Towards the Implementation of a New Multigrid Solver in the DNS Code FS3D for Simulations of Shear-Thinning Jet Break-Up at Higher Reynolds Numbers

Moritz Ertl, Jonathan Reutzsch, Arne Nägel, Gabriel Wittum, and Bernhard Weigand

Abstract Liquid jet break-up appears in many technical applications, as well as in nature. It consists of complex physical processes, which happen on very small scales in space and time. This makes them hard to capture by experimental methods; and therefore a prime subject for numerical investigations. The state-of-the-art approach combines the Volume of Fluid (VOF) method with Direct Numerical Simulations (DNS) as employed in the ITLR in-house code Free Surface 3D (**FS3D**). The simulation of these jets is dependent on very fine grids, with most of the computational costs incurred by solving the Pressure Poisson Equation. In order to simulate larger computational domains, we tried to improve the performance of FS3D by the implementation of a new multigrid solver. For this we selected the solver contained in the **UG4** package developed by the Goethe Center for Scientific Computing at the University of Frankfurt. We will show simulations of the primary break-up of shear-thinning liquid jets and explain why larger computational domains are necessary. Results are preliminary. We demonstrate that the implementation of UG4 into FS3D provides a noticeable increase in weak scaling performance, while the change in strong scaling is yet detrimental. We will then discuss ways to further improve these results.

M. Ertl (✉)
Institut für Thermodynamik der Luft- und Raumfahrt, Universität Stuttgart, Pfaffenwaldring 31, 70569 Stuttgart, Germany
e-mail: moritz.ertl@itlr.uni-stuttgart.de

J. Reutzsch • B. Weigand
Institut für Thermodynamik der Luft- und Raumfahrt, Universität Stuttgart, Pfaffenwaldring 31, 70569 Stuttgart, Germany
e-mail: jonathan.reutzsch@itlr.uni-stuttgart.de; bernhard.weigand@itlr.uni-stuttgart.de

A. Nägel • G. Wittum
Goethe-Zentrum für Wissenschaftliches Rechnen, Goethe-Universität Frankfurt am Main, Kettenhofweg 139, 60325 Frankfurt am Main, Germany
e-mail: arne.naegel@gcsc.uni-frankfurt.de

W.E. Nagel et al. (eds.), *High Performance Computing in Science and Engineering '17*, https://doi.org/10.1007/978-3-319-68394-2_16

1 Introduction

Liquid jet break-up is the process where a fluid stream is injected into a surrounding medium and thereby disintegrates into many smaller droplets. Liquid jets appear in many technical applications as well as in nature. Well known examples are fuel injection in combustion engines or gas turbines, water or foam jets for fire fighting, irrigation or pesticides in agriculture, and spray painting or ink jet printing. In some cases like spray drying, which is used to produce functional particles with well defined properties, and which is used in the production of pharmaceuticals or in food processing, the injected fluid can additionally show a Non-Newtonian behaviour [8].

Newtonian jet break-up is already under intensive investigation, especially in the context of fuel injection. A very good introduction into the fundamentals of jet break-up is given by Lefebvre [18] and some more recent developments are summarised by Lin and Reitz [20]. When it comes to the investigation of jet break-up, numerical methods are increasingly used in addition to experiments. Numerical methods are very well suited for the analysis of phenomena like jet break-up, where the time scales are fractions of seconds and the investigated structures are often in the order of micrometers, and therefore reliable experiments may be difficult to perform. In order to get rough estimates of jet behaviour, Reynolds-Averaged Navier-Stokes (RANS) methods can be used, for example to aid industrial design as shown by Beau [1]. When more accurate results are required, and even though some very interesting advances using Large Eddy Simulations (LES) have been shown by Hermann [13], the current state-of-the-art numerical approach is Direct Numerical Simulations (DNS). Klein [15] uses the Volume of Fluid (VOF) method to investigate liquid sheets at moderate Reynolds numbers. He analyses the surface deformations and compares the numerical results to experimental data. VOF DNS is also used by Sander and Weigand [34] to investigate the influence of different instability enhancing parameters on liquid sheets and round jets. Pan and Suga [22] combine the Level-Set (LS) method with DNS to investigate the break-up of laminar jets and Shinjo and Umemura [35] use one very highly resolved LS DNS to investigate the influence of vortices in the surrounding gas on the jet surface.

While many numerical studies of Newtonian liquid jet break-up exist, Non-Newtonian behaviour, such as shear-thinning, has been investigated to a lesser extend. Lakdawala [17] did 2D LS numerical simulations of shear dependent Non-Newtonian jets at low Reynolds numbers. A 3D LES with VOF and a power law model of a Non-Newtonian jet at a low Reynolds number has been investigated by Li-Ping [19] in OpenFOAM. The first author of the present report has started to do some three-dimensional numerical investigations into different aspects of the primary jet break-up of shear-thinning fluids using DNS in combination with the VOF method as implemented in the ITLR in-house code **F**ree **S**urface **3D** (**FS3D**). Striving for more accurate simulations and for investigating increasingly higher Reynolds numbers has led to a continuous increase in computational cells—from 33 millions in 2013 [42], over 450 million in 2014 [4] to 750 million during the last year [5]. The authors used the simulations to investigate the influence

of different parameters, such as the Reynolds number, the turbulent intensity, the ambient pressure as well as different concentrations of the shear-thinning solution. This growth of the number of cells has gone in hand with the need for an increasing amount of processors and therefore also an increase in parallel performance. A communication imbalance (which is described in Sect. 2.2) has been identified in the utilised code FS3D. In order to improve on this, the multigrid solver of the software package **UG4** [38] has been integrated into our code. For this solver, scalability for Poisson-type equations in 3D was demonstrated for thousands of processes.

2 Fundamental Method

FS3D is a *Direct Numerical Simulation* (DNS) code for incompressible multiphase flows. It was developed at ITLR (Institute of Aerospace Thermodynamics) in Stuttgart. Due to the use of DNS, very small temporal and spatial scales are resolved, hence, no turbulence modelling is needed. FS3D is based on the *Volume of Fluid* (VOF) method and solves the incompressible Navier-Stokes equations as well as the energy equation with temperature dependent thermo-physical properties. A wide variety of phenomena have been investigated in the last 15 years. These include drop and bubble dynamic processes, for instance droplet deformation [31] or droplet impact onto a thin film [10], furthermore, droplet collisions [33], droplet wall interactions [32], bubbles [41] and also more recently rigid particle interactions [24].

2.1 Mathematical Description

The conservation equations for mass and momentum read accordingly

$$\rho_t + \nabla \cdot [\rho \mathbf{u}] = 0, \tag{1}$$

$$[\rho \mathbf{u}]_t + \nabla \cdot [\rho \mathbf{u}\mathbf{u}] = \nabla \cdot [\mathbf{S} - \mathbf{I}p] + \rho \mathbf{g} + \mathbf{f}_\gamma, \tag{2}$$

where ρ denotes the density, $\mathbf{u}$ the velocity vector, $\mathbf{S}$ the viscous stress tensor, p the pressure, $\mathbf{g}$ the gravitational acceleration and $\mathbf{f}_\gamma$ the body force which is used to model surface tension in the vicinity of the interface. The viscous stress tensor is given by

$$\mathbf{S} = \mu \left[\nabla \mathbf{u} + (\nabla \mathbf{u})^T \right]. \tag{3}$$

The shear-thinning viscosity $\mu = \mu(\dot{\gamma})$ is calculated as a function of the shear rate $\dot{\gamma}$ with the *Carreau-Yasuda model* [37]

$$\frac{\mu(\dot{\gamma}) - \mu_\infty}{\mu_0 - \mu_\infty} = \left[1 + (\tau \dot{\gamma})^a\right]^{\frac{(n-1)}{a}}. \tag{4}$$

The five constants are: the viscosity at high shear rates μ_∞ and at zero shear μ_0, as well as the three model parameters τ, a and n, which are obtained from experimental data.

2.2 Numerical Approach

In FS3D a finite volume method is used to discretise the Navier-Stokes equations. According to the *Marker and Cell method* [11] (MAC), velocities are stored on the cell faces, scalars, for instance pressure or density, at the cell centres, respectively. Additional indicator variables f_i are used to identify different phases. This method is broadly based on the VOF method [14]. The VOF variable f_i is defined as

$$f_i(\mathbf{x},t) = \begin{cases} 0 & \text{in the continuous phase,} \\ ]0,1[& \text{at the interface,} \\ 1 & \text{in the disperse phase.} \end{cases}$$

The index i describes different phases: f_1 for the liquid, f_2 for the vapour and f_3 for the solid phase. Due to the one-field formulation of the VOF method, local variables are defined by the local f value and the corresponding values for, e.g., the pure gaseous (index g) and liquid (index l) phases. Thus, the density and the viscosity read

$$\rho(\mathbf{x},t) = \rho_g + [\rho_l - \rho_g] f(\mathbf{x},t), \tag{5}$$

$$\mu(\mathbf{x},t) = \mu_g + [\mu_l - \mu_g] f(\mathbf{x},t). \tag{6}$$

The volume fraction f is transported across the computed domain by solving the transport equation

$$f_t + \nabla \cdot [f\mathbf{u}] = \mathfrak{S}, \tag{7}$$

where $\mathfrak{S}$ is equal to zero when no source or diffusion terms are considered. It is non zero, when phase change, for instance solidification or evaporation, is taken into account. For further details the reader is referred to [3].

To achieve a successful advection of f a sharp interface is required. For this computation FS3D makes use of the *piecewise linear interface reconstruction* (PLIC) method [29]. The reconstruction of the interface becomes necessary due to the f variable only representing the amount of the disperse phase, for instance the fluid, inside a cell; cf. Fig. 1a. In order to capture the interface a plane orthogonal to the local normal vector $\hat{\mathbf{n}}_\gamma = \nabla f / |\nabla f|$ is placed iteratively into cells containing the interface in such a way, that the volume under the plane is in accordance with the volume fraction f. The vector is defined by the negative gradient of the volume

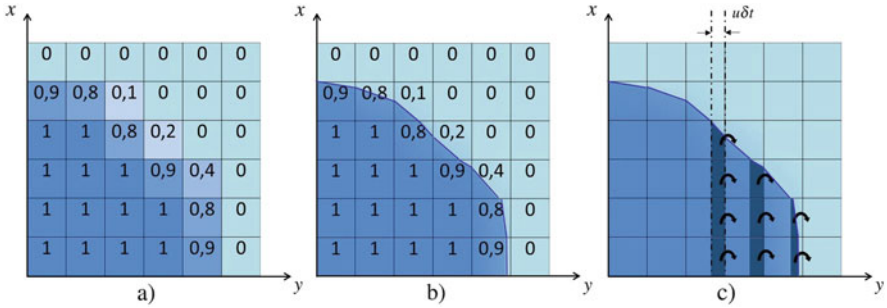

Fig. 1 (**a**) f-Field without interface information; (**b**) Interface reconstruction with the PLIC-method; (**c**) Calculation of the f-flux $u\delta t$ with δt being the timestep from a PLIC reconstructed interface

fraction from the surrounding cells. Figure 1b shows the PLIC reconstruction in 2D. Currently, there are two mass advection methods implemented in the code. On the one hand, three one-dimensional non-conservative transport equations are considered successively. Therefore, interfaces have to be reconstructed three times. With a permutation of the equation sequence [36] and a specific choice for the divergence correction second-order accuracy in space and time can be achieved. The fluid is transported parallel to the velocities perpendicular to the cell faces. This is shown in Fig. 1c. On the other hand, a new formulation [12], which makes a one-step advection possible, was implemented in FS3D. In this method a general, six-faced polyhedron is constructed for a more realistic approximation of the volume fluxes. More details regarding the new method can be found in Reitzle et al. [28]. To compute the surface tension several models are available: the conservative continuous surface stress (CSS) model by Lafaurie et al. [16], the continuum surface force model (CSF) by Brackbill et al. [2], and the balanced force approach by Popinet [23].

FS3D is fully parallelised using MPI and OpenMP; therefore, we are able to perform simulations with very high spatial resolutions, which is crucial for DNS. Heretofore, simulations with a maximum up to two billion computational cells have been conducted at the High Performance Computing Center Stuttgart (HLRS) and during the studies presented here first tests with over eight billion cells have been undertaken successfully. The code is validated and has good performance on the Cray XC40 Hornet supercomputer [25], however, the performance for very high amounts of processes, especially weak scaling, is not yet satisfying. The reasons for this, and a new approach to solve this issue is explained in the following.

Due to the infinitely fast propagation of pressure waves, which occurs because of the incompressibility of the fluids, the momentum equation cannot be solved explicitly. Therefore, the pressure term must be discretised in an implicit way. That leads to the Pressure Poisson Equation, which is defined as

$$\nabla \cdot \left[\frac{1}{\rho(f)} \nabla p \right] = \frac{\nabla \cdot \mathbf{u}}{\Delta t}. \tag{8}$$

To solve the resulting set of linear equations Rieber implemented a multigrid solver into FS3D [30]. A Red Black Gauss Seidel algorithm is used for smoothing and the algorithm can be run in a V- or W-cycle scheme. For time integration schemes a first-order explicit Euler scheme and a second order Runge Kutta scheme are implemented. Solving the pressure correction requires a great amount of the computational time of FS3D: over 70%. This is due to an unfavourable communication imbalance. The whole domain is agglomerated onto a single processor at a certain level of coarsening and all the coarse solutions are gathered on one root process. This means, that all processes need to first send, and then later receive the entire domain to and from a single process. In between this process computes all the coarser levels while all other processes are idling. Therefore, a new approach is taken into consideration, which is explained in the next subsection.

2.3 UG4 and Multigrid Solver

Besides the above described, conventional method for solving the Pressure Poisson Equation, we recently integrated a massively parallel geometric multigrid solver of the software package **UG4** [38] into our code. The software framework UG4 was developed at the Goethe Center for Scientific Computing at the Goethe University in Frankfurt, and it was designed for the solution of partial differential equations. It uses grid-based discretisation methods, such as the finite volume method, and is implemented in C++. A main focus lies on efficient and highly scalable solvers, using algebraic and geometric multigrid methods. UG4 is parallelised using MPI, thus, it is a suitable application for a fast solving of the Pressure Poisson Equation. The MPI calls are subsumed in a separate library called **pcl** (***p**arallel **c**ommunication **l**ayer*), which yields easy structures for graph-based parallelisation. One advantage is that global IDs are redundant by storing parallel copies on each process in a well defined order in interface containers. For further details regarding pcl and the features of UG4 the reader is referred to Vogel et al. [38] or Reiter et al. [27]. Scalability and parallel performance of UG4 was tested extensively [39].

For integration into FS3D the parallel hierarchical multigrid solver of UG4 is of principal interest. A short overview of the algorithm for the parallel version can be found in [40] and is explained in the following: The discretised Pressure Poisson Equation reads $\mathbf{A}_L\mathbf{p}_L = \mathbf{b}_L$, where L denotes the index for the finest Level, $\mathbf{p}_L$ the desired solution, which is computed iteratively. With the defect $\mathbf{d}_L = \mathbf{b}_L - \mathbf{A}_L\mathbf{p}_L$ a multigrid correction $\mathbf{c}_L = \mathbf{M}_L(\mathbf{d}_L)$ is calculated. The multigrid operator $\mathbf{M}_L$ is added to the approximate solution $\mathbf{p}_L := \mathbf{p}_L + \mathbf{c}_L$. Several auxiliary coarse grid matrices $\mathbf{A}_l$, $L_B \leq l \leq L$, are used to compute the correction $\mathbf{c}_L$; L_B is the base Level. Afterwards, the multigrid cycle is defined recursively. By means of a smoothing operator the correction is partly computed with a given defect $\mathbf{d}_l$ on a certain level l. Subsequently, the latter is transferred on the next coarser level applying the algorithm to the restricted defect $\mathbf{d}_{l-1}$. Then the coarse grid correction $\mathbf{c}_{l-1}$ is prolongated to the finer level and added to the correction on

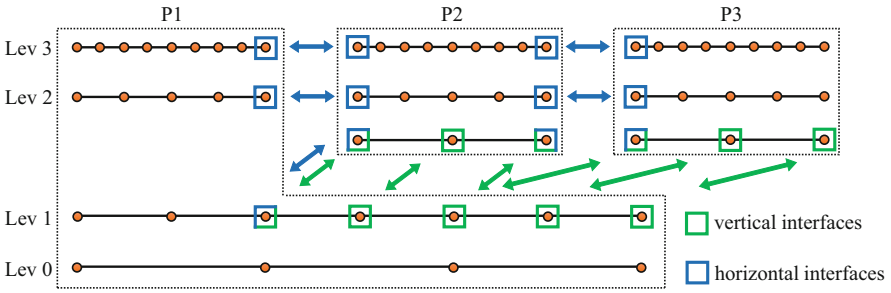

Fig. 2 Schematic illustration of a 1D parallel multigrid hierarchy distributed onto three processes (redrawn from [38]). One top process stores the coarse grid and refines to a certain level. Parallel copies are identified with horizontal (blue), respectively vertical (green) interfaces

level l. Eventually the exact correction $\mathbf{c}_L = \mathbf{A}_l^{-1}\mathbf{d}_l$ is computed on the base level L_B. The corresponding algorithm is described in detail in Reiter et al. [27]. To gain an efficient parallelisation we need an appropriate distributed multigrid hierarchy. Figure 2 (cf. [27]) shows an overview of the hierarchy and the interfaces starting from a coarse grid, which describes the domain loaded onto one process. Then the grid is refined and new levels of the multigrid hierarchy are created until the finest grid level is reached. The latter is distributed to a larger set of processes and communication structures, the vertical interfaces (green), are established. This procedure can be iterated, so that a tree structure of processes holding parts of the hierarchical grid is created. The parallelisation of the transfer between the grid levels is achieved by using the communication structures in vertical direction. The transfer operators can work completely process-locally when no vertical interface is present. On every level simple iterative schemes, also called smoothers, are applied. The horizontal interfaces (blue) are needed for the communication within these multigrid smoothers on each grid level. They are required for the computation of the level-wise correction in a consistent way. Due to the described hierarchical distribution only a smaller number of processes are needed on lower levels and most processes idle. On finer grid levels, however, the major amount of overall runtime is required. Nevertheless, a good computation vs. communication ratio is achieved by means of sophisticated parallel smoothing, prolongation and restriction operations [39].

UG4 is written in C++. Therefore, the integration into FS3D, which is a Fortran code, is done via additional custom Fortran interfaces. The compiled UG4 library is statically linked into FS3D. A new custom module has been written in FS3D for the interaction with UG4. The solver settings and especially the layout of the parallel communication are set up in an initialisation routine, once, at the start of a simulation. During the calculation cycles of the time steps, the matrix and the right hand side of the Pressure Poisson Equation, as well as the boundary layers are set and passed to UG4. There the equation system is solved and the solution (the pressure field) is returned to FS3D.

3 Numerical Setup

All calculations have been performed on a three-dimensional regular grid. The computational domain for the jet simulations is shown schematically in Fig. 3. The nozzle is chosen with a fixed Diameter of $D = 2.5 \times 10^{-3}$ m for the injection of the jet. The domain is set up as a rectangular cuboid with a quadratic base with a width of $W = H = 16\,D$ and a length of $L = 40\,D$. The x-axis points in the direction of the jet injection, the y- and z-direction are orthogonal to it, respectively. We use three different discretisations for the domain. The amount of cells used in x- and in y- and z- direction, as well as the total number of cells are given in Table 1.

In the y-z-plane a grid refinement is applied leading to smaller cells in the center around the jet. The size of the cell edges in this region is also given in Table 1. The boundary condition at the nozzle side (gray) is a no-slip wall with an inflow boundary condition in the center. The latter is circular and has a nozzle diameter of $D = 2.5 \times 10^{-3}$ m. A block profile is chosen as the initial velocity profile at the inlet

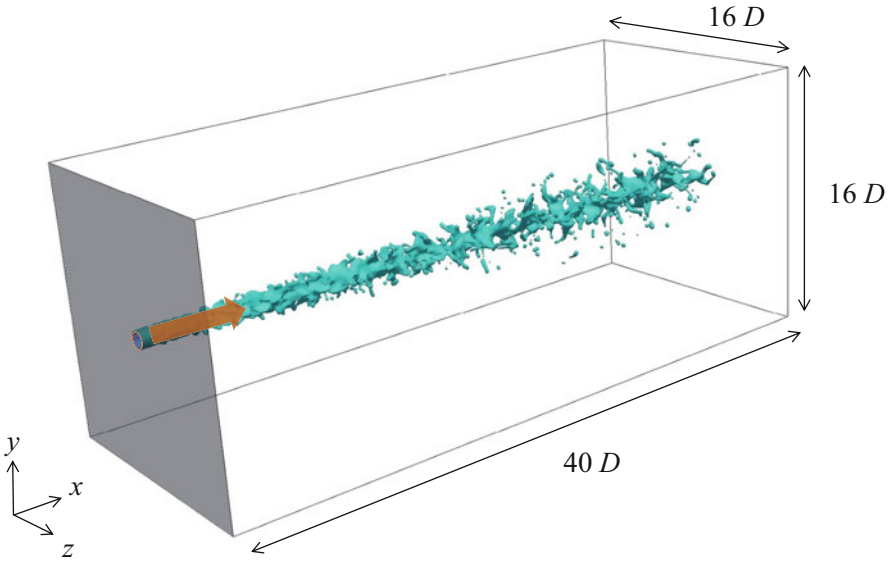

Fig. 3 Computational domain and coordinate system of jet simulation. No slip wall and inflow boundary condition on the left side. Orange arrow indicates the direction of the injection

Table 1 Discretisations of the domain

	Cells in x	Cells in y and z	$\Delta x = \Delta y = \Delta z$ in in the core region (m)	Total number of cells
Fine grid	2304	768	4.34×10^{-5}	1.3×10^{9}
Medium grid	1152	384	8.68×10^{-5}	1.7×10^{8}
Coarse grid	576	192	1.74×10^{-4}	2.1×10^{7}

Table 2 Liquid properties

	Density	Surface tension	Viscosity	Viscosity	Model parameter		
	ρ $(\mathrm{kg/m^3})$	σ (mN/m)	μ_0 (Pa s)	μ_∞ (Pa s)	n (–)	a (–)	τ (–)
Praestol 2500 0.3%	999.4	73.15	0.046	0.004	0.576	1.036	0.157
Air	1.19		1.8×10^{-5}				

with a velocity of $U_0 = 55.25\,\mathrm{m/s}$. The turbulent intensity is set to $Tu = 10\%$ and the turbulent length scale is $L_t = D/8 = 3.125 \times 10^{-5}\,\mathrm{m}$. The five other sides are defined as continuous (Neumann) boundary conditions. Gravitational acceleration is neglected.

As an adequate material with shear-thinning properties an aqueous solution of the polyacrylamide Praestol 2500 at 0.3% weight is chosen. The model parameters as well as the material properties of the solution and of the surrounding air at 20 °C are given in Table 2 [6]. Details regarding the viscosity model can be found in [7].

Therefore, the relevant dimensionless numbers for the simulation, the Reynolds Number and the Ohnesorge Number, can be calculated as

$$Re = \frac{\rho_l U_0 D}{\mu_0} = 3000, \tag{9}$$

and

$$Oh = \frac{\mu_0}{\sqrt{\rho_l \sigma D}} = 0.1. \tag{10}$$

The simulated jet is in the atomisation regime according to the Ohnesorge diagram [18]. The Kolmogorov length scale is calculated as $\lambda_K = 2.6 \times 10^{-5}\,\mathrm{m}$, hence, all three grids are in the order of magnitude of the smallest dissipative length scale necessary for DNS. This is essential to be able to produce physically correct results.

4 Results

We investigate the spatial development of the jet with the medium grid in time, to gain a basic understanding of the break-up process of the jet simulation. For all our analyses we use the dimensionless time

$$t^* = \frac{tU}{D}. \tag{11}$$

The jet from the fine grid at four different times after injection $t^* = 4.4$, $t^* = 15.5$, $t^* = 26.5$ and $t^* = 42.0$ is shown in Fig. 4. Directly after injection at $t^* = 4.4$ the jet core is still cylindrical. We can see the disturbances caused by the nozzle

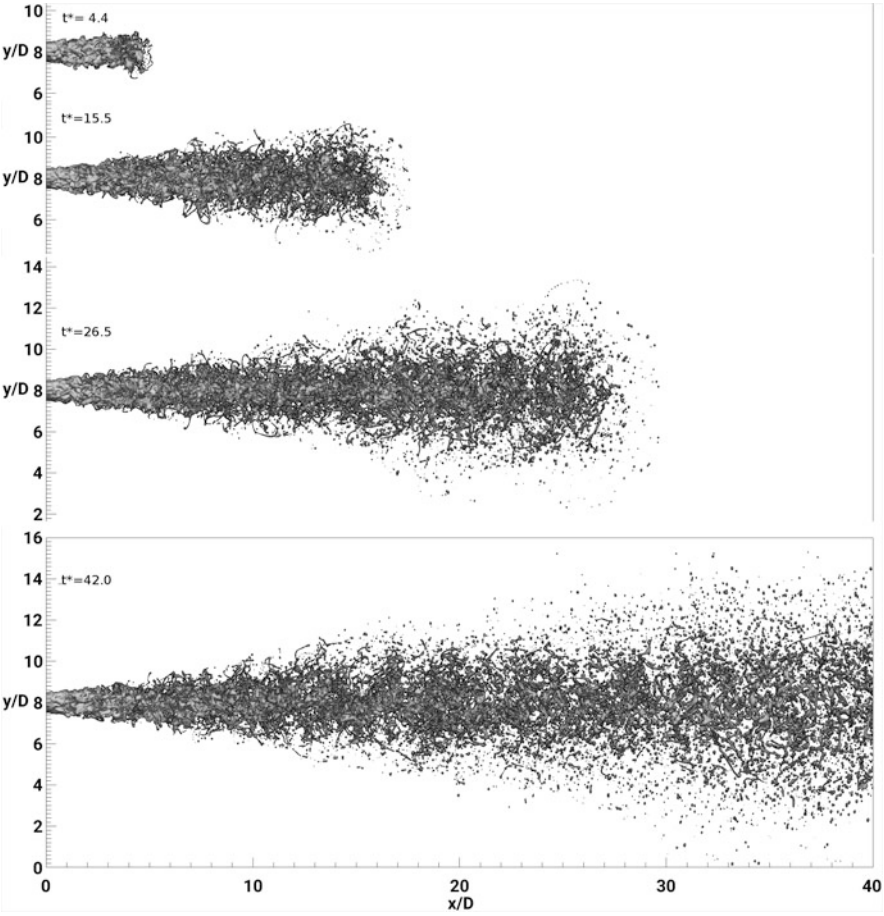

Fig. 4 The spatial development of the jet with the fine grid after injection at dimensionless times $t^* = 4.4$, $t^* = 15.5$, $t^* = 26.5$ and $t^* = 42.0$

turbulence as three-dimensional waves on the jet surface. At $t^* = 15.5$ several of the surface waves have deformed further, due to the interaction with the surrounding air, and have started to detach from the jet core, forming ligaments. Several ligaments have disintegrated further and have separated into droplets. This process continues towards $t^* = 26.5$. At this time we can observe a lot more ligaments and droplets. With increasing length the jet also shows an increasing expansion in radial direction. Particularly the separated droplets are moving away from the jet. This expansion also causes the jet core to become thinner in downstream direction—well visible for example at $x/D = 23$. At $t^* = 42.0$ the jet has reached the end of the computational domain. The disintegration has progressed even further in the regions downstream of $x/D = 21$; no jet core is visible any more. Instead we observe expanding agglomerations of ligaments. The amount of droplets has also further increased. We are observing the onset of atomisation.

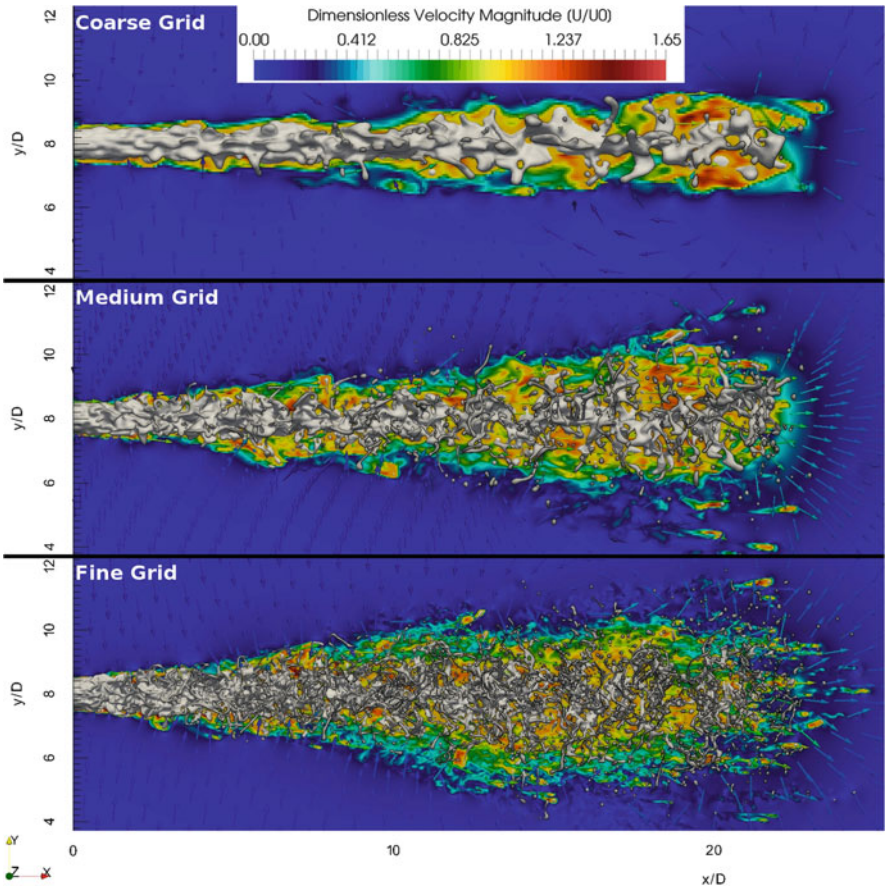

Fig. 5 Visual comparison of the three discretisations at $t^* = 22.1$. From top to bottom: coarse grid, medium grid, fine grid

In order to analyse the influence of the grid we start with a visual analysis of the simulations at $t^* = 22.1$. The impact of the grid size is shown in Fig. 5. The jet, from the simulation with the coarse grid (top), has developed a strongly disturbed surface, but only very few ligaments or droplets are visible. The jet has a cylindrical shape. When we look at the jet resulting from the medium grid, we see a smaller jet core with more ligaments around it. Towards the jet tip the core is getting displaced from the center line and exhibiting a sinusoidal shape. Additionally, towards the tip we can observe an increasing amount of detached ligaments and droplets. The jet shape is still cylindrical close to the injection, but becomes conical further downstream. The jet, which was simulated with the fine grid, looks similar to the jet from the medium grid, but breaks up into even finer structures. A continuous core is only visible up to about $x/D = 10$. Afterwards, the jet starts atomising into drops,

Table 3 Jet properties at $t^* = 22.1$

	Average jet angle (°)	Average droplet diameter (cm)	Droplet count
Fine grid	9.37	1.98×10^{-4}	4465
Medium grid	6.91	3.35×10^{-4}	625
Coarse grid	2.46	5.79×10^{-4}	23

ligaments and larger continuous liquid parts, sometimes connected by ligaments. The jet has a conical shape.

We also used post processing, to identify all separate liquid structures, also at $t^* = 22.1$. We counted the amount of structures and calculated the mean diameter of the structures. We furthermore calculated the angle at which the jet expands. The results of these calculation are give in Table 3. The values confirm our observations from the visual analysis, showing that the size of the liquid structures, as well as the amount, are highly dependent on the resolution. This was to be expected due to the numerical methods we employ, but it is noteworthy that large differences are still visible even though all three cases are within the order of magnitude of the Kolmogorov length scale, therefore, providing a high enough resolution for DNS. It has to be noted, that while all three simulations gave different quantitative results, the simulation with the fine grid and the simulation with the medium grid exhibit a similar qualitative behaviour. Even the simulation with the medium sized grid can therefore be used to understand the basic processes of jet break-up. But it becomes obvious, that in order to obtain reliable quantitative results for jets at high Reynolds numbers fine computational grids with a high amount of cells are indispensable.

5 Computational Performance

We analysed the performance of FS3D with the newly implemented UG4 multigrid solver. We compared it to the performance of FS3D with the old multigrid solver. For the performance analysis we simulate an oscillating droplet. We use the droplet instead of the jet for two reasons: First, because it is a symmetric case which distributes the load somewhat evenly. Second and more importantly, because it can be used for the performance analysis from the beginning, as opposed to a jet simulation, which can only be sensibly analysed after the jet has passed through the computational domain once. The simulation was set up with the following parameters: An elongated droplet is initialised as an ellipsoid with the semi-principal axis $a = b = 1.357\,\text{mm}$ and $c = 0.543\,\text{mm}$ at the center of a cubic computational domain with an edge length of $x = y = z = 8\,\text{mm}$. The fluid of the droplet is an aqueous solution of Praestol modelled with the Carreau-Yasuda model as described in Sect. 3. The domain is discretised with a cubic Cartesian grid.

The results for the strong scaling are shown in Fig. 6. For the strong scaling analysis we used a fixed computational grid with 512^3 cells and a second grid with

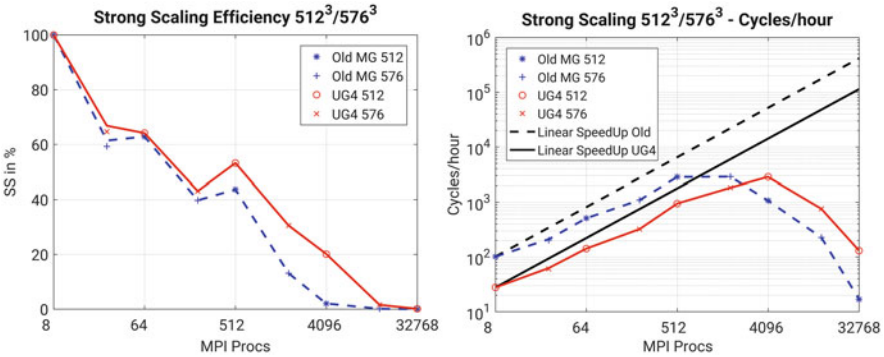

Fig. 6 Left: Strong scaling efficiency in %, for FS3D with the old multigrid solver and with UG4. Right: Strong scaling performance in cycles per hour

Table 4 Strong scaling setups

	Problem size								
	512^3					576^3			
Processors	2^3	4^3	8^3	16^3	32^3	3^3	6^3	12^3	24^3
Cells per processor	256^3	128^3	64^3	32^3	16^3	192^3	96^3	48^3	24^3
Nodes	2	4	32	256	2048	6	9	72	576
Processors per node	4	16	16	16	16	5	24	24	24

576^3 cells. We varied the number of processors from 2^3 to 32^3. The amount of processors used, as well as the resulting amount of cells which were distributed onto a processors are given in Table 4. Although FS3D has OpenMP implemented as well as MPI, and is therefore capable of hybrid operation, only the MPI parallelisation was investigated this time. For information on FS3D performance with hybrid OpenMP and MPI operation please see [26] and [9].

After running all these simulations for 1 h, we took the number of computational cycles, which were calculated during that time, and displayed the cycles per hour (cph) in Fig. 6 on the right. We then calculated the strong scaling efficiency

$$SS = \frac{t_1}{N\, t_N} \cdot 100\%, \tag{12}$$

where t_1 denotes the amount of cycles per hour, which were needed for the calculation on one processor, N is the amount of processors used in the calculation and t_N is the amount of cycles per hour, which were obtained by calculating with N processors. The strong scaling efficiency is displayed in Fig. 6 on the left.

The strong scaling efficiency of both solvers looks similar with the efficiency of UG4 slightly higher at more processors. The direct comparison of the cycles per hour shows, however, that the serial performance of the old solver was about three times better. Therefore, the performance obtained from the old solver was also much

better up to 512 processors (or 64^3 cells per processor). The better efficiency of UG4 becomes only relevant at 4096 processors—while FS3D with the old multigrid solver is then actually getting slower despite the increase in processors—for UG4 an increase in cycles per hour can still be achieved. We have to conclude, that in its current state of implementation the new multigrid solver does not improve strong scaling. The strong scaling efficiency looks better, but due to the worse serial performance it doesn't achieve a higher cycles per hour performance compared to the old solver, even using more processors. At 32,768 processors UG4 does perform about ten times faster, but at this point both solvers have slowed down even further. Numerical setups that distribute less then 32^3 cells per processors are not sensible for FS3D.

One additional comment: The grid with 576^3 cells shows a somewhat worse performance, since this case was calculated with 24 processors per node (ppn), while the grid with 512^3 cells was calculated on 16 ppn, providing more bandwidth for communication. Cases with more than 128^3 cells per processor had to use a smaller amount of processors per node (and therefore more nodes) in order to provide the necessary amount of memory.

The results for the weak scaling study are shown in Fig. 7. We fixed the amount of cells per processor to 64^3 and varied the amount of processors from 2^3 to 32^3—creating problem sizes with 128^3 to 2048^3 cells. The number of processors and the corresponding problem sizes are given in Table 5. The achieved cycles per hour are

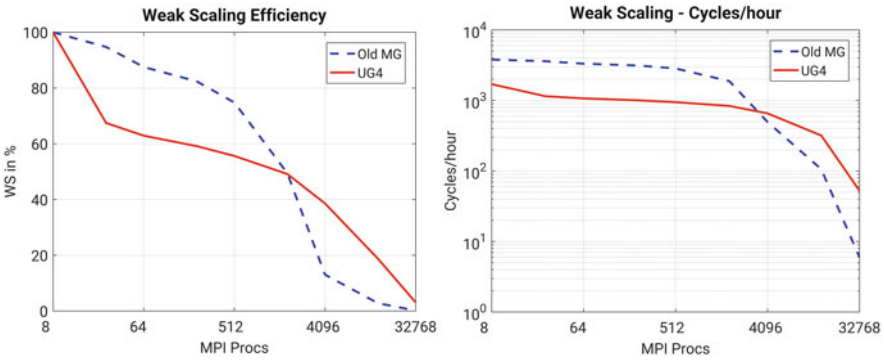

Fig. 7 Left: Weak scaling efficiency in %, for FS3D with the old multigrid solver and with UG4. Right: Weak scaling performance in cycles per hour

Table 5 Weak scaling setups

Problem size	128^3	192^3	256^3	384^3	512^3	768^3	1024^3	1536^3	2048^3
Processors	2^3	3^3	4^3	6^3	8^3	12^3	16^3	24^3	32^3
Cells per processor	64^3								
Nodes	2	3	4	9	32	72	256	576	2048
Processors per node	4	9	16	24	16	24	16	24	16

shown in Fig. 7 on the right. We also calculated the weak scaling efficiency

$$WS = \frac{t_1}{t_N} \cdot 100\%, \tag{13}$$

and displayed it in Fig. 7 on the left.

When we look at the weak scaling efficiency, we can see that for low numbers of processors both multigrid solvers scale in a similar way, but above 512 processors the efficiency of UG4 is better. When we get to a very high number of processors (32,768), the performance is bad for both multigrid solvers, with UG4 still scaling slightly better. When looking at the weak scaling performance in cycles per hour, we observe the same behaviour as seen in the strong scaling analysis—for up to 512 processors the curves run in parallel with the performance of the old multigrid solver being about three times better. From there on the performance of UG4 develops much better compared to the old solver, overtaking the old solver at 4096 processors and providing three times cph at 13,824 processors, and nine times cph at 32,768.

For jet break-up simulations we estimate, under the assumption of quadratic cells, that the necessary amount of cycles increases linearly with the domain size. The length of a cycle, or time step, for these simulations is dictated by the Courant-Friedrichs-Levy condition (CFL) [21]. Therefore, an increase in cells to obtain a finer resolution will lead to a smaller time step per cycle and thus to an increase in cycles.

We also isolated the performance of the pressure correction by timing it with calls to the `MPI_Wtime` function. The results are shown in Fig. 8. We can see that both strong scaling and weak scaling curves are in a good agreement with the above shown performance analysis for the entire code. This seems sensible since the pressure correction generally accounts for over 70 % of FS3D's computational time.

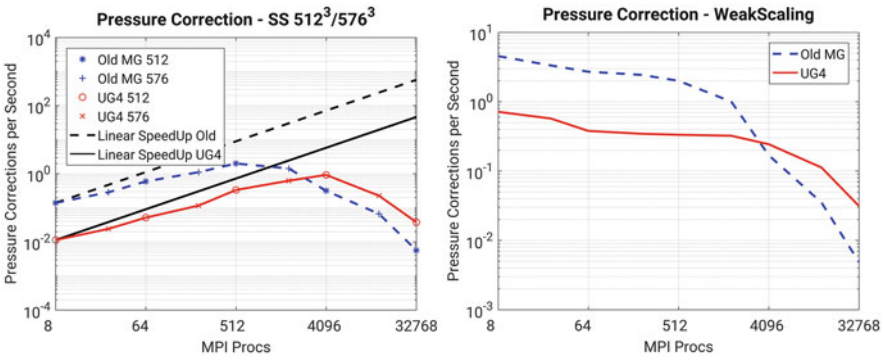

Fig. 8 Left: Strong scaling performance of the pressure correction in cycles per hour. Right: Weak scaling performance of the pressure correction in cycles per hour

6 Conclusions

We presented our multiphase flow DNS code **FS3D**, which is based on VOF and PLIC, and capable of jet break-up simulations. Moving to larger Reynolds numbers goes in hand with a demand for larger computational domains and higher resolutions. The higher resolutions become necessary due to the smaller cell size dictated by the Kolmogorov length scale and the larger domain size is due to the jet expanding more strongly in radial direction at higher Reynolds numbers. As this corresponds to an increase in size of the pressure equation sub-problem, an efficient linear solver is crucial for the overall performance.

To improve the parallel performance in such cases, we integrated the multigrid solver from UG4 into FS3D. In a preliminary performance analysis, we compared the performance of FS3D with the old multigrid solver to the performance of FS3D with replacement. However, this implementation of UG4 into FS3D was only finished shortly before submitting this report. That means no optimisation has been done on fine tuning this implementation, yet. The preliminary results are as follows:

The strong scaling analysis showed a generally better performance with the old solver. The weak scaling analysis showed the old solver to be better at low numbers of processors, while UG4 provided better performance from 4096 processors upwards. Especially around 10^4 processors we were able to obtain acceptable cph with UG4, while the old solvers performance collapsed. At 32^3 processors UG4 outperformed the old solver by a factor of 9, but the performance at this point was still deemed unacceptable for both solvers.

In these first preliminary results, we did observe an improved performance for more than 4096 processes. Since this is the region that we aim for in typical jet simulations, this is an encouraging start. However, the results show that the solver performance is still behind expectations and does not scale well. In particular, it differs from the excellent scalability (e.g., 80 % at 32^3 processors) reported for UG4 for Poisson-type problems before [27, 38, 40].

As next steps, we now seek an in-depth investigation of the observed behaviour. It must be investigated if the number of multigrid cycles for each solver call was optimal, i.e., independent of the mesh size and number of involved processes. If this is the case, the second and equally likely reason for the deficient performance is a load imbalance. As outlined in Sect. 2.3, the performance crucially depends on equally divided load among the number of involved processes at all times. At process numbers larger than 1000, it is not acceptable if just one process performs the work, while others are idling. On the other hand, it must be avoided that on coarser levels always all processes are involved, so that communication becomes the limiting factor. Both effects potentially ruin scalability, in both strong and weak sense.

In addition to these problems arising specifically in a parallel environment, the node-level performance of the UG4 solver was inferior. We have learned from the study at hand, that for low process numbers, timings were worse than for the old solver. It can be expected a tightly integrated solver provides better performance

than the solver replacement being called via an external library. Yet the reason for performance differences of a factor of 5 need to be investigated more closely. At the current stage, there are lots of opportunities for optimisations. First, one can try to change the way the matrix and the operators are set up. Right now they are reinitialised once for every time step. We are planning to change this, so that they are only initialised once and then reused and only refiled with new values during the time steps. A second aspect could be moving the boundary conditions inside the main matrix. This is done in the old multigrid solver, but not yet in the UG4 implementation. This will reduce the problem size and therefore, the computational demand as well as the communication.

Acknowledgements The authors kindly acknowledge the *High Performance Computing Center Stuttgart* (HLRS) for support and supply of computational time on the Cray XC40 platform under the Grant No. FS3D/11142 and the financial support by the Deutsche Forschungsgemeinschaft (DFG) for the Collaborative Research Center SFB-TRR75.

References

1. P. Beau, M. Funk, R. Lebas, F. Demoulin, Cavitation applying quasi-multiphase model to simulate atomization in diesel engines. SAE Technical Papers 01-0220 (2005)
2. J.U. Brackbill, D.B. Kothe, C. Zemach, A continuum method for modeling surface-tension. J. Comput. Phys. **100**(2), 335–354 (1992)
3. K. Eisenschmidt, M. Ertl, H. Gomaa, C. Kieffer-Roth, C. Meister, P. Rauschenberger, M. Reitzle, K. Schlottke, B. Weigand, Direct numerical simulations for multiphase flows: an overview of the multiphase code FS3D. J. Appl. Math. Comput. **272**(2), 508–517 (2016) https://doi.org/10.1016/j.amc.2015.05.095
4. M. Ertl, B. Weigand, Investigation of the influence of atmospheric pressure on the jet breakup of a shear thinning liquid with DNS, in *ILASS 2014*, Bremen (2014)
5. M. Ertl, B. Weigand, Analysis methods for direct numerical simulations of primary breakup of shear-thinning liquid jets. Atomization Sprays **27**(4), 303–317 (2017)
6. M. Ertl, N. Roth, G. Brenn, H. Gomaa, B. Weigand, Simulations and experiments on shape oscillations of newtonian and non-Newtonian liquid droplets, in *ILASS 2013* (2013), p. 7
7. M. Ertl, G. Karch, F. Sadlo, T. Ertl, B. Weigand, Investigation and visual analysis of direct simulations of quasi-steady primary break-up of shear thinning liquids, in *Proceedings 9th International Conference on Multiphase Flow: ICMF 2016*, Firenze (2016)
8. U. Fritsching, *Process-Spray: Functional Particles Produced in Spray Processes* (Springer, Cham, 2016)
9. C. Galbiati, M. Ertl, S. Tonini, G.E. Cossali, B. Weigand, DNS investigation of the primary breakup in a conical swirled jet, in *High Performance Computing in Science and Engineering'15 Transactions of the High Performance Computing Center, Stuttgart (HLRS)* (Springer, Cham, 2016), pp. 333–347
10. H. Gomaa, I. Stotz, M. Sievers, G. Lamanna, B. Weigand, Preliminary investigation on diesel droplet impact on oil wallfilms in diesel engines, in *ILASS – Europe 2011, 24th European Conference on Liquid Atomization and Spray Systems*, Estoril, September 2011
11. F.H. Harlow, J.E. Welch, Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. Phys. Fluids **8**(12), 2182–2189 (1965)
12. J. Hernández, J. López, P. Gómez, C. Zanzi, F. Faura, A new volume of fluid method in three dimensions—part I: multidimensional advection method with face-matched flux polyhedra. Int. J. Numer. Methods Fluids **58**(8), 897–921 (2008). https://doi.org/10.1002/fld.1776

13. M. Herrmann, A dual-scale les subgrid model for turbulent liquid/gas phase interface dynamics, in *13th Triennial International Conference on Liquid Atomization and Spray Systems ICLASS 2015*, Tainan, August 23–27 (2015)
14. C.W. Hirt, B.D. Nichols, Volume of fluid (VOF) method for the dynamics of free boundaries. J. Comput. Phys. **39**(1), 201–225 (1981). https://doi.org/10.1016/0021-9991(81)90145-5
15. M. Klein, Direct numerical simulation of a spatially developing water sheet at moderate Reynolds number. Int. J. Heat Fluid Flow **26**, 722–731 (2005)
16. B. Lafaurie, C. Nardone, R. Scardovelli, S. Zaleski, G. Zanetti, Modelling merging and fragmentation in multiphase flows with SURFER. J. Comput. Phys. **113**(1), 134–147 (1994)
17. A. Lakdawala, R. Thaokar, A. Sharma, Break-up of a non-newtonian jet injected downwards in a newtonian liquid. Sadhana Indian Acad. Sci. **40**, 819–833 (2015)
18. A.H. Lefebvre, *Atomization and Sprays* (Hemisphere, New York, 1989)
19. H. Li-Ping, Z. Meng-Zheng, D. Qing, L. Ning, X. Zhen-Yan, Large eddy simulation of atomization process of non-newtonian liquid jet. Adv. Sci. Lett. **8**, 285–290 (2012)
20. S.P. Lin, R.D. Reitz, Drop and spray formation from a liquid jet. Annu. Rev. Fluid Mech. **30**, 85–105 (1998)
21. C.D. Munz, T. Westermann, *Numerische Behandlung gewöhnlicher und partieller Differenzialgleichungen* (Springer, Berlin, 2006). ISBN 978-3-540-29867-3
22. Y. Pan, H. Suga, A numerical study on the breakup process of laminar liquid jets into a gas. Phys. Fluids **18**, 052101 (2006)
23. S. Popinet, An accurate adaptive solver for surface-tension-driven interfacial flows. J. Comput. Phys. **228**(16), 5838–5866 (2009). https://doi.org/10.1016/j.jcp.2009.04.042
24. P. Rauschenberger, B. Weigand, Direct numerical simulation of rigid bodies in multiphase flow within an Eulerian framework. J. Comput. Phys. **291**, 238–253 (2015). https://doi.org/10.1016/j.jcp.2015.03.023
25. P. Rauschenberger, J. Schlottke, K. Eisenschmidt, B. Weigand, Direct numerical simulation of multiphase flow with rigid body motion in an Eulerian framework, in *ILASS - Europe 2011, 24th European Conference on Liquid Atomization and Spray Systems*, Estoril (2011)
26. P. Rauschenberger, J. Schlottke, B. Weigand, A computation technique for rigid particle flows in an Eulerian framework using the multiphase DNS code FS3D, in *High Performance Computing in Science and Engineering'11 Transactions of the High Performance Computing Center, Stuttgart (HLRS)* (2011). https://doi.org/10.1007/978-3-642-23869-7_23
27. S. Reiter, A. Vogel, I. Heppner, M. Rupp, G. Wittum, A massively parallel geometric multigrid solver on hierarchically distributed grids. Comput. Vis. Sci. **16**(4), 151–164 (2013). https://doi.org/10.1007/s00791-014-0231-x
28. M. Reitzle, C. Kieffer-Roth, H. Garcke, B. Weigand, A volume-of-fluid method for three-dimensional hexagonal solidification processes. J. Comput. Phys. **339**, 356–369 (2017). https://doi.org/10.1016/j.jcp.2017.03.001
29. W.J. Rider, D.B. Kothe, Reconstructing volume tracking. J. Comput. Phys. **141**(2), 112–152 (1998). https://doi.org/10.1006/jcph.1998.5906
30. M. Rieber, Numerische Modellierung der Dynamik freier Grenzflächen in Zweiphasenströmungen. Dissertation, Universität Stuttgart, 2004
31. M. Rieber, F. Graf, M. Hase, N. Roth, B. Weigand, Numerical simulation of moving spherical and strongly deformed droplets, in *Proceedings ILASS-Europe* (2000), pp. 1–6
32. N. Roth, J. Schlottke, J. Urban, B. Weigand, Simulations of droplet impact on cold wall without wetting, in *ILASS* (2008), pp. 1–7
33. N. Roth, H. Gomaa, B. Weigand, Droplet collisions at high weber numbers: experiments and numerical simulations, in *Proceedings of DIPSI Workshop 2010 on Droplet Impact Phenomena & Spray Investigation*, Bergamo (2010)
34. W. Sander, B. Weigand, Direct numerical simulation of primary breakup phenomena in liquid sheets, in *High-Performance Computing in Science and Engineering 2006: Transactions of the High Performance Computing Center Stuttgart (HLRS)* (Springer, Berlin, 2006), pp. 223–236
35. J. Shinjo, A. Umemura, Surface instability and primary atomization characteristics of straight liquid jet sprays. Int. J. Multiphase Flow **37**, 1294–1304 (2011)

36. G. Strang, On the construction and comparison of difference schemes. SIAM J. Numer. Anal. **5**(3), 506–517 (1968)
37. R.I. Tanner, *Engineering Rheology*. Oxford Engineering Science Series, 2nd edn. (Oxford University Press, Oxford, 2002)
38. A. Vogel, S. Reiter, M. Rupp, A. Nägel, G. Wittum, UG4: a novel flexible software system for simulating PDE based models on high performance computers. Comput. Vis. Sci. **16**(4), 165–179 (2013). https://doi.org/10.1007/s00791-014-0232-9
39. A. Vogel, A. Calotoiu, A. Strubem, S. Reiter, A. Nägel, F. Wolf, G. Wittum, 10,000 performance models per minute – scalability of the UG4 simulation framework, in *Euro-Par 2015*, ed. by J. Träff, S. Hunold, F. Versaci, vol. 9233 (2015), pp. 519–531. https://doi.org/10.1007/978-3-662-48096-0
40. A. Vogel, A. Calotoiu, A. Nägel, S. Reiter, A. Strube, G. Wittum, F. Wolf, Automated performance modeling of the UG4 simulation framework, in *Software for Exascale Computing - SPPEXA 2013–2015*, ed. by H. Bungartz, P. Neumann, W.E. Nagel. Lecture Notes in Computational Science and Engineering, vol. 113 (Springer, Cham, 2016), pp. 467–481. https://doi.org/10.1007/978-3-319-40528-5_21
41. H. Weking, J. Schlottke, M. Boger, C.D. Munz, B. Weigand, DNS of rising bubbles using VOF and balanced force surface tension, in *High Performance Computing on Vector Systems* (Springer, Berlin, 2010)
42. C. Zhu, M. Ertl, B. Weigand, Effect of Reynolds number on the primary jet breakup of inelastic non-newtonian fluids from a duplex nozzle using direct numerical simulation (DNS), in *ILASS 2013* (2013)