometrical shape of the engine, and not by the amount of delivered fresh gas.

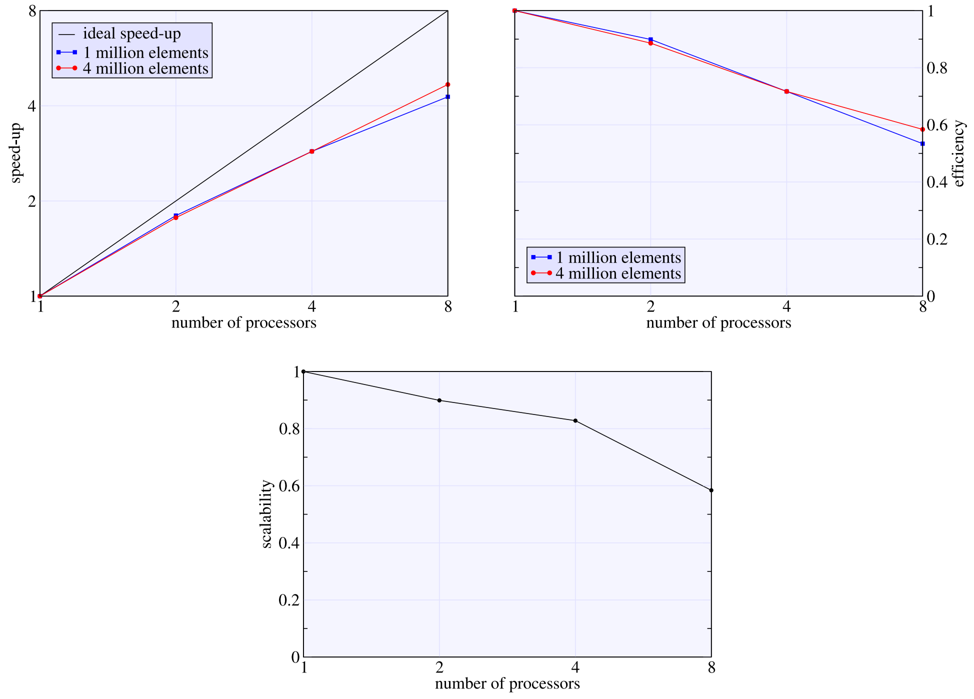


Fig. 10 Parallelization: speed-up (*upper left*) and associated efficiency (*upper right*) for the simulations on the 1 million and 4 million element meshes and the scalability (*bottom*) for a computation on the 500 000 (1 processor), 1 million (2 processors), 2 million (4 processors), and 4 million (8 processors) element meshes

The advantage of the standard inflow condition, on the other hand, is its ease of use. No time-dependent measurements on a real engine need to be conducted, and no time-dependent massflow value needs to be enforced at the inflow boundary. But an interaction of the flow with this kind of boundary condition results in a variable massflow rate, which makes it difficult to compare different geometries.

9.4 Parallelization

The code was executed on IBM Regatta p650 and p690+ with up to 32 Power 4+ processors with 1.7 GHz in shared memory operation.

9.4.1 Partitioning

As stated in Sect. 7, the parallelization was performed using the OpenMP standard for shared memory parallel architectures. The concept of multiple simultaneous partitions of the same mesh, explained in Sect. 7.4, relies on the shared memory concept.

9.4.2 Performance Analysis

Two types of studies can be conducted. One is to increase the number of processors for the same problem. Thus, by doubling the number of processors, the size of the problem per processor is halved. This approach leads to the concept of speed-up and efficiency (see Sect. 7.1). The other type consists in increasing the problem according to the number of processors such that the problem size per processor remains constant. This is measured by the scalability (as defined in Sect. 7.1). Both types of studies have been performed. The results can be seen in Fig. 10. The speed-up (upper left) and the associated efficiency (upper right) for the 1 million and 4 million element mesh is quite good for such a complex algorithm. With 8 processors the larger mesh performs slightly better. This trend continues probably on more processors. The scalability (bottom) is based on simulations on 500 000, 1 million, 2 million, and 4 million elements meshes with 1, 2, 4, and 8 processors respectively. Also, here a relatively good performance is demonstrated.

9.5 Visualization and Characteristical Diagrams

The visualization of large data is a very important issue in analyzing the flow through a complex geometry. The visualization software, that is part of our software package, is based on the library GRAPE (for GRAphics Programming Environment) (see [54–57, 79–81]) and provides an interface for our mesh structures. The visualization software reuses large parts of the code of the main simulation software, e.g. to move the piston and crankshaft and to calculate the grid merging. Additionally, several visualization routines have been implemented to display certain aspects of the special algorithms used in the simulation. These include the visualization of the local time-step levels, the partitioning for the parallel algorithm with the ghost cells, and the opaque clipping routine to enhance the clarity of the presentation. Already present in GRAPE is a wide variety

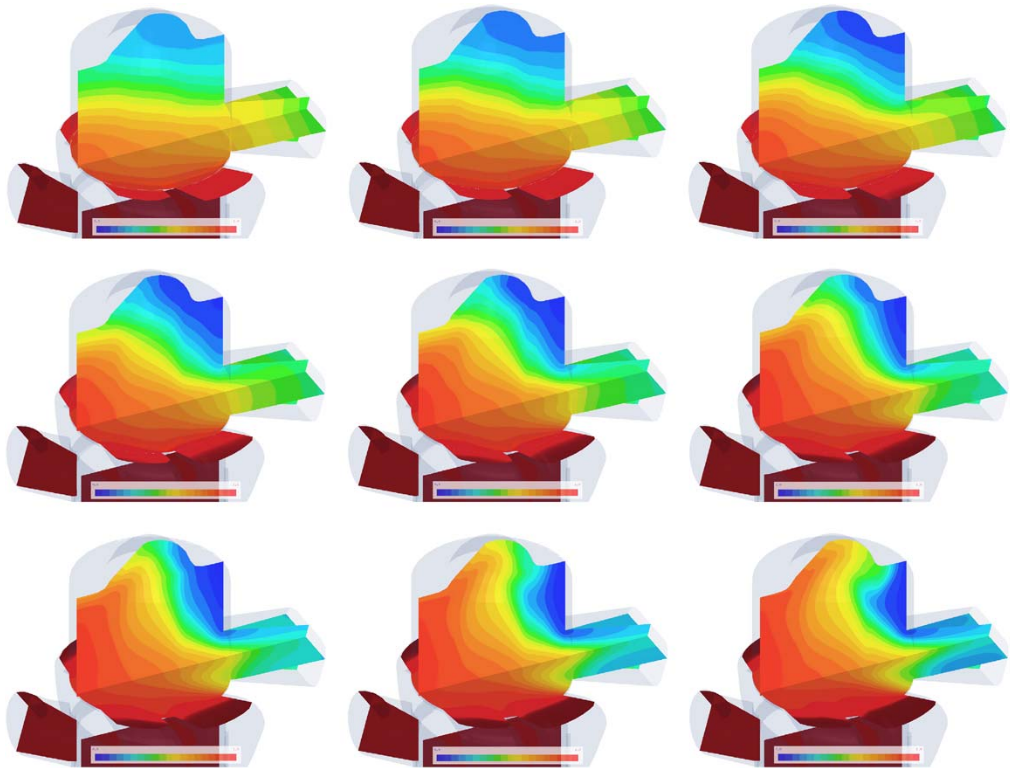


Fig. 11 Convergence of the flow structure for the meshes from 15 000 to 4 000 000 elements (line-by-line) at 200° crank angle atdc

of display methods, e.g. the level display method (as seen in Figs. 16 and 17), the particle trace method (in Figs. 14 and 15), and multiple-isoline and vector field clipping displays (e.g. Fig. 19). All of these routines can be used in a time-dependent context, thus taking into account the variations in the flow with time (compare also the presentation of the results in the next Sect. 9.6). With the help of these very powerful routines an exhaustive analysis of the flow structure in the two-stroke engine is made possible.

Furthermore, for the quantitative study of the exhaust characteristics of the engine several output files are created during the simulation. This data can be used to generate, e.g. massflow and trapping efficiency diagrams (as displayed e.g. in Figs. 21 and 20). In this manner the quality of two or more geometries can be compared with each other.

9.6 The Simulation

9.6.1 Convergence in Time

The difference of the numerical solution, measured in a suitable norm, between two consecutive rotations vanishes. This has to be verified within the simulation. We continue the calculation as long as the influence of the initial data is still above a certain threshold. As the scavenge ratio SR and the trapping efficiency TE are the main characteristic values to assess the engine geometry, the difference of the SR and the TE of two consecutive rotations is taken as indicator for a convergence in time. The difference after three periods is for the scavenge ratio less than 1% and even less than 0.5% for the trapping efficiency.

9.6.2 Convergence for Mesh Width $h \rightarrow 0$

When the convergence in time is assured, the calculated numerical solution can be checked for mesh dependency. This is done by comparing the solutions from different meshes with each other. Here the isoline images on the meshes as well as the scavenge ratio SR–trapping efficiency TE graph are taken as indicators for convergence. As the mesh width h decreases, the solutions, and consequently the associated SR–TE graphs, should converge to a mesh independent solution. This is studied for the flow structure of our simulation in Fig. 11. The flow pattern of the solution on the coarse meshes is clearly distinct from the one on the finer meshes. The loop structure is visible only with the finest resolutions. Although there are still differences in details of the looping structure, the general flow pattern converges on the finest meshes. This sequence of images demonstrates very impressively how important it is to calculate a solution on several different size meshes. For an analysis of the flow it is indispensable to know how much influence of the mesh resolution is still present in the numerical solution.

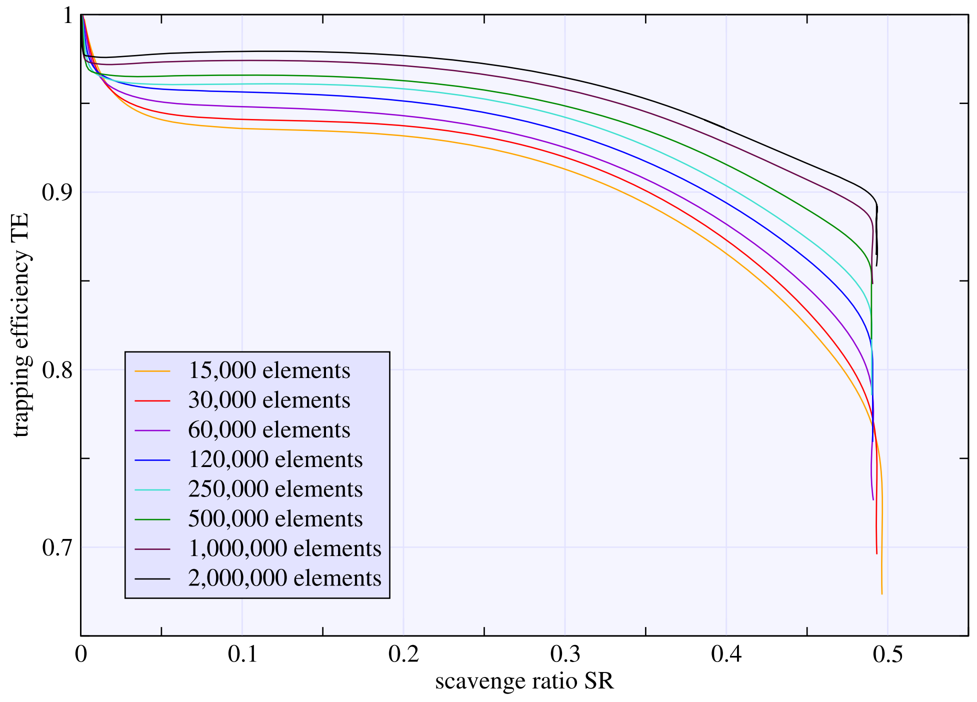


Fig. 12 Convergence of the scavenge ratio–trapping efficiency graph depending on the mesh size

In Fig. 12 the effect of this converging flow structure is shown in the SR–TE diagram. The convergence for the scavenge ratio SR is clearly visible even on the coarser meshes from 120 000 elements onwards. This is not surprising, because we force the delivery ratio DR to a fixed value due to the massflow condition at the inflow boundary. The convergence for the trapping efficiency TE starts to show on the finest meshes. Thus, the trapping efficiency is also assumed to not change much any more on higher resolutions. (For an in depth analysis of the characteristical diagrams see Sect. 9.6.5.)

9.6.3 Comparison with Measured Data

As we are simulating an existing two-stroke engine, experimental measurements are available. The simulated pressure in the cylinder during the scavenging process and the simulated pressure in the crankcase during the whole period are compared to this experimental data. The result is shown in Fig. 13. As the numerical pressure is an average from the whole cylinder and the whole crankcase respectively, the plot is smoother than the measured data that is gathered at one point. But the otherwise very good agreement of this data confirms the high quality of our numerical simulation.

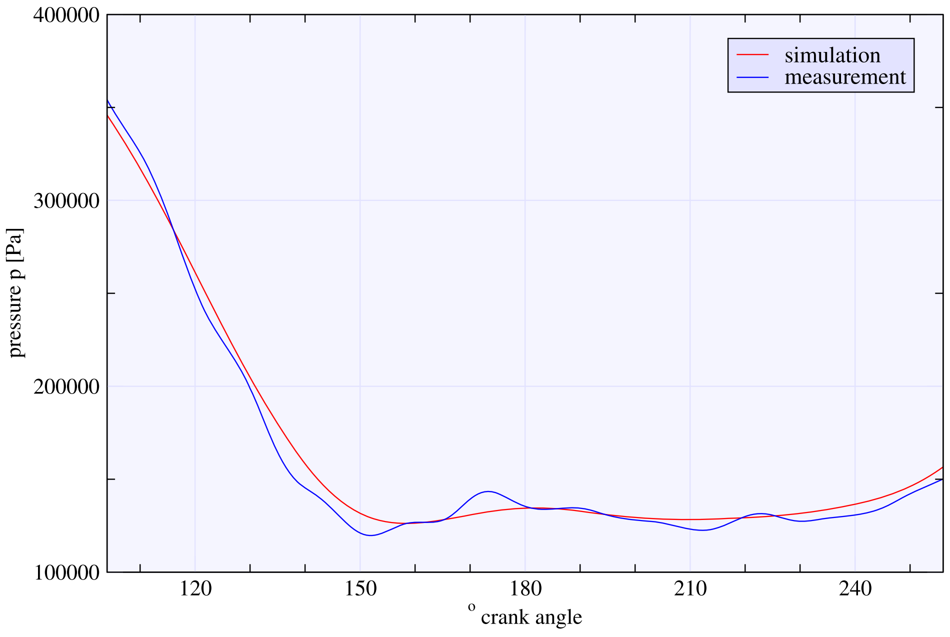
9.6.4 Visualization of the Flow Structure

The analysis of the large data sets of many million numbers calls for powerful visualization strategies. With our visualization software many time-dependent display methods are at hand to study the flow structure of the scavenging process.

First, the main problem of two-stroke engines, the short circuiting, can be demonstrated quite impressively by the particle trace method shown in Fig. 14. Here, the blue particles represent the exhaust gas, the orange and red ones the fresh gas. The dark blue exhaust particle stays in the cylinder during the next combustion, but more importantly, the red fresh gas particle is not trapped in the cylinder but leaves through the exhaust port before it is closed by the upwards moving piston. In Fig. 15, the intake flow into the crankcase is visualized. The light green particles start at the opening of the inlet port at 285° crank angle atdc (after top dead center), and at tdc they are already in the upper part of the crankcase (Fig. 15 (left)). The dark green particles are released at tdc and are pulled into the lower part of the crankcase by the suction of the crank shaft, whereas the earlier particles stay in the upper crankcase (Fig. 15 (middle and right)).

A very good method to display the scavenging process in detail is the three-dimensional iso-level routine. All points with a given value are connected to form a surface. Figures 16 and 17 display two different fresh gas concentrations (0.5 and 0.8) at two different times during the scavenging (180° and 220° crank angle atdc). The scavenging loss by short-circuiting is clearly visible in Fig. 16 (left). In this figure, a “tongue” pattern is evident. This leads not to an optimum scavenging, a high concentration of fresh gas reaches the exhaust port before it is closed (see also [82]).

Objects, that move with the flow and are deformed by it, visualize the time-dependent structure of this flow very intuitively (as shown in the series of pictures in Fig. 18). Here, the main problems of the geometry are demonstrated as well.



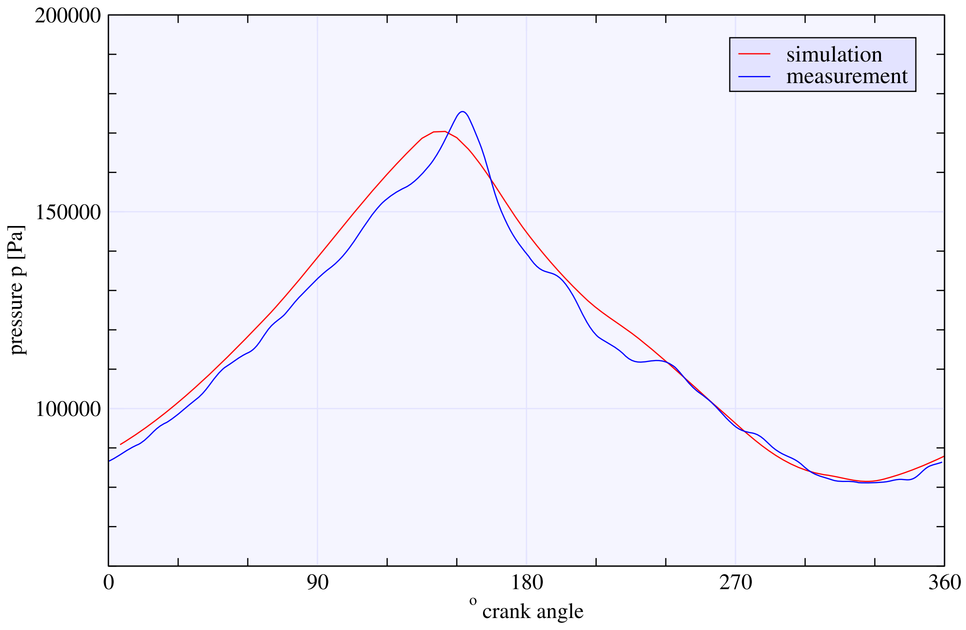


Fig. 13 Comparison with measured data: cylinder pressure during the scavenging process (*top*) and crankcase pressure (*bottom*)

A portion of the exhaust gas (transporting the blue ball) is left in the cylinder, and the short-circuiting transports a part of the red ball out of the exhaust port. The horseshoe-like shape of the blue exhaust object again reveals the "tongue" pattern of the scavenging flow.

With the isoline display, a more in detail view is possible. Figure 19 shows the possibilities of this kind of visualization technique. Also, in this figure the "tongue" pattern of the flow at 180° crank angle (middle part of the figure) is visible, resulting in a loop scavenging loss towards the end of the scavenging process (lower part of the figure). But this is not as damaging as the short-circuit flow from the main transfer port at the beginning of the scavenge process until bdc.

9.6.5 Characteristical Diagrams

Apart from the visualization of the flow through the two-stroke engine, it is important to quantitatively analyze the quality of the geometry by the study of characteristical values. These can be plotted into diagrams to give a better impression of temporal development of the flow. One of the most important diagrams is the scavenge ratio (SR)–trapping efficiency (TE) diagram. Furthermore, the temporal

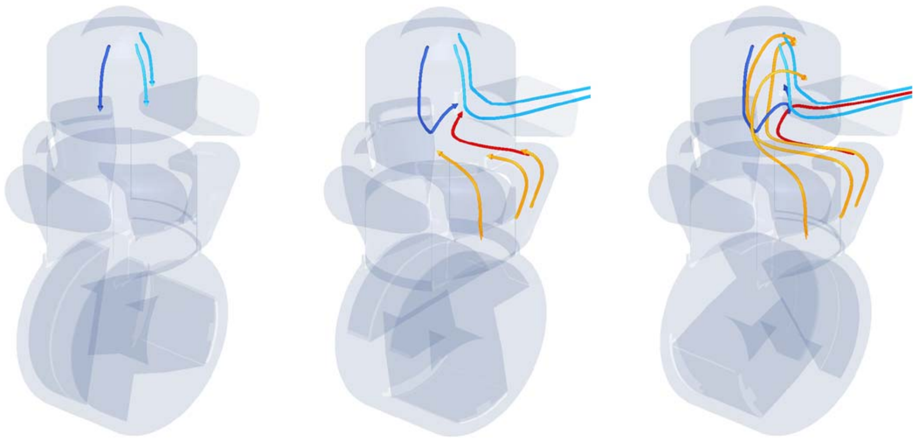


Fig. 14 Visualization of the flow structure: Particle traces at 100° (*left*), 170° (*middle*), and 240° crankshaft angle atdc (*right*). Exhaust gas particles (in the cylinder) and fresh gas particles (from the transfer ports) on their way through the engine, the leftmost exhaust particle does not leave the cylinder, the most advanced fresh gas particle reaches the outlet port (short-circuiting)

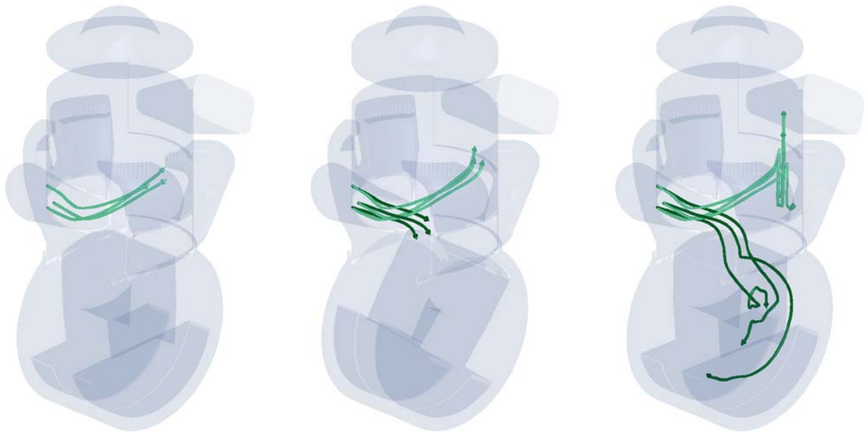


Fig. 15 Visualization of the flow structure: Particle traces at 0° (*left*), 40° (*middle*), and 360° crankshaft angle atdc (*right*). The darker particles start at identical positions but 75° crank angle later. Their path is completely distinct from the earlier ones

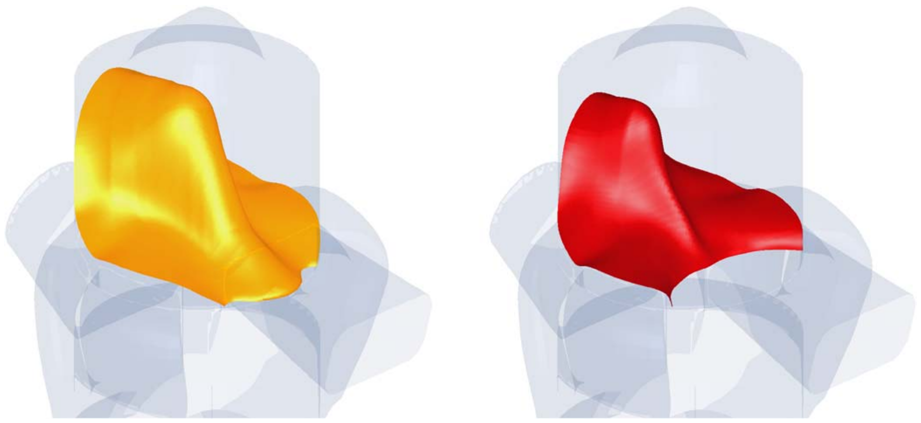


Fig. 16 Visualization of the flow structure: iso-surfaces of fresh gas concentration at 180° crank angle atdc for values of 0.5 (*left*), and 0.8 (*right*)

development of the massflow of the fresh and exhaust gas at the transfer ports and the outlet port is very useful.

In the following diagram in Fig. 20, the scavenge ratio value is measured during one rotation. It is therefore a temporal parameter and does not represent different engine loads. What can be seen in the trapping efficiency graph in Fig. 20 is that a short-circuit flow reaches the outflow port very fast. The trapping efficiency falls rapidly to a value of about .976 due to this short-circuiting. (For a detailed investigation to this short-circuiting see Sect. 9.8.) The vertical part at the upper end of the scavenge ratio in the diagram is caused by the closing of the transfer ports prior to the outlet port.

Figure 21 displays the two situations at the transfer port–cylinder connection (left) and the outlet port entry (right). Here a more detailed view of the development of the massflow can be obtained. On the right diagram one can see the loss of fresh charge through the outlet port which has a peak at 194° crank angle atdc (red curve).

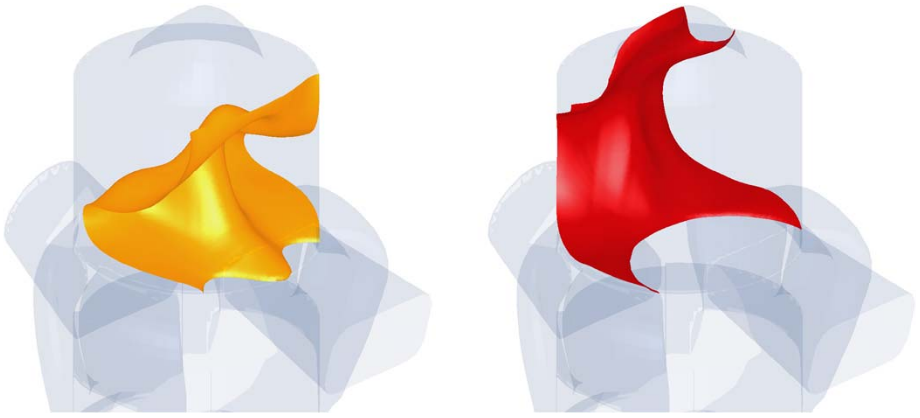


Fig. 17 Visualization of the flow structure: iso-surfaces of fresh gas concentration at 220° crank angle atdc for values of 0.5 (*left*), and 0.8 (*right*)

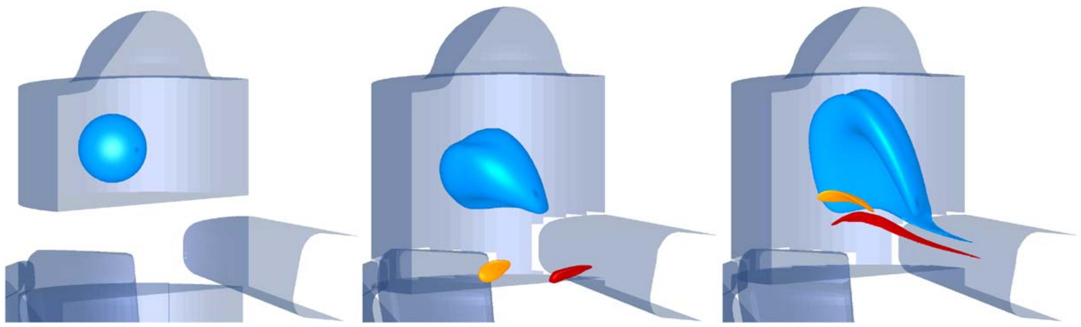


Fig. 18 Visualization of the flow structure: Objects moving with the flow at 90° (*left*), 180° (*middle*), and 210° crankshaft angle atdc (*right*). The ball in the cylinder moves with the exhaust gas flow, the right small ball starts in the main transfer port, and the left small one in the auxilliary transfer port

9.7 The Effect of the Crankshaft Rotation

The crankshaft rotation has a profound effect on the flow field in the crankcase itself (compare Fig. 15 with Fig. 22). But the interesting part is the flow field in the cylinder at the scavenging process. Is the rotating crankshaft changing the characteristic values which determine the quality of the engine, namely scavenge ratio (SR) and trapping efficiency (TE) as defined in Sect. 2.3? Thus, does the engineer have to worry about an altered scavenging process due to design changes made in the crankcase?

9.7.1 Setup of the Comparison

We examine this question with our real engine geometry. For this, the engine is simulated with the data as given in Sects. 9.1, 9.2, and 9.3. However, the crankshaft of the engine is fixed, and not rotating, in this second simulation. The piston motion is not affected by this procedure.

9.7.2 Results

In Fig. 23 it is made clear that there is hardly any difference between the scavenging process with the rotating crankshaft and the one with fixed crankshaft. Thus, the big difference in the flow field in the crankcase has almost no influence on the scavenging process in the cylinder. The only driving force for the scavenging seems to be the pressure difference between crankcase and cylinder which is not altered by the crankshaft rotation. This might possibly change for configurations for which the port window of the transfer ports is partly shadowed by the crankshaft at certain angles. This obstruction for the flow from crankcase to cylinder can have a measurable influence on the scavenging process. But in our geometry this is not the case.

9.8 The Study of Short-Circuiting

The problem of short-circuiting is the main reason for the bad exhaust characteristic in a two-stroke engine. Unburnt hydrocarbons escape the combustion process and reach the exhaust, polluting the environment. Thus, the main task of the engineer is to minimize this short-circuiting. Therefore, it is advantageous to know which transfer port causes how much loss of fresh gas.

9.8.1 Enhanced Fresh-Gas Tracking

In order to derive the origin of the escaped fresh gas, four different fresh gas species σ_i, $1 \leq i \leq 4$, that do not interact

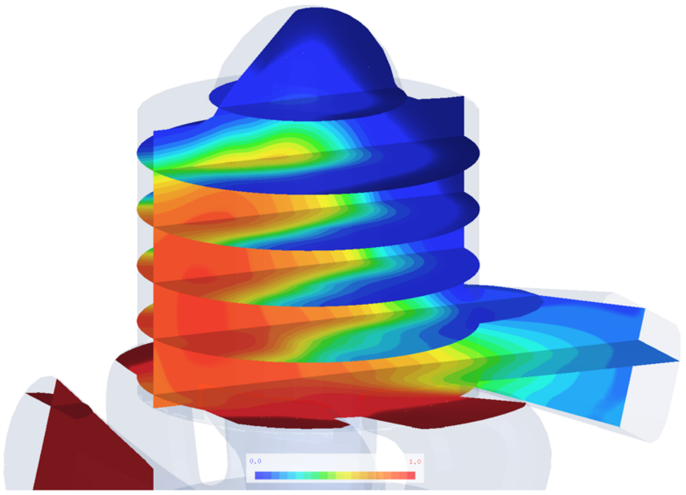
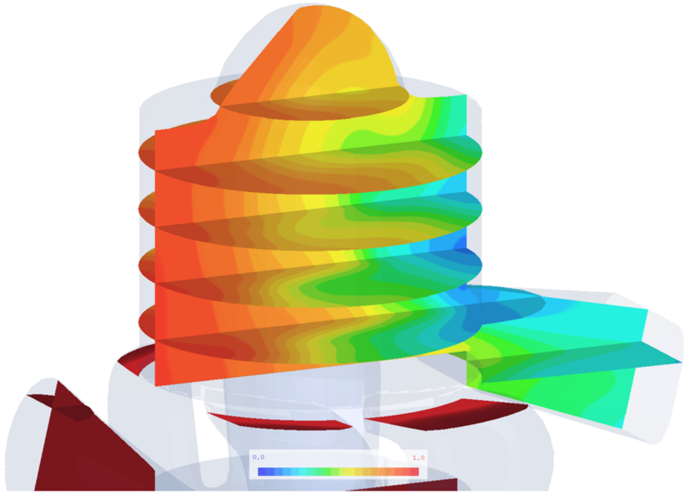



Fig. 19 Visualization of the flow structure: horizontal clipping planes of iso-lines of the fresh gas concentration at 180° (*top*) and 220° (*bottom*) crank angle atdc

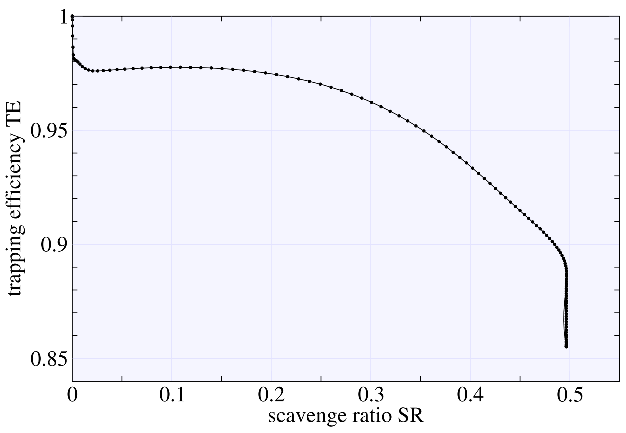


Fig. 20 Characteristical diagram: the scavenge ratio SR–trapping efficiency TE graph

with each other, are introduced. This extension of the numerical scheme as described in Sect. 4, is straightforward. In the crankcase, the fresh charge is composed homogeneously of these four fresh gas species ($\sigma_i := \frac{1}{4}\sigma,\ 1 \le i \le 4$). But as soon as a transfer port is reached, all the fresh gas is shifted to one species depending on the transfer port: $\sigma_j := \sum_{i=1}^{4} \sigma_i$ where j depends on the port number. Now, if the fresh gas enters the cylinder it can be exactly traced back to its port of entry (see Fig. 25). If some fresh gas reaches the outlet duct, it can be quantitatively attributed to the single transfer ports (as shown in Fig. 24).

This fresh-gas tracking is applied to the engine simulation with the data as given in the Sects. 9.1, 9.2, and 9.3.

9.8.2 Results

In Fig. 25 it can be guessed that the main transfer ports are the principal source of short-circuit losses. Design recommendations to improve the scavenging process might be derived from these figures.

In the diagrams shown in Fig. 24 a further detailed analysis of the scavenging losses is possible. These losses can be classified into three distinct mechanisms (cf. [4]). The first one is the short-circuit loss. This occurs at the beginning

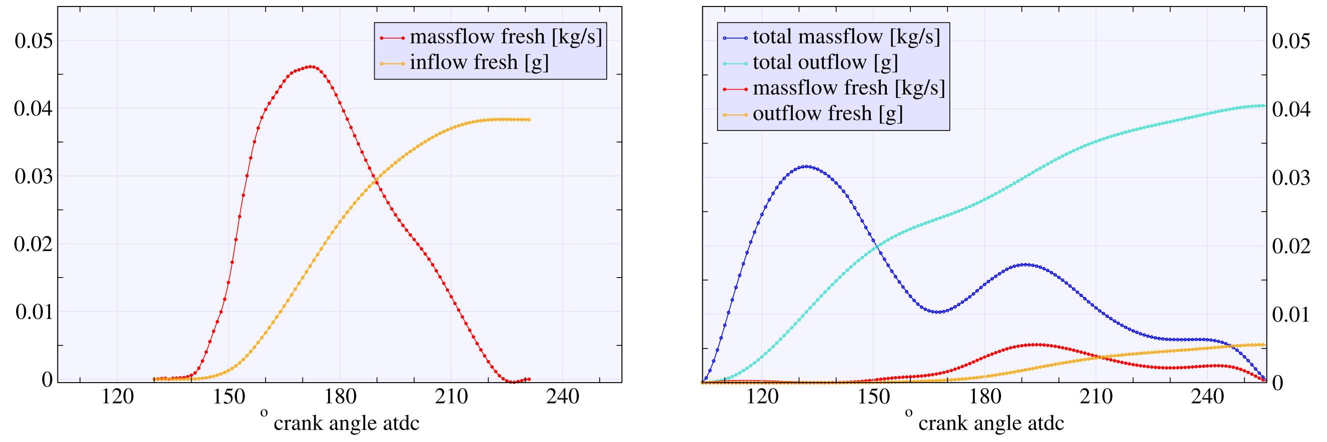


Fig. 21 Characteristical diagrams: the massflow and integrated massflow of fresh gas at the transfer port–cylinder connection (*left*), and the massflow and integrated massflow of the total charge (*upper curves*) and of the fresh charge (*lower curves*) at the outlet port entry (*right*)

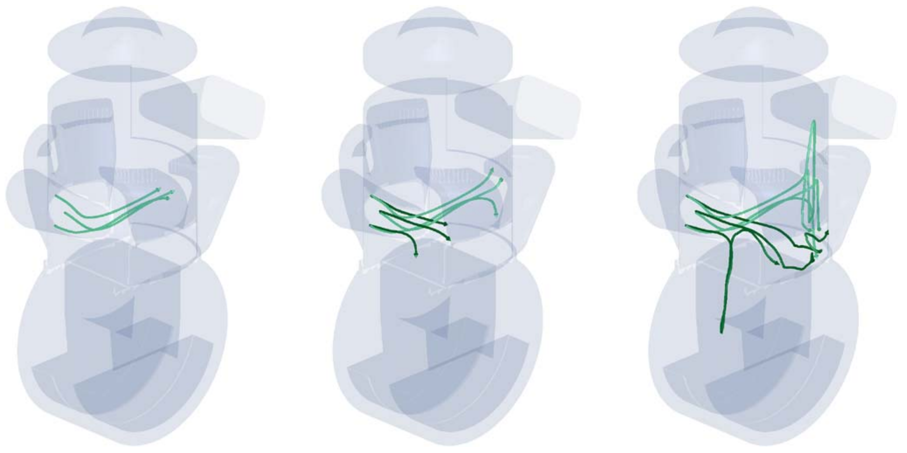


Fig. 22 Effect of crankshaft rotation: Particle traces at 0° (*left*), 40° (*middle*), and 360° crankshaft angle atdc (*right*) with non-rotating crankshaft. The darker particles start at identical positions but 75° crank angle later. Their path is particularly strongly influenced by the fixed crankshaft

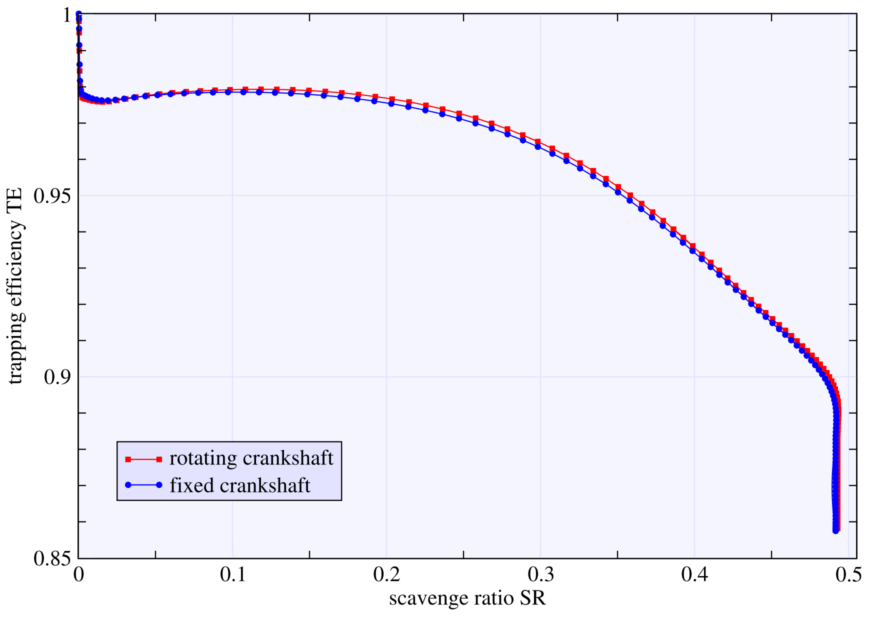


Fig. 23 The effect of the crankshaft rotation: difference in the scavenge ratio–trapping efficiency graph

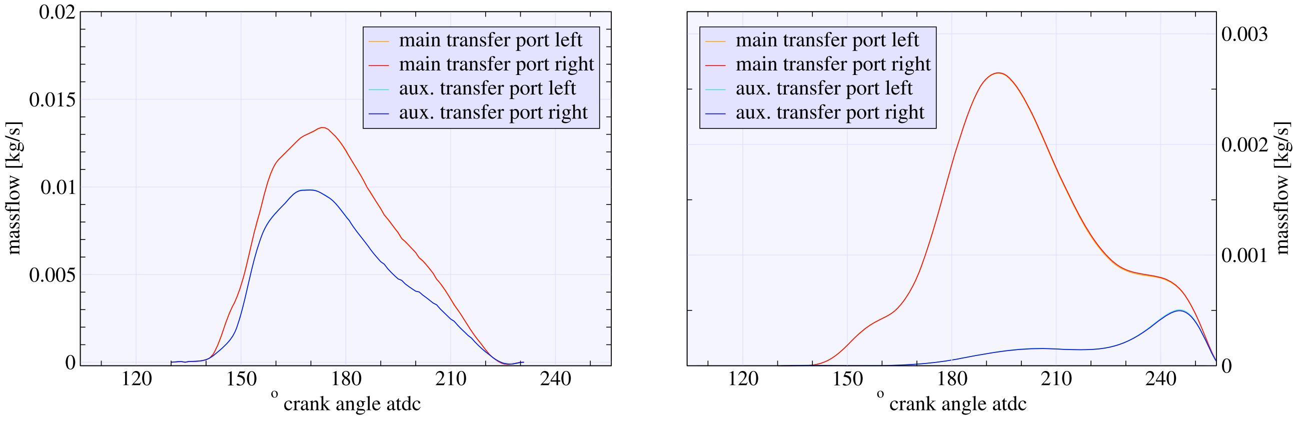
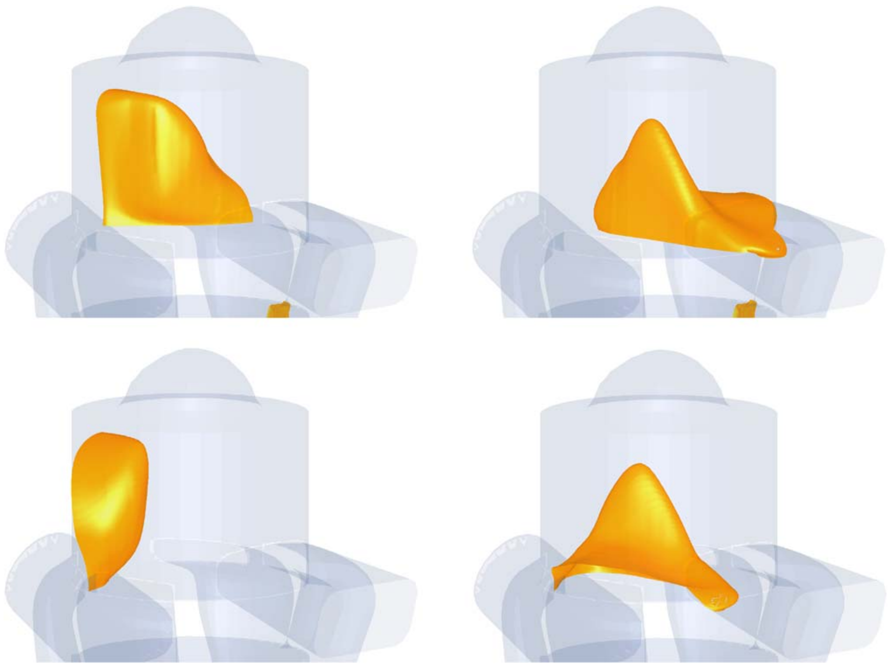


Fig. 24 Fresh-gas tracking: the fresh charge massflow at the transfer port–cylinder connection (*left*) and at the exhaust port entry (*right*) attributed to the single transfer ports

Fig. 25 Fresh-gas tracking: shown is the iso-surface of 0.3 fresh-gas concentration coming from the single transfer ports



of the scavenging. In the right graph of Fig. 24 the main transfer ports (red/orange curve) account for this loss up to 165° crank angle atdc. The second mechanism is the central loss between 165° and 225° crank angle atdc. Here the main transfer port also causes this kind of scavenging loss. And finally, from 225° crank angle atdc until the closing of the exhaust port, the loop scavenge loss, produced by all transfer ports, is the main cause for fresh charge loss at the end of the scavenging process (barely visible in the bottom left image of Fig. 17).

What also can be seen, are the completely symmetrical losses of fresh charge.

10 Summary and Outlook

10.1 Conclusion

A software package has been implemented, which is able to efficiently simulate and analyze the viscous flow through a time-dependent two-stroke engine geometry.

The used numerical scheme is based on the Navier–Stokes equations, which describe a viscous time-dependent fluid and include thermal conductivity. In order to apply this scheme to the real-world problem of the two-stroke engine in motion, certain mesh-related algorithms had to be realized. It is very difficult to represent the complex geometry of the engine by one single mesh. Thus, the assembling of the mesh from several partial meshes is made possible by the grid merging technique. The piston motion is realized using the snapper algorithm. This algorithm was extended to handle the lower part of the piston as well. The rotational motion of the crankshaft is treated with the help of the curved interface method.

Because an accurate simulation has a big need for resources, it was essential to incorporate several techniques to reduce the computational time. A temporally consistent adaptive local time-stepping method is very efficient if meshes with different size elements are used, as is the case in real-world applications. The dynamic local mesh adaption with all the special mesh considerations, as the grid merging, the moving parts of the mesh, and a boundary fitting of refined elements, saves computational time by refining elements in regions where a higher resolution is needed. The temporal convergence to a periodical solution is accelerated by an enhanced initial data approach. Finally, the software was parallelized for the shared memory architecture using the OpenMP standard. Dynamic load balancing is achieved by an extended partitioning algorithm to account for the mesh manipulating routines. The different parts of the developed software have been rigorously validated on several test problems. The efficiency of the local time-stepping, the adaptive mesh refinement, and the parallelization have been proved by the simulation of an existing two-stroke engine.

Three important questions of the engineer have been investigated with the resulting software package. The simulation has been analyzed by the developed visualization techniques, and the characteristic exhaust data has been presented.

Thus, a software tool has been created that can be used to make an in depth analysis of the flow through a given two-stroke engine geometry within a reasonable time. It is also possible to quantitatively compare two or more geometries regarding their efficiency and exhaust characteristics.

10.2 Further Steps

There are several ways to further improve the quality of the simulations. What can already be included in the simulation, is the treatment of ring crevice volumes and rounded port corners (cf. [3, 47]). If the engine geometry incorporates these features, the finer meshes can be constructed accordingly. No changes need to be made in the actual software.

To enhance the properties of the two-stroke engine, further studies of the impact of an asymmetrical transfer port, especially in its upper part, could be conducted. For this analysis, the software can be used without modification.

One further promising approach to reduce the short-circuiting in the scavenge process is the “layering” of the fresh gas supply. At the onset of the scavenge process, pure air displaces the exhaust gas. Only later in the process the fuel charged fresh gas enters the cylinder, thus decreasing the possibility of a short-circuit flow. Two possibilities are thinkable to integrate this approach into the simulation. The underlying engine geometry could be meshed to incorporate the supply duct of fresh air to the transfer ports, or the composition of the gas within the transfer ports could be modified prior to transfer port opening. An adaption of the code is easily feasible.

To increase the accuracy of the numerical (discrete) approximation of the (continuous) Navier–Stokes equations, new higher order methods have been developed, namely the Discontinuous Galerkin methods, which seem to be very promising, regarding their efficiency. But higher order schemes are more time consuming than first order schemes on the same mesh. Their advantage is the accelerated convergence on finer meshes. But, generally, a certain mesh size has to be reached in order to experience the improved accuracy of higher order schemes (cf. [83, 84]). This is especially true for calculations with the complex mesh routines that need to be raised to this higher order as well, which adds to the cost of such an approach.

A further extension of the used numerical scheme would be the application of a turbulence model (cf. e.g. [85]). As the computational power is, despite our accelerating methods, far from performing a *direct numerical simulation* (DNS) of the small scale turbulences (at Reynolds numbers as high as 10^5 in our application), a turbulence model would account for these small scale effects on the flow. Unfortunately, up to now, no rigorous approach has been proposed to deal with this problem in an exhaustive manner. But nevertheless, to include a heuristic turbulence model would be an improvement for the accuracy of the simulation.

And finally, in order to extend the software towards a tool for analyzing the combustion process, and optimizing the combustion chamber, a detailed three-dimensional combustion model could be integrated. Even though this is a difficult task due to the different time scales in combustion processes, this would increase the versatility of this software package even more.

Acknowledgements This thesis was partly supported by the BMBF, the ‘Bundesministerium für Forschung und Technologie’, under the reference number 03 KRM 1 FR, and the ‘Graduiertenkolleg Nichtlineare Differentialgleichungen: Modellierung, Theorie, Numerik, Visualisierung’ of the German Research Association DFG, the ‘Deutsche Forschungsgemeinschaft’.

The author wants to thank especially Prof. Dr. Dietmar Kröner for pointing towards this very interesting topic and support during this work.

References

1. Leep LJ, Strumolo GS, Griaznov VL, Sengupta S, Brohmer AM, Meyer J (1994) CFD investigations of the scavenging process in a

two-stroke engine. SAE Paper No 941929, Society of Automotive Engeneers, Warrendale, PA
2. Mitianiec W (2002) Analysis of loop scavenging process in a small power SI two-stroke engine. SAE Paper No 2002-01-2181, Society of Automotive Engeneers, Warrendale, PA
3. Raghunathan BD, Kenny RG (1997) CFD simulation and validation of the flow within a motored two-stroke engine. SAE Paper No 970359, Society of Automotive Engeneers, Warrendale, PA
4. Rosskamp H, Klimmek A, Pretzsch P, Mugele M (2001) Scavenge Loss mechanisms and their driving forces in loop-scavenged high-performance two-stroke engines. SAE Paper No 2001-01-1826/4247, Society of Automotive Engeneers, Warrendale, PA
5. Werner A (2000) 3D simulation of instationary turbulent flow and combustion in internal combustion engines. In: Krause E, Jäger W (eds) High performance computing in science and engineering. Springer, New York
6. Zeng Y, Strauss S, Lucier P, Craft T (2004) Predicting and optimizing two-stroke engine performance using multidimensional CFD. SAE Paper No 2004-32-0039, Society of Automotive Engeneers, Warrendale, PA
7. Blair GP (1996) Design and simulation of two-stroke engines. Society of Automotive Engineers, Inc
8. Ferziger J, Perić M (1997) Computational methods for fluid dynamics. Springer, Berlin
9. Truckenbrodt E (1989) Fluidmechanik, Band I. Grundlagen und elementare Strömungsvorgänge dichtebeständiger Fluidet. Springer, Berlin
10. Godlewski E, Raviart P-A (1996) Numerical approximation of hyperbolic systems of conservation laws. Springer, New York
11. Kröner D (1997) Numerical schemes for conservation laws. Wiley&Teubner, Stuttgart
12. Bardos C, Leroux AY, Nedelec JC (1979) First order quasilinear equations with boundary conditions. Commun Partial Differ Equ 4(9):1017–1034
13. Otto F (1996) Initial-boundary value problem for a scalar conservation law. C R Acad Sci Paris Sér I 322(8):729–734
14. Schlichting H, Gersten K (1997) Grenzschicht-theorie. Springer, Berlin
15. Klassen L, Klimmek A, Kröner D, Trescher D (2003) Numerical optimization of scavenging in two-stroke engines with transfer ducts, an exhaust port and a moving piston. In: Jäger W, Krebs H-J (eds) Mathematics, key technology for the future. Springer, New York
16. Verein Deutscher Ingenieure (1997) VDI–Wärmeatlas: Berechnungsblätter für den Wärmeübergang. Hrsg. VDI–Gesellschaft Verfahrenstechnik und Chemieingenieurwesen. Springer, Berlin
17. Singer IM, Thorpe JA (1967) Lecture notes on elementary topology and geometry. Springer, New York
18. Liou M-S, Steffen CJ (1993) A new flux splitting scheme. J Comput Phys 107:23–39
19. Wada Y, Liou M-S (1997) An accurate and robust flux splitting scheme for shock and contact discontinuities. SIAM J Sci Comput 18(3):633–657
20. Egelja A, Kröner D, Schwörer R (1999) Adaptive grids for time dependent conservation laws: theory and applications in CFD. In: High performance scientific and engineering computing, Munich, 1998. Lect notes comput sci eng, vol 8. Springer, Berlin, pp 25–37
21. Egelja A, Kröner D, Schwörer R, Lanson N, Mancip M, Vila JP (1998) Combined finite volume and smoothed particle method. In: Numerical flow simulation, I, Marseille, 1997. Notes numer fluid mech, vol 66. Vieweg, Braunschweig, pp 50–74
22. Krause E, Meinke M (2000) CFD–applications on NEC SX-4. In: Krause E, Jäger W (eds) High performance computing in science and engineering. Springer, Berlin
23. Jenny P (1997) On the numerical solution of the compressible Navier–Stokes equations for reacting and non–reacting gas mixtures. Doctoral Dissertation, ETH Zürich
24. Klassen L, Kröner D, Schott P (2003) Finite volume method on unstructed grids in 3D with applications to the simulation of gravity waves. Met Atm Phys 82(1–4):259–270
25. Schott P (2001) Dreidimensionale Strömungssimulation in der Wettervorhersage. Diplomarbeit, Mathematisches Institut, Universität Freiburg
26. Schwörer R (1997) Entwicklung und Parallelisierung von Finite Volumen Verfahren zur Lösung der kompressiblen Navier–Stokes Gleichungen in 2–D. Diplomarbeit, Mathematisches Institut, Freiburg
27. Wierse M, Kröner D, Müller A, Schupp B, Schwörer R (1998) Simulation of a 3-D piston driven flow. In: Friedrich R et al (eds) Computation and visualization of three-dimensional vortical and turbulent flows. Proc of the 5th CNRS-DFG workshop on numerical flow simulation, München, Germany, Dec 6–7, 1996. Notes numer fluid mech, vol 64. Vieweg, Wiesbaden, pp 333–349
28. Wierse M, Kröner D (1996) Higher order upwind schemes on unstructured grids for the nonstationary compressible Navier–Stokes equations in complex time-dependent geometries in 3D. Preprint Nr 2, Universität Freiburg
29. Wierse M (1995) Higher order upwind schemes on unstructured grids for the compressible Euler equations in timedependent geometries in 3D. Doctoral Dissertation, Mathematisches Institut, Freiburg, SFB256 Preprint 393, Bonn
30. Demirdžić I, Perić M (1990) Finite volume method for prediction of fluid flow in arbitrarily shaped domains with moving boundaries. Int J Numer Methods Fluids 10:771–790
31. Hirsch C (1992) Numerical computation of internal and external flows, vol II: computational methods for inviscid and viscous flows. Wiley, New York
32. Poinsot TJ, Lele SK (1992) Boundary conditions for direct simulations of compressible viscous flows. J Comput Phys 101:104–129
33. Geßner T (2001) Dynamic mesh adaption for supersonic combustion waves modeled with detailed reaction mechanisms. Doctoral Dissertation, Mathematisches Institut, Freiburg
34. Hirsch C (1992) Numerical computation of internal and external flows, vol I: fundamentals of numerical discretization. Wiley, New York
35. LeVeque RJ (1990) Numerical methods for conservation laws. Birkhäuser, Basel
36. Geiben M, Kröner D, Rokyta M (1993) A Lax-Wendroff type theorem for cell centered, finite volume schemes in 2D. Preprint 278, SFB 256, Bonn
37. Geßner T (1994) Zeitabhängige Adaption für Finite Volumen Verfahren höherer Ordnung am Beispiel der Euler–Gleichungen der Gasdynamik. Institut für Angewandte Mathematik, Diplomarbeit, Universität Bonn
38. Mavriplis DJ, Jameson A (1989) Multigrid solution of the Navier–Stokes equations on triangular meshes. AIAA Paper 89-0120, AIAA 27th Aerospace Sciences Meeting, Reno
39. Coquel F, Perthame B (1998) Relaxation of energy and approximate Riemann solvers for general pressure laws in fluid dynamics. SIAM J Numer Anal 35(6):2223–2249
40. Dedner A (2003) Solving the system of radiation magnetohydrodynamics for solar physical simulations in 3d. Doctoral Dissertation, Mathematisches Institut, Freiburg
41. Fujii K (1995) Unified zonal method based on the fortified solution algorithm. J Comput Phys 118:92–108
42. Steger JL, Benek JA (1987) On the use of composite grid schemes in computational aerodynamics. Comput Methods Appl Mech Eng 64:301–320
43. Peterson NA (1999) Hole-cutting for three-dimensional overlapping grids. SIAM J Sci Comput 21(2):646–665
44. Chesshire G, Henshaw WD (1994) A scheme for conservative interpolation on overlapping grids. SIAM J Sci Comput 15(4):819–845

45. Hirt CW (1974) An arbitrary Lagrangian–Eulerian computing method for all flow speeds. J Comput Phys 14:227–253
46. Probert J, Hassan O, Peraire J, Morgan K (1988) Transient adaptive methods for moving boundary problems. In: Gruber (ed) Proceeding of the 5th international symposium on numerical methods in engineering
47. Amsden AA, O'Rourke PJ, Butler TD (1992) Comparisons of computed and measured three-dimensional velocity fields in a motored two-stroke engine. SAE Paper No 920418, Society of Automotive Engeneers, Warrendale, PA
48. Fumeaux C, Baumann D, Leuchtmann P, Vahldieck R (2004) A generalized local time-step scheme for efficient FVTD simulations in strongly inhomogeneous meshes. IEEE Trans Microw Theory Technol 52(3):1067–1076
49. Olson KM, MacNeice P, Fryxell B, Ricker P, Timmes FX, Zingale M (2000) PARAMESH: a parallel, adaptive mesh refinement toolkit and performance of the ASCI/FLASH code. In: Proceeding of the AAS 195th meeting
50. Hwang CJ, Wu SJ (1992) Global and local remeshing algorithms for compressible flows. J Comput Phys 102:98–113
51. Vilsmeier R, Hänel D (1993) Adaptive methods on unstructured grids for Euler and Navier–Stokes equations. Comput Fluids 22(45):485–499
52. Bänsch E (1989) Local mesh refinement in 2 and 3 dimensions. Report 6, SFB 256, Bonn
53. Bänsch E (1991) An adaptive finite-element strategy for the three-dimensional time-dependent Navier–Stokes equations. J Comput Appl Math 36:3–28
54. Grape-Team. (1999) GRAPE GRAphics programming environment manual. Available from WWW: www.mathematik.uni-freiburg.de/IAM/Research/grape/DOC/HTML/manual.html
55. Neubauer R, Ohlberger M, Rumpf M, Schwörer R (1997) Efficient visualization of large-scale data on hierarchical meshes. In: Visualization in scientific computing '97. Springer, New York, pp 125–137
56. Ohlberger M, Rumpf M (1997) Hierarchical and adaptive visualization on nested grids. Computing 59(4):365–385
57. Wierse M, Rumpf M (1992) GRAPE, Eine interaktive Umgebung für Visualisierung und Numerik. In: Informatik, Forschung und Entwicklung, vol 7, pp 145–151
58. Kallinderis Y (1996) Grid adaptation by redistribution and local embedding. In: Computational fluid dynamics. von Karman Institute for Fluid Dynamics, Lecture Series 1996-06
59. Kröner D, Ohlberger M (2000) A posteriori error estimates for upwind finite volume schemes for nonlinear conservation laws in multi dimensions. Math Comput 69:25–39
60. Ohlberger M (2001) A posteriori error estimate for finite volume approximations to singularly perturbed nonlinear convection–diffusion equations. Numer Math 87(4):737–761
61. Ohlberger M, Vovelle J (2003) Error estimate for the approximation of non-linear conservation laws on bounded domains by the finite volume method. Preprint 03-32, Mathematisches Institut, Freiburg
62. Nessyahu H, Tassa T, Tadmor E (1994) The convergence rate of Godunov type schemes. SIAM J Numer Anal 32(1):1–16
63. Muzaferija S (1994) Adaptive finite volume method for flow prediction using unstructured meshes and multigrid approach. Doctoral Dissertation, Department of Mechanical Engineering, University of London
64. Champier S (1998) Error estimates for the approximate solution of a nonlinear hyperbolic equation with source term given by the finite volume scheme. Preprint, CNRS UMR 5585, Université de Saint–Etienne, available from WWW: numerix.univ-lyon1.fr/publis/publiv/1997/publis.html
65. Klöfkorn R, Kröner D, Ohlberger M (2002) Local adaptive methods for convection dominated problems. Int J Numer Methods Fluids 40(1–2):79–91
66. Geßner T, Kröner D (2001) Dynamic mesh adaption for supersonic reactive flow. In: Proceedings: hyperbolic problems: theory, numerics, applications, eighth international conference, Magdeburg, 2000, pp 415–424
67. Geßner T, Kröner D (2001) Godunov type methods on unstructured grids and local mesh refinement. In: Toro EF (ed) Godunov methods theory and applications, New York, pp 527–548
68. van der Velde E (1994) Concurrent scientific computing. Springer, New York
69. Schiano P, Matrone A (1995) Parallel CFD applications: experiences on scalable distributed multicomputers. In: Satofuka N, Periaux J, Ecer A (eds) Parallel computational fluid dynamics, new trends and advances. North-Holland, Amsterdam
70. Schupp B (1999) Entwicklung eines effizienten Verfahrens zur Simulation kompressibler Strömungen in 3D auf Parallelrechnern. Doctoral Dissertation, Mathematisches Institut, Freiburg
71. Tanenbaum AS, Goodman J (1999) Computerarchitektur. Prentice-Hall, München
72. Dedner A, Rohde C, Schupp B, Wesenberg M (2004) A parallel, load balanced MHD code on locally adapted, unstructured grids in 3D. Comput Vis Sci 7(2):79–96
73. MPI-Team (1997) MPI: a message-passing interface standard (1994), and MPI-2: extensions to the message-passing interface. Available from WWW: www.mpi-forum.org/docs/docs.html
74. OpenMP-Team (2002) Official OpenMP specifications, C/C++ version 2.0. Available from WWW: www.openmp.org/drupal/mp-documents/cspec20.pdf
75. Kallinderis Y (1996) Domain partitioning and load balancing for parallel computation. In: Computational fluid dynamics. von Karman Institute for Fluid Dynamics, Lecture Series 1996-06
76. Trescher D (2005) Development of an efficient 3-D CFD software to simulate and visualize the scavenging of a two-stroke engine. Doctoral Dissertation, Mathematisches Institut, Freiburg
77. ICEM CFD HEXA, ICEM CFD Engineering (2004)
78. Trescher D (2000) Theorie und Numerik Transparenter Randbedingungen. Diplomarbeit, Mathematisches Institut, Universität Freiburg
79. Becker J, Preuser T, Rumpf M (2000) PDE methods in flow simulation post processing. Comput Vis Sci 3(3):159–167
80. Ohlberger M, Rumpf M (1999) Adaptive projection operators in multiresolution scientific visualization. IEEE Trans Vis Comput Graph 5(1):74–94
81. Rumpf M, Schupp B (1995) Visualization of parallel data based on procedural access. In: Proc MathVis Conference, Berlin
82. Jante A (1968) Scavenging and other problems of two-stroke spark-ignition engines. SAE Paper No. 680468, Society of Automotive Engeneers, Warrendale, PA
83. Klassen L, Kröner D (2000) Discretization of higher order for conservation laws on nonconformal unstructured rectangular grids. Manuscript
84. Herbin R, Kröner D (2002) Finite volumes for complex applications III. Hermes Penton Science, London
85. Klöker J (1992) Numerische Simulation einer dreidimensionalen, kompressiblen, reibungsbehafteten Strömung im Zylinder eines Modellmotors. Doctoral Dissertation, RWTH Aachen