POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

# Development of a lagrangian wall film approach for the modelling of spray-wall kinematic and thermal interaction

Tesi di Laurea Magistrale in
Mechanical Engineering - Ingegneria Meccanica

Author: **Angelo Chiappa**

Student ID: 247621
Advisor: Prof. Gianluca Montenegro
Academic Year: 2024-25

# Abstract

Liquid films play a decisive role in internal combustion engines and exhaust after-treatment systems such as Selective Catalytic Reduction (SCR) units. This thesis presents the development and implementation of a fully Lagrangian wall film model within the open-source CFD framework *OpenFOAM*. In contrast to the conventional lagrangian–eulerian approach available in the standard library, where droplets are tracked in a Lagrangian framework and the wall film is described by eulerian surface fields, the proposed methodology represents the wall film itself as a collection of Lagrangian parcels constrained to move along solid boundaries. Particular attention is devoted to the treatment of momentum exchange at impact, splashing regimes and heat transfer. The model has been designed to reduce computational cost and to be more robust with respect to mesh quality. The implementation adheres to the object-oriented architecture of OpenFOAM, ensuring modularity, extensibility, and seamless integration with existing spray and combustion solvers. Methodological inspiration is drawn from established Lagrangian surface film strategies employed in widely used engine simulation codes such as KIVA and CONVERGE. A verification and validation study have been conducted with respect to an established experimental test. The results demonstrate that the proposed fully Lagrangian wall film model achieves good predictive capability in capturing film formation and transport while delivering measurable gains in computational efficiency compared to the existing lagrangian–eulerian implementation.

**Keywords:** CFD, multiphase, Lagrangian wall-film, Thin-film dynamics

# Abstract in lingua italiana

I film liquidi svolgono un ruolo determinante nei motori a combustione interna e nei sistemi di post-trattamento dei gas di scarico, quali i catalizzatori SCR. Questa tesi presenta lo sviluppo e l'implementazione di un modello di film liquido a parete completamente lagrangiano all'interno del framework CFD open-source *OpenFOAM*. A differenza dell'approccio convenzionale lagrangiano–euleriano disponibile nella libreria standard, in cui le gocce sono tracciate in modo lagrangiano mentre il film a parete è descritto in modo euleriano, la metodologia proposta rappresenta il film stesso come un insieme di particelle lagrangiane vincolate a muoversi lungo le superfici solide. Particolare attenzione è stata dedicata alla modellazione dello scambio di quantità di moto all'impatto, dei regimi di splash e del trasferimento di calore. Il modello è stato progettato con l'obiettivo di ridurre il costo computazionale e di aumentare la robustezza rispetto alla qualità della mesh. L'implementazione segue l'architettura object-oriented di OpenFOAM, garantendo modularità, estendibilità e integrazione con i solver esistenti per spray e combustione. L'impostazione metodologica trae ispirazione da modelli lagrangiani per film superficiali adottati in programmi ampiamente diffusi quali KIVA e CONVERGE. L'attività di verifica e validazione è stata condotta rispetto ad un noto caso test sperimentale. I risultati dimostrano che il modello di film a parete completamente lagrangiano proposto offre una buona corrispondenza rispetto al dato sperimentale, garantendo al contempo un miglioramento in termini di costo computazionale rispetto all'implementazione Lagrangiano–Euleriana esistente.

**Parole chiave:** CFD, multifase, wall-film lagrangiano, dinamica di film sottili

# Contents

# Introduction

In internal combustion engines and exhaust after-treatment devices such as Selective Catalytic Reduction (SCR) systems, liquid films forming on solid surfaces strongly influence evaporation, chemical conversion, pollutant formation, and overall system efficiency. The ability to model the dynamics of these wall films with both fidelity and computational efficiency is therefore essential for the design and optimization of modern propulsion and emissions-control technologies.

*OpenFOAM* provides a flexible, Open-Source framework for multiphase flow simulation and Lagrangian spray modelling. However, the existing wall film treatment in *OpenFOAM* is based on the lagrangian–eulerian approach, in which droplets are tracked in a Lagrangian manner while the film is represented on the wall surface using eulerian fields. Although this strategy has proven effective in many contexts, it introduces non-negligible computational overhead and may be strongly affected by mesh quality.

This thesis presents the development of a fully Lagrangian wall film model implemented within the *OpenFOAM* framework. The proposed model represents the wall film as a collection of Lagrangian parcels constrained to move along solid boundaries.

The model has been designed with the objectives of reducing computational cost and simplifying the setup with respect to the existing lagrangian–eulerian wall film implementation available in *OpenFOAM*. This was achieved by introducing additional force, friction, and heat transfer sub-models into the lagrangian library, in order to account for the specific physical mechanisms governing the film dynamics. The implementation follows the object-oriented architecture of *OpenFOAM* to ensure modularity and extensibility, allowing straightforward integration with existing spray and combustion solvers.

The development takes inspiration from established methodologies employed in widely used engine simulation codes such as KIVA and CONVERGE, which have demonstrated the effectiveness of Lagrangian-based surface film modeling. Through a verification and validation study, this work evaluates the performance of the new model in representative configurations and quantifies its advantages in terms of computational efficiency.

Overall, this work introduces a novel lagrangian wall film modelling framework within *OpenFOAM*, enabling simpler and computationally more efficient simulations of liquid–wall interactions in engineering applications where wall films play a decisive role.

# 1 | Theoretical Background

## 1.1. Eulerian and Lagrangian Approaches

When observing a fluid flow, every fluid element follows its own trajectory. If some fluid elements are grouped, it is possible to imagine the construction of a fluid body that occupies a given region of the space at instant $t_1$. At the time instant $t_2$, the shape and the volume of the fluid body will be changed, as it can be seen in Figure 1.1

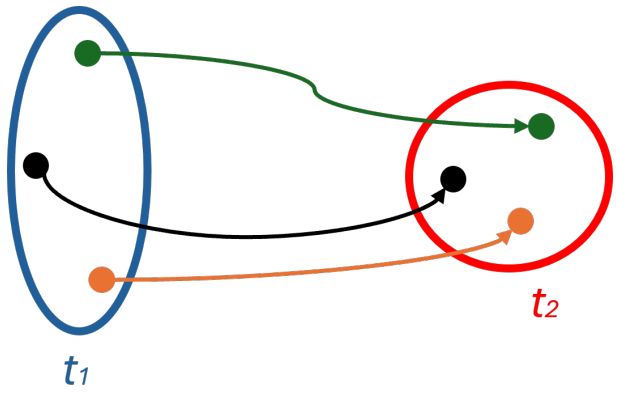



Figure 1.1: Fluid body variation between $t_1$ and $t_2$.

With the Lagrangian approach, any position in space where a fluid element arise can be generally described as function of the initial position and the time. To study the single fluid element, the position at $t_0$ and the time $t_0$ must be known. In other words, knowing $\vec{x}_0$ and $t_0$, it is possible to find:

$$\vec{X}_1 = \vec{x}(\vec{x}_0, t_1) \tag{1.1}$$

This can be generalized to all the Lagrangian quantities:

$$\phi^L = \phi^L(\vec{x}_0, t) \tag{1.2}$$

This approach is normally not used in fluid dynamics. When studying a flow, generally the interest is about a fixed region in space, hence a control volume $\Omega$ in which the fluid

enters and exits. As an example we can consider the fluid coming from a reservoir and exchanging work in a turbomachinery reported in Figure 1.2. In this case, there is no need to study the behaviour of the single fluid particle, on the other hand it is important to characterize the spatial domain where the turbomachinery is placed.

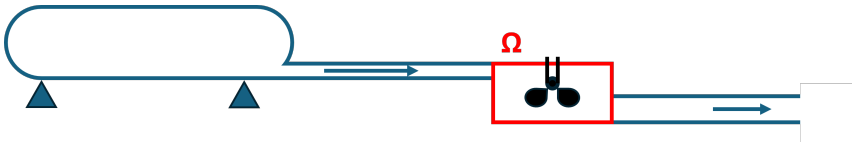



Figure 1.2: Generic turbomachinery and control volume Ω.

In other words it is required to pass from the Lagrangian approach expressed in Equation 1.2 to the Eulerian approach, where the generic function $\phi$ that depends only on $\vec{x}$ and t. As it can be seen in Equation 1.3, the dependency of $\phi^E$ from the initial position becomes implicit.

$$\phi^E = \phi^E(\vec{x}, t) = \phi^E(\vec{x}(\vec{x}_0, t), t) \tag{1.3}$$

This change of notation changes also the way a quantity is derived with respect to time:

$$\frac{d\phi^L}{dt} = \frac{\partial \phi^L}{\partial t} \tag{1.4}$$

$$\frac{d\phi^E}{dt} = \frac{\partial \phi^L}{\partial t} + \frac{\partial \phi^L}{\partial \vec{x}} \frac{\partial \vec{x}}{\partial t} = \frac{\partial \phi^L}{\partial t} + \vec{\nabla} \phi^E \vec{U} = \frac{D\phi^E}{Dt} \tag{1.5}$$

The derivative operator in the Eulerian approach is called substantial derivative and is written with capital D, as it can be seen in Equation 1.5. This approach is valid instantaneously: there is no need to know the history of the flow.
When considering a fluid body which is still in a reservoir, every fluid element has its own mass and volume $(dM, dV)$.

$$\frac{dM}{dV} = \rho \rightarrow M = \int_V \rho dV \tag{1.6}$$

If the fluid body starts moving, as shown already in Figure 1.1, the volume changes $(V = V(t))$, while mass doesn't change $(\frac{dM}{dt} = 0)$, this leads to:

$$\frac{d}{dt} \int_{V(t)} \rho dt = 0 \tag{1.7}$$

At this point it is required to pass from the domain of the fluid body $(V = V(t))$ to the control volume Ω. This is done with the *Reynolds Transport theorem* that transforms

an instantaneous balance equation into the rate of change with time of an extensive quantity. Taking as an example the generic function $\phi$ and evaluating it into a fluid body it is possible to write:

$$\frac{d\phi}{dt} = \frac{d}{dt}\int_{V(t)} \phi d\Omega \tag{1.8}$$

The derivative with respect to time can be done resorting to kinematic analysis only once the rate of change with time of the material volume $\Omega(t)$ is known:

$$\frac{d\phi}{dt} = \int_{V(t)} \frac{d(\phi d\Omega)}{dt} = \int_{V(t)} \frac{d\phi}{dt} d\Omega + \int_{V(t)} \phi \frac{d(d\Omega)}{dt} \tag{1.9}$$

Because of the continuos deformation of the fluid body, $d\Omega$ changes with time and can be proven that:

$$\frac{d(d\Omega)}{dt} = \vec{\nabla} \cdot \vec{U} d\Omega = (\frac{\partial U_x}{\partial x} + \frac{\partial U_y}{\partial y} + \frac{\partial U_z}{\partial z}) d\Omega \tag{1.10}$$

By substituting Equation 1.10 into Equation 1.9 and remembering that in the Eulerian approach, to derive in time the substantial derivative is required, as seen in Equation 1.5 the following equation holds:

$$\frac{d\phi}{dt} = \int_{V(t)} \frac{\partial \phi}{\partial t} d\Omega + \int_{V(t)} \vec{\nabla} \cdot (\phi \vec{U}) d\Omega \tag{1.11}$$

Since the right hand side of Equation 1.11 does not require to evaluate any integral rate of change but only instantaneous integrals, it is possible to choose a fluid volume $V = V(t)$ instantaneously equal to the control volume $\Omega$:

$$\frac{d\phi}{dt} = \int_{\Omega} \frac{\partial \phi}{\partial t} d\Omega + \int_{\Omega} \vec{\nabla} \cdot (\phi \vec{U}) d\Omega = \frac{d}{dt} \int_{\Omega} \phi d\Omega + \oint_{S} \phi \vec{U} \cdot d\vec{S} \tag{1.12}$$

In this way, the rate of change of an extensive fluid quantity is rewritten as a combination of a local unsteady term and a flux convective term.

## 1.2. Fluid Mechanics equations

A conservation equation states that the variation of the total amount of a quantity inside a given domain is equal to the balance between the amount of that quantity entering and leaving the considered domain, plus the contributions from eventual sources generating that quantity. Three conservation equations are required to completely determine the behaviour of a fluid:

- Conservation of mass, *(continuity equation)*
- Conservation of momentum, *(momentum equation)*
- Conservation of energy, *(energy equation)*

Additional information are used to determine the state of the fluid *(equation of state)*. When applied to a viscous flow, this set of equations is known as the *Navier-Stokes equations*, see Equation 1.13 [25].

$$\begin{cases} \dfrac{\partial \rho}{\partial t} + \vec{\nabla}\cdot(\rho\vec{U}) = 0, \\ \dfrac{\partial(\rho\vec{U})}{\partial t} + \vec{\nabla}\cdot\Big(\rho\,\vec{U}\otimes\vec{U} \;-\; p\,\overline{\overline{I}} \;+\; \overline{\overline{\boldsymbol{\tau}}}\Big) = \rho\,\vec{f_e}, \\ \dfrac{\partial(\rho E)}{\partial t} + \vec{\nabla}\cdot\Big(\rho\,H\,\vec{U} \;-\; k\,\vec{\nabla}T \;-\; \overline{\overline{\boldsymbol{\tau}}}\,\vec{U}\Big) = \rho\,\vec{f_e}\cdot\vec{U} \;+\; q_H. \end{cases} \tag{1.13}$$

Where $\tau_{i,j}$ represents the mechanical response of the fluid to deformation:

$$\overline{\overline{\boldsymbol{\tau}}} = \mu\Big(\vec{\nabla}\vec{U} + (\vec{\nabla}\vec{U})^T\Big) + \lambda\,(\vec{\nabla}\cdot\vec{U})\,\overline{\overline{I}}. \tag{1.14}$$

By applying the Reynolds Transport Theorem, introduced in section 1.1, these equations can be expressed in integral form form over a control volume $V$ bounded by a surface $S$ with outward normal vector $\vec{n}$:

$$\begin{cases} \dfrac{d}{dt}\displaystyle\int_V \rho\,dV + \int_S \rho\,\vec{U}\cdot\vec{n}\,dS = 0, \\ \dfrac{d}{dt}\displaystyle\int_V \rho\vec{U}\,dV + \int_S \Big(\rho\,\vec{U}\otimes\vec{U}\Big)\cdot\vec{n}\,dS = \int_V \rho\,\vec{f_e}\,dV + \int_S \left(\overline{\overline{\sigma}}\cdot\vec{n}\,dS\right), \\ \dfrac{d}{dt}\displaystyle\int_V \rho E\,dV + \int_S \Big(\rho H\vec{U} - k\,\vec{\nabla}T - \overline{\overline{\boldsymbol{\tau}}}\cdot\vec{U}\Big)\cdot\vec{n}\,dS = \int_V \rho\,\vec{f_e}\cdot\vec{U}\,dV + \int_V q_H\,dV. \end{cases} \tag{1.15}$$

## 1.3. Introduction to CFD

Computational Fluid Dynamics (CFD) is the analysis of systems involving fluid flow and heat transfer by means of computer simulations. It is based on the Navier-Stokes equations just presented. The steps for a generic CFD simulation are:

1. Definition of the computational domain ($\Omega$) and its discretization into a finite number of control volumes, forming the computational mesh.
2. Selection of the physical models (e.g. turbulence, heat transfer)
3. Specification of the boundary and initial conditions
4. Selection of the discretization schemes for the governing equations
5. Execution of the numerical simulation
6. Post-processing and visualization of the results

In the following a brief explanation of all these steps will be given.

### 1.3.1. Computational Domain

The definition and discretization of the computational domain are necessary because computers cannot directly solve equations defined over continuous variables. Instead, the domain is subdivided into a finite number of small control volumes, over which the governing equations are approximated in discrete form, allowing integrals to be evaluated as algebraic summations. In finite volume discretization, the standard in CFD, space is discretized by creating a grid of points, and combining them to form volumes. For each control volume (cell) the flow quantities are stored at the cell centroid. The centroid is defined as the point whose position $x_p$ satisfies:

$$\int_V (x - x_p)\, dV = \mathbf{0} \tag{1.16}$$

where $V$ is the cell volume and $x$ is the position vector. This definition ensures that the centroid represents the geometric centre of the control volume.

### 1.3.2. Physical models

Navier-Stokes equation are non-linear: the main non-linearity is in the convection term, $\vec{U} \otimes \vec{U}$ in Equation 1.13. The role of non-linearity is quantified by Reynolds number, $Re = \frac{Ud}{\nu}$, which expresses the ratio between inertia and viscous forces and whose value

the drastically influences the character of the flow regime. If $Re$ is below a given threshold (which is problem dependent), the flow is laminar, else the flow is turbulent. Most of engineering relevant flows are turbulent: turbulence is a three-dimensional, unsteady, rotational fluid motion with broad-banded fluctuations of flow quantities (velocity, pressure, temperature) occurring in both time and space. Turbulence greatly enhances the rates of momentum, heat, and mass transfer. Moreover, wall heat flux, friction and drag forces, largely depend on turbulence level.

Solving the Navier–Stokes equations in their original form, as reported in Equation 1.13, for turbulent flows requires an extremely fine spatial and temporal discretization in order to resolve all turbulent scales. This approach, known as Direct Numerical Simulation (DNS), demands prohibitive computational resources and is therefore not suitable for industrial applications. Moreover, resolving the smallest turbulent fluctuations is often unnecessary, since engineering analyses are primarily concerned with the macroscopic effects on the mean flow. What is usually done, instead, is to solve the so called Reynolds Averaged Navier Stokes equations (RANS). This set of equations is derived by taking the mean of Equation 1.13, after rewriting each flow quantity as the summation of a mean ($\overline{\phi}$) and a fluctuating ($\phi'$) component:

$$\overline{\phi}(x) = \frac{1}{\Delta T}\int_{t-\Delta T/2}^{t+\Delta T/2} \phi(x,t)\,dt \qquad ; \qquad \phi = \overline{\phi}(x) + \phi' \tag{1.17}$$

In this averaging procedure, the following properties hold:

$$\begin{cases} \overline{\varphi + \phi} = \overline{\varphi} + \overline{\phi} \\ \overline{\lambda\varphi} = \lambda\overline{\varphi}, \qquad \lambda \text{ scalar} \end{cases} \qquad \begin{cases} \overline{\dfrac{\partial\varphi}{\partial\alpha}} = \dfrac{\partial\overline{\varphi}}{\partial\alpha}, \qquad \alpha = t, x \\ \overline{\displaystyle\int \varphi\, d\alpha} = \displaystyle\int \overline{\varphi}\, d\alpha \end{cases} \tag{1.18}$$

From the properties of the Reynolds averaging operator, the fluctuating component has zero mean value and the average of the product between a mean quantity and its fluctuation is zero:

$$\overline{\phi'} = 0 \qquad \overline{\overline{\phi}\,\phi'} = \overline{\phi}\,\overline{\phi'} = 0$$

However, the average of the product of fluctuating quantities is, in general, non–zero,

$$\overline{\phi'\phi'} \neq 0 \qquad \overline{\phi'\psi'} \neq 0$$

Applying the Reynolds averaging procedure to the mass conservation equation gives:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \left(\rho \vec{\overline{U}}\right) + \nabla \cdot \left(\rho \vec{U'}\right) = 0 \tag{1.19}$$

Taking the Reynolds average of the equation, the fluctuating contribution vanishes:

$$\overline{\frac{\partial \rho}{\partial t} + \nabla \cdot \left(\rho \vec{\overline{U}}\right) + \nabla \cdot \left(\rho \vec{U'}\right)} = 0 \quad \Rightarrow \quad \frac{\partial \rho}{\partial t} + \nabla \cdot \left(\rho \vec{\overline{U}}\right) = 0 \tag{1.20}$$

The averaged momentum equation is more complex due to the non-linear convective term. Applying the Reynolds decomposition, yields:

$$\frac{\partial (\rho \vec{\overline{U}})}{\partial t} + \overline{\nabla \cdot \left[\rho \left(\vec{\overline{U}} + \vec{U'}\right) \otimes \left(\vec{\overline{U}} + \vec{U'}\right)\right]} = \rho \vec{g} - \nabla \overline{P} + \mu \nabla^2 \vec{\overline{U}} + \frac{1}{3}\mu \nabla (\nabla \cdot \vec{\overline{U}}) \tag{1.21}$$

Expanding the non-linear convective term and applying the Reynolds averaging operator gives:

$$\overline{\nabla \cdot \left[\rho \left(\vec{\overline{U}} + \vec{U'}\right) \otimes \left(\vec{\overline{U}} + \vec{U'}\right)\right]} = \nabla \cdot \left(\rho \vec{\overline{U}} \otimes \vec{\overline{U}}\right) + \nabla \cdot \left(\rho \overline{\vec{U'} \otimes \vec{U'}}\right) \tag{1.22}$$

The linear terms are replaced by their corresponding means, the non-linear term becomes a non linear mean term and a non linear combination of fluctuating components. The fluctuating component behaves like a stress which increases the viscous one and it is known as Reynolds stress tensor:

$$\overline{\overline{r}} = \rho \overline{\vec{U'} \otimes \vec{U'}} = -\rho \begin{bmatrix} \overline{u'^2} & \overline{u'v'} & \overline{u'w'} \\ \overline{v'u'} & \overline{v'^2} & \overline{v'w'} \\ \overline{w'u'} & \overline{w'v'} & \overline{w'^2} \end{bmatrix} \tag{1.23}$$

The trace of this tensor can be expressed in the following way:

$$tr(\overline{\overline{r}}) = -\rho(\overline{u'^2} + \overline{v'^2} + \overline{w'^2}) = -2\rho k \tag{1.24}$$

Where $k$ is the turbulent kinetic energy: $k = 1/2(\overline{u'^2} + \overline{v'^2} + \overline{w'^2})$. In this form, the equations are not closed, hence modelling is needed to achieve a solution. The Reynolds stress tensor can be decomposed into an isotropic and a deviatoric component:

$$\overline{\overline{r}} = -2/3\rho k \overline{\overline{I}} + \overline{\overline{a}} \quad \Rightarrow \quad \overline{\overline{a}} = \overline{\overline{r}} + 2/3\rho k \overline{\overline{I}} \tag{1.25}$$

The isotropic Reynolds stress is added to the pressure term since it acts evenly in all directions. To achieve an effective modelling of the deviatoric (anisotropic) component,

Boussinesq proposed an analogy with the Newton's stress-strain-rate law:

$$\overline{\overline{a}} = 2\mu_t \overline{\overline{\overline{D}}} \tag{1.26}$$

So, with the Boussinesq hypothesis, this component is reduced to be proportional to the mean strain rate $\overline{\overline{\overline{D}}}$ through a turbulent viscosity $\mu_t$. The most famous model of closure to this problem is the $k-\varepsilon$ model by Jones and Launder. This model adds two equations to the RANS set for the computation of the turbulent viscosity. One equation for the transport of the turbulent kinetic energy $k$, and one for its dissipation rate $\varepsilon$. Given the presence of the term $\varepsilon/k$ in the $\varepsilon$ equation, this model is singular at the wall due to the adherence condition that imposes 0 velocity for the fluid there. For this reason, when using this model wall functions are required. To overcome this problem, Wilcox proposed the $k-\omega$ model, which substitutes the $\varepsilon$ equation with a transport equation for $\omega = \varepsilon/k$. This allows the simulation of the flow near the wall.

### 1.3.3. Specification of the boundary and initial conditions

As discussed in subsection 1.3.1, the solution of a CFD problem requires the definition of a finite computational domain, which is discretized into a finite number of control volumes. In order for the mathematical problem to be well-posed, appropriate boundary conditions must be specified on all domain boundaries. At solid walls (material boundaries), the no-slip condition is typically imposed, enforcing zero fluid velocity relative to the wall. Additionally, a zero normal gradient condition is commonly applied to the pressure field. Artificial or immaterial boundaries, such as inlets, outlets, and far-field boundaries, require particular care, since inappropriate boundary conditions may lead to unphysical results or numerical instability. For incompressible flows, a common approach is to prescribe the velocity at the inlet and the static pressure at the outlet, ensuring a properly constrained solution. Initial conditions must also be specified to initialize the flow field; however, for steady-state simulations, their influence is generally limited, as the solution evolves iteratively toward a converged state. To solve the additional transport equations introduced by turbulence closure models, appropriate boundary conditions must also be specified for the turbulent quantities. A well-posed problem is obtained by prescribing the turbulent kinetic energy $k$ and either the dissipation rate $\varepsilon$ or the specific dissipation rate $\omega$ at the domain inlet, depending on the selected turbulence model. Since these quantities are generally not known a priori, they are commonly estimated from empirical relations based on the turbulence intensity and a characteristic length scale of the flow.

### 1.3.4. Selection of the discretization schemes for the governing equations

Recalling the definition of the cell centroid given in Equation 1.16, and assuming a linear variation of the transported quantity within each control volume, the value of a generic scalar field $\phi$ can be expressed through a Taylor series expansion about the cell centroid $x_p$:

$$\phi = \phi_p + (x - x_p) \cdot (\nabla\phi)_p + \mathcal{O}\left(|x - x_p|^2\right) \tag{1.27}$$

This shows that the approximation is second–order accurate in space, provided the gradients are evaluated with second–order accuracy. Analogously, a face centroid $x_f$ can be defined as:

$$\int_S (x - x_f)\, dS = 0 \tag{1.28}$$

and the same linear approximation can be written for the face values:

$$\phi = \phi_f + (x - x_f) \cdot (\nabla\phi)_f + \mathcal{O}\left(|x - x_f|^2\right) \tag{1.29}$$

The integral conservation equations presented in Equation 1.15 and averaged through the Reynolds procedure are applied to each individual control volume as well as to the entire computational domain. When summed over all control volumes, the fluxes across internal faces cancel in pairs, leaving only the boundary contributions and thus recovering the global conservation equations for the whole domain. To obtain an algebraic equation for a cell, the surface and volume integrals over the control volume must be approximated.

#### Volume integral

Replacing the volume integral with the value of the integrand at the cell centroid multiplied by the volume of the cell it is possible to achieve $2^{\text{nd}}$ order accuracy:

$$\int_V \phi dV = \phi_p V_p + \mathcal{O}\left(\Delta x^2\right) \tag{1.30}$$

#### Surface integral

The surface integrals are approximated using the midpoint rule, in which the integrand is evaluated at the face centroid:

$$\int_S \phi\, \vec{n}\, dS \;\approx\; \phi_f\, \vec{S} \tag{1.31}$$

It must be noted that $\phi_f$ is not directly known, since all variables are stored at the cell centroids. Therefore, the face value must be obtained by interpolation from neighbouring cell values. Different discretization schemes can be used for this purpose.

The simplest approach is the *upwind scheme*, in which the face value is taken equal to the value in the upstream cell, identified according to the sign of the mass flux through the face:

$$\phi_f = \begin{cases} \phi_P & \text{if } \vec{U}_f \cdot \vec{S} > 0 \\ \phi_N & \text{if } \vec{U}_f \cdot \vec{S} < 0 \end{cases} \tag{1.32}$$

where $P$ denotes the upstream (owner) cell and $N$ the downstream (neighbour) cell. This scheme is first–order accurate and numerically stable, but introduces artificial diffusion.

A higher–accuracy alternative is the *linear scheme* (central differencing), which assumes a linear variation of the variable between adjacent cell centroids. The face value is obtained by linear interpolation:

$$\phi_f = \lambda\, \phi_P + (1 - \lambda)\, \phi_N \tag{1.33}$$

where the interpolation factor $\lambda$ depends on the relative position of the face centroid between the two cell centroids. For uniform meshes, this reduces to:

$$\phi_f = \frac{\phi_P + \phi_N}{2} \tag{1.34}$$

This scheme is second–order accurate in space, but is unconditionally unstable. Several higher–resolution schemes are also available, such as Total Variation Diminishing (TVD) and linear upwind schemes, which achieve an order of accuracy higher than first order while preserving numerical stability and reducing non-physical oscillations; however, their detailed formulation is beyond the scope of the present discussion.

The gradient operator, which appears in diffusion terms, is typically approximated using Gauss's theorem: the volume integral of the gradient over a cell is converted into a surface integral over the cell faces:

$$\int_V \vec{\nabla}\phi dV = \int_S \phi \vec{n} dS \approx \sum_f \phi_f \vec{S}_f \tag{1.35}$$

where $\phi_f$ is evaluated at the face centroid using an appropriate interpolation scheme. This approach ensures that the diffusive flux is computed consistently across cell faces, and the accuracy of the gradient approximation depends on the interpolation method used for $\phi_f$.

The convective term, expressed as the divergence of a flux, is also converted to a surface

integral via Gauss's theorem:

$$\int_V \vec{\nabla} \cdot (\phi \vec{U}) dV = \int_S \phi \vec{U} \cdot \vec{n} dS \approx \sum_f \phi_f (\vec{U}_f \cdot \vec{S}_f) \tag{1.36}$$

Here, the face value $\phi_f$ must be reconstructed from the neighbouring cell values. Upwind schemes provide stability for convection-dominated flows, whereas higher-order schemes like linear or TVD improve accuracy but require careful treatment to avoid oscillations. In summary, the present analysis highlights the importance of the chosen discretization schemes. Proper selection of these methods is essential for accurately capturing transport phenomena while maintaining numerical stability.

### 1.3.5. Execution of the numerical simulation

#### Time advancment

Once the governing equations are discretized in space, their solution requires a suitable treatment of the temporal and algebraic terms. Two main approaches can be adopted for the time advancing: explicit and implicit schemes. In an explicit scheme, the value of a variable at the new time level is computed directly from quantities known at the previous time level:

$$\phi_p^{t+1} = \frac{Q - \sum_N a_n \phi_n^t}{a_p} \tag{1.37}$$

where $Q$ represents the source term associated to the cell $p$, $a_n$ are the coefficients associated with neighbouring cells, and $a_p$ is the coefficient for cell $p$. Since all neighbour values are known, no system of equations must be solved, making explicit schemes computationally inexpensive per time step. However, their stability is conditionally constrained by the time step size. The CFL number, defined in Equation 1.38 represents the ratio between the physical distance travelled by a fluid particle during a time step and the characteristic cell size. Explicit schemes require sufficiently small CFL numbers (below 1) to ensure numerical stability, whereas implicit schemes allow larger values.

$$\mathrm{Co} = \frac{|\vec{U}| \, \Delta t}{\Delta x} \tag{1.38}$$

In an implicit scheme, the unknown value at the new time level appears within the discretized equations:

$$\phi_p^{t+1} = \frac{Q - \sum_N a_n \phi_n^{\mathrm{t+1}}}{a_p} \tag{1.39}$$

This results in a system of algebraic equations that must be solved iteratively. Although

computationally more expensive per time step, implicit schemes allow the use of larger time steps. For steady-state simulations, the governing equations are typically solved in a fully implicit form, treating time as a pseudo-time to reach convergence.

### Solution algorithms

For incompressible flows, pressure and velocity are coupled through the continuity and momentum equations. The *SIMPLE* (Semi-Implicit Method for pressure-linked Equations) algorithm was conceived to derive an alternative equation for the pressure, derived from the momentum and continuity equations. Its procedure can be summarized as follows:

1. An initial pressure field is guessed.
2. The momentum equations are solved using the current pressure field to obtain an intermediate velocity field.
3. A pressure correction equation is derived from the continuity equation.
4. The pressure field is corrected using the solution of the pressure correction equation.

The SIMPLE algorithm is primarily designed for steady-state problems. Variants of this method have been developed to extend applicability. The *PISO* (Pressure Implicit with Splitting of Operators) algorithm introduces additional correction steps within each time step. The *PIMPLE* algorithm combines features of both SIMPLE and PISO.

## 1.3.6. Post-processing of numerical results

Post–processing of the numerical data is performed to analyse and interpret the physical behaviour of the simulated flow. This phase relies on dedicated visualization software, which allow the flow field to be examined through contours, vector plots, streamlines, and other graphical representations In addition to visualization, quantitative analysis is carried out by probing and plotting specific variables at selected locations or along defined lines and surfaces. This enables a detailed evaluation of local flow properties. Furthermore, integral quantities—such as forces, moments or mass flow rates are extracted from the simulation. These global metrics are essential for comparing different configurations and for validating the numerical results against experimental measurements or reference data.

## 1.4. CFD modelling of two phase flows

Two-phase flows occur in a wide range of engineering applications, particularly in propulsion and power systems. These includes: cavitating flows, flashing flows in valves or expanders, and fuel spray atomization processes in internal combustion engines.

It is possible to divide these fluids into two categories:

- two immiscible fluids divided by an interface (sea waves).
- the flow of a single liquid in phase transition (cavitating, flashing).

In the following, the two most relevant approaches to solve a flow of immiscible fluids separated by a sharp interface will be explained.

### 1.4.1. Volume of fluid (VoF) method

The Volume of fluid method is an Eulerian based approach: in this method, there is only one mass and momentum equation for the system, similar to the ones reported in Equation 1.15, and the pressure-velocity coupling is the same as for the incompressible, single phase flow. The interface between the two fluids is prescribed as an initial condition and it is evolved during the calculations: this is done by tracking its dynamics as the convection of the volume fraction of the phase inside the cell (passive scalar $\alpha$):

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot (\alpha \vec{U}) = 0 \tag{1.40}$$

The volume fraction is also used to compute the density and the viscosity inside each cell:

$$\rho_{mix} = \alpha \rho_1 + (1 - \alpha)\rho_2 \tag{1.41}$$

$$\mu_{mix} = \alpha \mu_1 + (1 - \alpha)\mu_2 \tag{1.42}$$

In Figure 1.3 on the left, the initial field where there is a net separation between the two fluids 1 and 2 is reported. As velocity field establishes, the interface (yellow dotted line) is moved, as reported on the right side of Figure 1.3. Across the interface, neither the velocity nor the pressure change, but only the volume fraction $\alpha$.

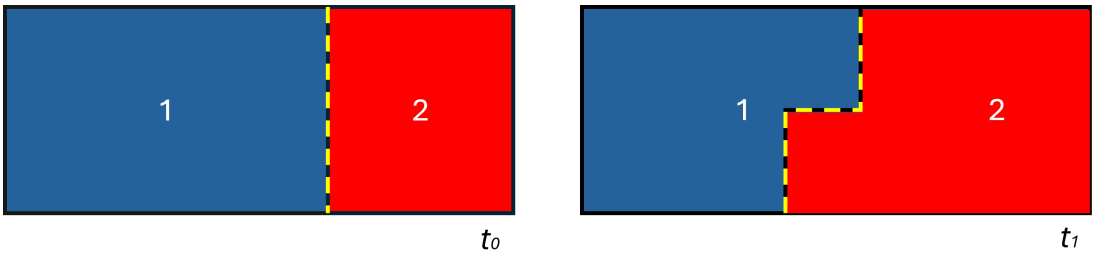



Figure 1.3: Interface at initial conditions and after a $\Delta t$.

The interface does not follow necessarily the discretized domain, but can cut the cells. To reproduce the interface properly, an "intra-cell" volume fraction F is defined, so that:

$$\alpha_{cell} = \frac{\int_{Vcell} F(x)dV}{Vcell} \quad \begin{cases} F(x) = 1 & \text{if } \alpha_x = 1 \\ F(x) = 0 & \text{if } \alpha_x = 0 \end{cases} \tag{1.43}$$

In other words, if $0 < \alpha_{cell} < 1$, the cell contains the interface. Across the interface, the surface tension exert its effects and is considered with an additional term ($F_s$) in the momentum equation.

$$F_s = \sigma\kappa(x) \cdot \vec{n} \tag{1.44}$$

Where $\vec{n}$ is the unit vector normal to the interface ($n = \frac{\nabla\alpha}{|\nabla\alpha|}$) and $\kappa(x)$ is the curvature of the interface ($\kappa(x) = \nabla \cdot \vec{n}$).

## Interface tracking schemes

To track the interface there are two main methods:

- Donor-acceptor scheme;
- Piecewise linear interface calculation (PLIC).

They both start by identifying the interfacial cells ($0 < \alpha_{cell} < 1$). The donor-acceptor scheme divides the interfacial cells in donors (the cells that "donates" fluid 1) and acceptors. The amount of fluid 1 convected across a cell boundary is limited either by the filled volume in the donor cell or the free volume in the acceptor cell. The interface orientation can be either horizontal or vertical, depending on the direction of the volume of fluid. This approach provides good representation of $\alpha$, but the definition of the interface is bad. On the other hand, the PLIC algorithm assumes that the interface between two fluids has a linear slope within each cell. The position and the inclination of the interface are computed based on the VoF function and its gradient. The calculation of the convected

fluid through each face are done using the interface representation velocity components, while the calculation of the cell volume fraction is done with the balance of fluxes.

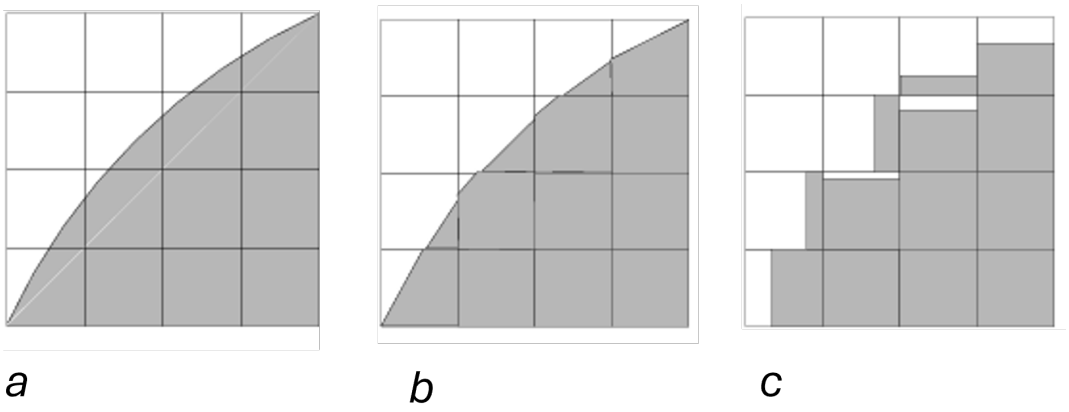



Figure 1.4: Actual interface shape (a), shape represented by PLIC (b) and shape represented by donor-acceptor scheme (c). Picture taken from [2].

### 1.4.2. Lagrangian method

The Lagrangian approach is used when there is a liquid spray carried by a gas flow. The spray, is modelled with a finite number of discrete parcels. Every parcel contains a statistical number $N_p$ of droplets with the same properties. The parcels are tracked through the gas in a Lagrangian fashion according to mass, momentum and energy exchange with the carrier phase. It must be noticed that, in the gas phase is still solved using the Eulerian approach: the interaction with the spray is accounted for by introducing appropriate source terms in the Equation 1.13. In this approach, the Lagrangian droplets do not occupy the cells, hence as a rule of thumb, the $\alpha_{cell} < 0.2$ to maintain a good validity of the equations. This aspect, implies that the mesh for a should be refined only up to a certain extent, because a mesh which is too fine results in inaccurate results. In the following, the equations governing this approach will be given, starting from gas phase:

$$\begin{cases} \dfrac{\partial \rho}{\partial t} + \vec{\nabla}\cdot(\rho\vec{U}) = \dot{\rho_s}, \\ \dfrac{\partial(\rho\vec{U})}{\partial t} + \vec{\nabla}\cdot\left(\rho\,\vec{U}\otimes\vec{U} \;-\; p\,\overline{\overline{I}} \;+\; \overline{\overline{\tau}}\right) = \rho\,\vec{g} + F_s, \\ \dfrac{\partial(\rho h_s)}{\partial t} + \vec{\nabla}\cdot\left(\rho\,\vec{U}\,h_s\right) - \vec{\nabla}\cdot\left(D_{eff,h}\,\nabla h_s\right) = \dfrac{Dp}{Dt} + \dot{Q}^c + \dot{Q}^s. \end{cases} \tag{1.45}$$

As it can be seen, from Equation 1.45, in the continuity equation, the $\dot{\rho_s}$ source term appears, which accounts for the evaporation of the spray. In the momentum equation, $F_s$

accounts for the rate of momentum gain/loss per unit volume due to the spray (drag). For what concerns the energy equation, here it is expressed with the sensible enthalpy $h_s$: $Q^c$ accounts for the heat coming from chemical reactions and $Q^s$ is the heat coming from the the spray (evaporation and heat transfer). It is worth mentioning that also a chemical specie equilibrium equation is needed, where $Y_i$ is the specie mass fraction:

$$\frac{\partial \rho Y_i}{\partial t} + \nabla \cdot (\rho \vec{U} Y_i) - \nabla (D_{eff} \cdot \nabla Y_i) = \dot{\omega}_i + \dot{\rho}_i \tag{1.46}$$

Similarly to what already seen in the continuity equation, $\dot{\rho}_i$ is an evaporation source term for the single liquid component, while $\dot{\omega}_i$ accounts for chemical reactions.At this point the focus can be moved towards the dispersed phase, hence the droplets, described with the Lagrangian approach. The only term that causes variations to the mass balance of a liquid droplet is evaporation:

$$\frac{dm_d}{dt} = \dot{m}_d \tag{1.47}$$

Where $\dot{m}_d$ is the evaporation rate. The momentum equation can be expressed like:

$$m_d \frac{d\vec{U}_d}{dt} = \vec{F} \tag{1.48}$$

where $m_d$ and $U_d$, are the droplet mass and velocity respectively, while $\vec{F}$ is the force acting on the droplet. The force acting on a droplet can be modelled as the drag force acting on a sphere plus the gravity contribution:

$$\vec{F} = -\frac{\pi D^2}{8} \rho C_d \mid \vec{U}_d - \vec{U} \mid (\vec{U}_d - \vec{U}) + m_d \vec{g} \tag{1.49}$$

With the following steps, it is possible to define the momentum relaxation time $\tau_u$, hence the time needed for the droplet to adjust its speed to the one of the flow:

$$\frac{d\vec{U}_d}{dt} = \frac{\vec{U}_d - \vec{U}}{\tau_u} + g \tag{1.50}$$

$$\tau_u = \frac{4}{3} \frac{\rho_d D}{\rho C_d \mid \vec{U}_d - \vec{U} \mid} \tag{1.51}$$

Going on with the energy equation, it is possible to write the conservation of the droplet sensible enthalpy $H_d$:

$$m_d \frac{dH_d}{dt} = \dot{m}_d H_v + \pi D k N u (T - T_d) f \tag{1.52}$$

The first term on the right is a contribution due to vaporization, in fact $H_v$ is the droplet heat of vaporization, which is function of temperature. The second term is due to convection and is function of the droplet Diameter D, and temperature T, together with the Nusselt number $Nu$ and the mass transfer factor $f$. The mass equation is not directly solved, because in the Lagrangian frame the mass is conserved automatically. However, the amount of mass loss from the particle due to evaporation can be modelled with the $D^2$ law, which states that the droplet evaporates at a rate proportional to its surface area and the square of its diameter decreases linearly with time.

$$\frac{dD^2}{dt} = C_{evap} \tag{1.53}$$

For this approach to work a lot of sub-models are required:

- Injection, to specify initial droplet diameter and velocity;
- Heat transfer with the gas phase;
- Evaporation;
- Atomization, to describe the primary breakup of the liquid jet and create secondary droplets;
- Break-up, to model droplet diameter reduction and deformation;
- Drag;
- Wall impingement;
- Droplet to droplet collision;
- Turbulent dispersion, which takes into account the interaction between the droplets and turbulent eddies.

The scope of this work is to develop a Lagrangian Wall film model to be integrated in OpenFOAM v8, as will be explained in the following chapters.

# 2 | State of the art

Liquid film formation and spreading on a solid surface is a complex phenomenon which involves a lot of physical events. The formation of a thin liquid layer happens in a lot of engineering relevant devices, in particular studies have been conducted for gas turbines fuel injection [5], water washing [1], and lubrication [10]. Other studies focused on train aerodynamics under rainfall environment [12], internal combustion engines fuel injection [11][28] and SCR devices [15] [16]. In the field of internal combustion engines, the creation of a thin film of fuel on a solid surface and subjecting that film to shear from high-velocity air flows is a common method of fuel atomization, given the big difference between its velocity and the one of the carrier phase [22]. On the other hand an excessive film formation would lead to unburned hydrocarbons. As a result, the processes governing fuel film formation and spreading are of considerable interest to fuel injector designers. The same considerations can be extended to the design of urea injectors in an SCR device.

## 2.1. Physical phenomenon

The physical mechanisms involved in the formation and evolution of a liquid wall film are:

- droplets impingement on a solid surface;
- splashing phenomena;
- liquid film dragging under the effect of the carrier phase;
- evaporation;
- droplets separation from the film due to geometrical discontinuities or waves.

All these aspects are summarized in Figure 2.1.

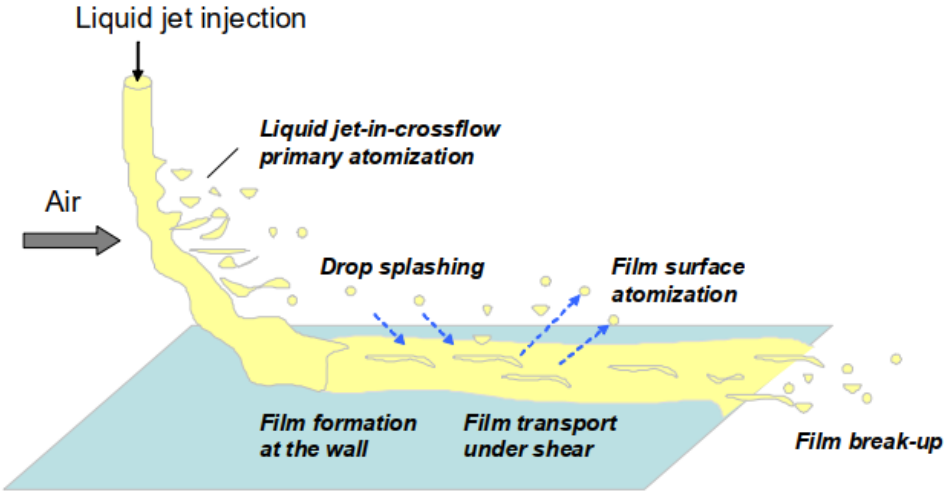



Figure 2.1: Physical mechanism of a wall film [22].

The formation of a liquid film on a flat surface and its shear-driven displacement due to cross-flow are experimentally evaluated in [22]. Here, the tested configurations are classified with a regime map proposed by [26] which is based on jet-to-crossflow momentum-flux ratio ($q$) and aerodynamic Weber number ($We_a$).

$$q = \frac{\rho_d V_d^2}{\rho_a V_a^2} \tag{2.1}$$

$$We_a = \frac{\rho_a V_a^2 d}{\sigma} \tag{2.2}$$

As it can be seen from Figure 2.2, as $We_a$ increases, the mechanism of primary atomization shifts from a column breakup mode to a shear stripping mode. On the other hand, the momentum-flux ratio is an indicator of liquid jet penetration in the carrier phase.

In [22] all the experiments lie in the surface and shear breakup regimes. In the former droplet stripping from the liquid column occurs because of the shear generated by the carrier phase, while in the latter droplets are stripped due to instabilities generated by the liquid momentum.

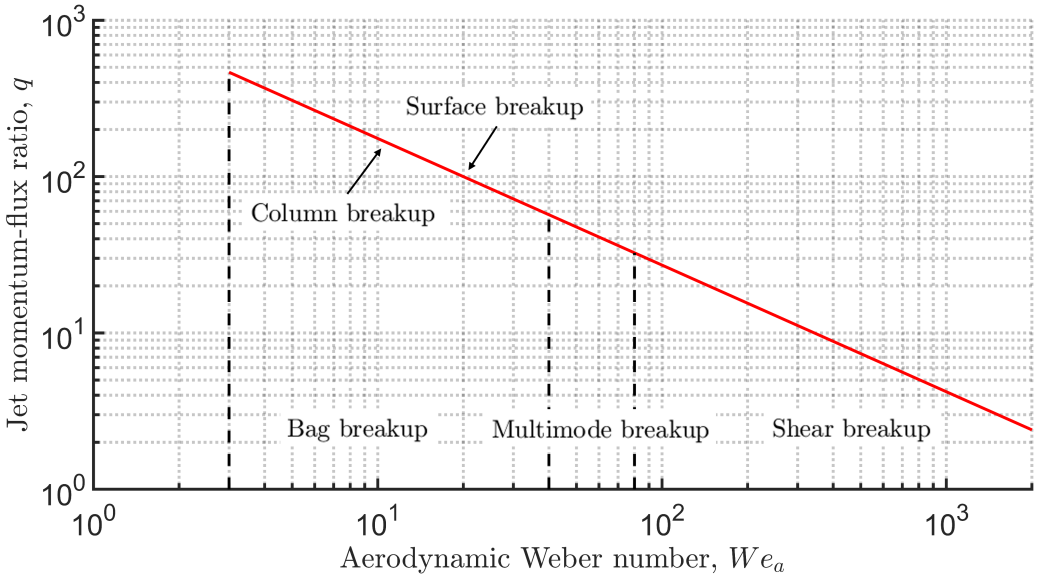



Figure 2.2: Atomization regime map as proposed by [26].

Some of the experiments are represented in Figure 2.3: in each row $We_a$ is kept constant and $q$ decreases. As expected from the regime map, increasing $q$ increases the liquid penetration in the cross flow, while the increase of $We_a$ is correlated with the shift from column breakup to shear stripping breakup. From Figure 2.3 it can also be seen that, for $q \geq 20$ the liquid jet impinged on the surface while still being a column, hence without a primary breakup, while for $q \leq 8$, the spray didn't appear to impinge on the surface at all. The same conditions of Figure 2.3 are reported in Figure 2.4, when seen from the bottom, through the transparent surface of the filmer plate.

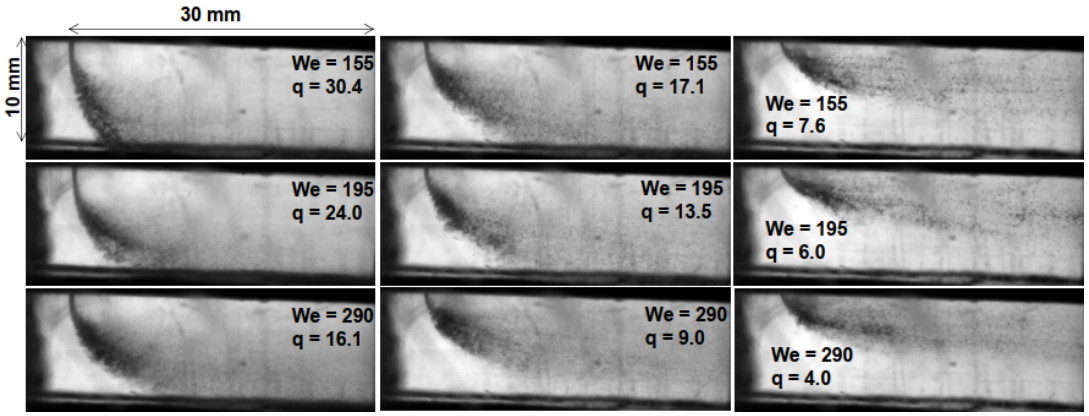



Figure 2.3: Spray trajectory and penetration as function of $We_a$ and $q$ [22].

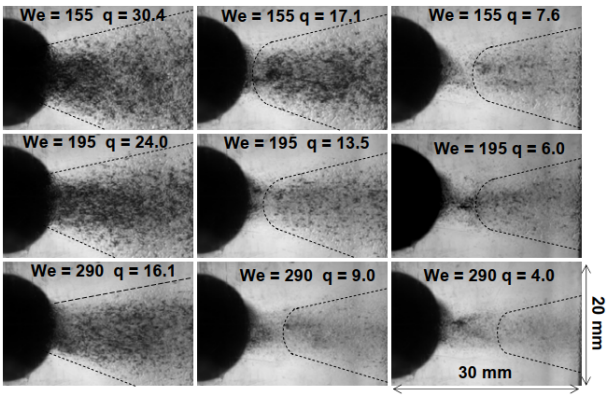
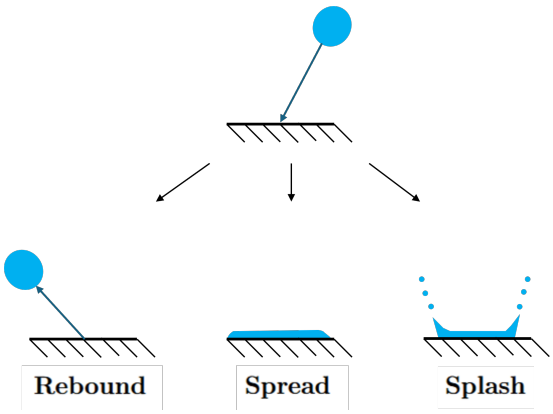



Figure 2.4: Film formation as function of $We_a$ and $q$ [22]. The semi-circle on the left is the nozzle and the dashed lines are the boundaries of the film observed throughout the experiment

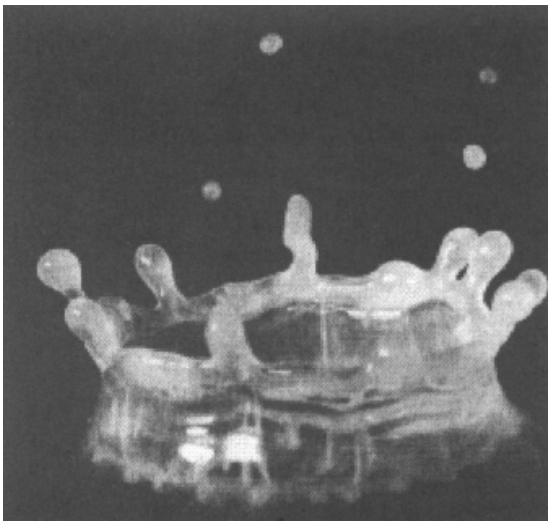
At this point it is worth analysing about what happens when a droplet hits a wall: the impact can have many different outcomes according to liquid properties, surface conditions, and kinematic parameters. The most important dimensionless group governing this phenomenon are the following [13]:

- Weber number:
$$We_d = \frac{\rho V^2 d_0}{\sigma}$$
- Reynolds number:
$$Re_d = \frac{\rho V d_0}{\mu}$$
- Ohnesorge number:
$$Oh = \frac{\mu}{\sqrt{\rho \sigma d_0}}$$
- Surface roughness:
$$S_l = \frac{R_t}{d_0}$$
- Film thickness:
$$\delta = \frac{h}{d_0}$$

Where $d_0$ is the initial diameter of the parcel, $R_t$ the mean surface roughness of the wall surface and all the quantities are referred to the dispersed (liquid) phase. When considering a droplet impinging on a solid surface, the possible outcomes of the impact are the ones of Figure 2.5 [21, 27].




(a) Possible impact regimes



(b) Splash: crown development and secondary droplets creation[23]

Figure 2.5: Impact of a drop on a solid surface.

Considering a single droplet hitting a wall, if the dimensionless parameter $K = Oh \cdot Re_d^{1.25}$ is below a threshold $K_{crit}$ (which depends on the surface conditions, namely $\delta$ and $S_l$), the droplet is completely deposited (Spread condition in Figure 2.5). If $K > K_{crit}$, the lamina of fluid that deposits on the surface is unstable and generates secondary droplets (Splash in Figure 2.5 [13]. Other criteria for spread-splash transition have been developed from [27] and [24], but the equivalence with the one just explained is contained in [14]. The presence of a thin film ($\delta > 0$) on the target surface causes a redirection of the out-flowing fluid underneath the droplet in a direction normal to the wall. Increasing $\delta$ increases the dissipation of kinetic energy at the impact, hence $K_{crit}$ becomes higher [14]. On the other hand an impact on a rough surface ($S_l > 10^{-3}$) lowers the value of $K_{crit}$ because roughness act as a "trigger" for the fluid to change direction.

The importance of the wall film in the atomization process can be inferred qualitatively from Figure 2.6: in all the depicted cases, a relevant quantity of fine droplets was generated. For lower values of $q$, the generation of fine mist is due to the film surface atomization, while in the cases where $q \geq 20$, a second distribution of droplets appears due to the liquid film flowing off the surface and breaking up. Surface film atomization is triggered by two mechanisms: splashing and film instabilities. Splashing occurs when a droplet impinges on a surface and spreads, forming a thin lamella which detaches from the wall after forming a crown. Film instabilities involves the formation on the film of small ripples that detach; an hypothesis to explain this phenomenon is that these ripples experience a Kelvin-Helmotz instability [8].

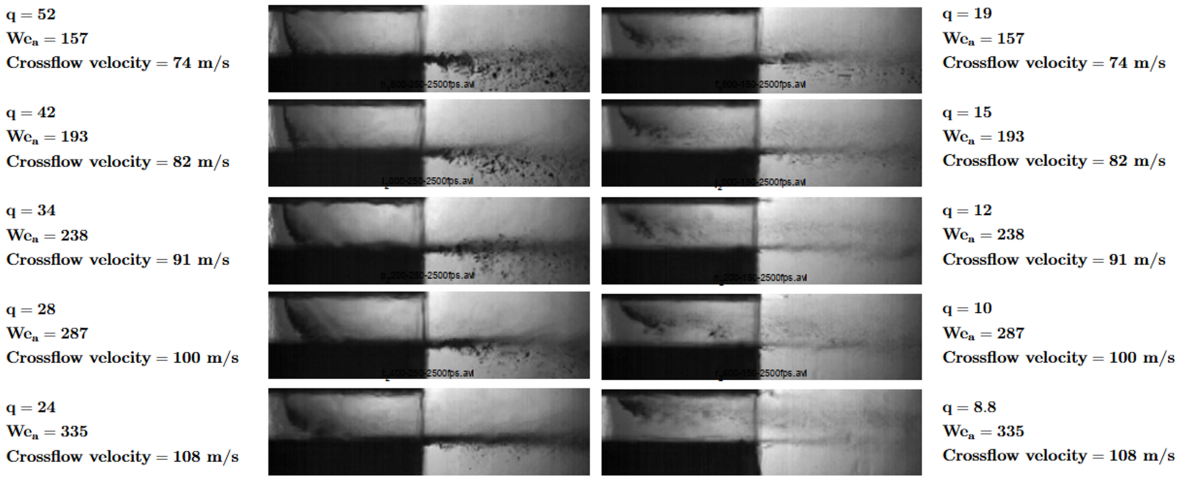



Figure 2.6: Qualitative visualization of variations in atomization due to changes in gas and liquid jet flow [22].

The flow of thin liquid films around corners, which may cause the phenomenon of detachment seen for $q \geq 20$ (also called stripping) has been analysed by [18]. This paper formulates a criterion based on the balance of inertial ($F_i$), body ($F_b$) and surface ($F_s$) forces. If the sum of these three contributions is negative or zero, the film detaches from the surface:

$$F_{\text{net}} = F_s + F_i + F_b \tag{2.3}$$

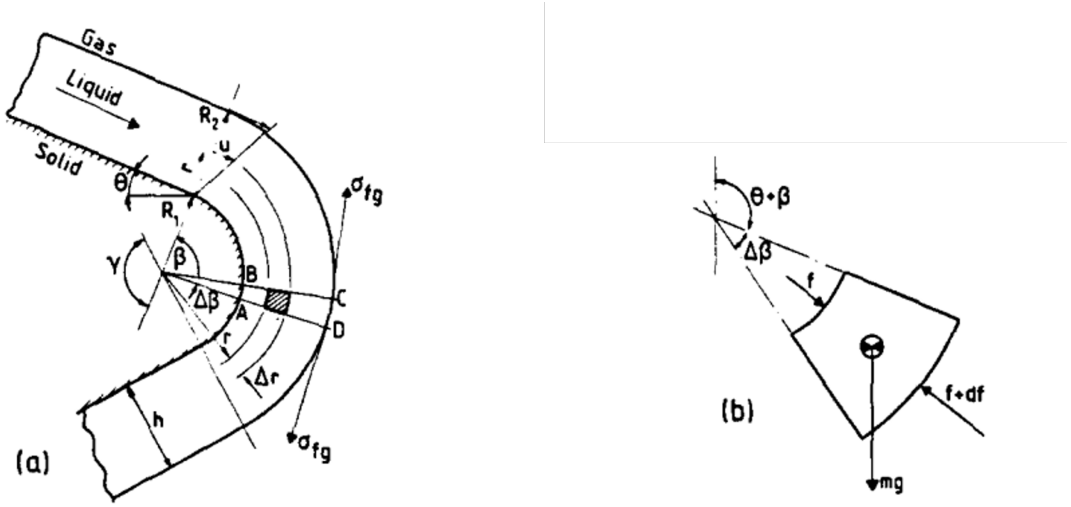



Figure 2.7: Film flow analysis around a corner [18].

Figure 2.7 is a representation of a film of uniform thickness $\delta$, around a corner of radius $R_1$. Under these conditions, surface forces tend to keep the liquid film attached and flowing

along the corner, whereas inertial forces promote its detachment from the surface. The role of gravity depends on the relative orientation between the gravitational acceleration and the corner geometry: it may either enhance film separation or, conversely, stabilize the film and delay detachment. The contribution of surface forces can be evaluated as:

$$F_s = \frac{\sigma}{R_2} \qquad where : R_2 = R_1 + \delta \tag{2.4}$$

The inertial force depends on the density $\rho$, film height $\delta$, and the internal curvature radius ($R1$):

$$F_i = -\frac{72}{60}\delta\rho|U^2|\frac{1}{R_1} \tag{2.5}$$

The body force can be expressed as:

$$F_b = -\frac{1}{2}\rho|\vec{g}|\frac{R_1^2 - R_2^2}{R_1}cos\beta \tag{2.6}$$

## 2.2. Eulerian wall film sub-model

Once the impingement regime of each particle hitting the wall is known, it is possible to compute the mass of liquid that forms the film. There are two approaches to model liquid films and they are both Eulerian. In other words, all the lagrangian parcels that impinge on a wall with an impact condition which is compatible with the formation of a film are removed and their mass, momentum and energy is transferred to specific solution fields on the surface of the wall. The representation of these fields is what differentiates the two available model.

### 2.2.1. Finite Area Method

The Finite Area Method (FAM) is a variant of the Finite Volume Method (FVM) that operates on 2D surfaces in a 3D domain. In this case, the Finite Area mesh is simply the boundary patch of the computational domain on which the film forms. The conservation equations are applied to each wall film cell: integrating in the direction orthogonal to the surface ($n_w$ in Figure 2.8) and under the assumptions of a thin film, the equations are reduced to a 2D domain [23]. This is possible if:

- The gradients in the tangential direction are negligible with respect to the ones in the normal direction (boundary layer approximation).
- liquid pressure $p_L$ is the sum of: gas $p_g$, droplet impact $p_d$, hydrostatic $p_h$ and

capillary $p_\sigma$ pressures.

- Film motion is caused by the spatial variation in the tangential direction of $p_L$, shear (at the wall and free surface), tangential momentum provided by incoming droplets (which is treated as a source term in the eulerian wall film formulation) and body forces.

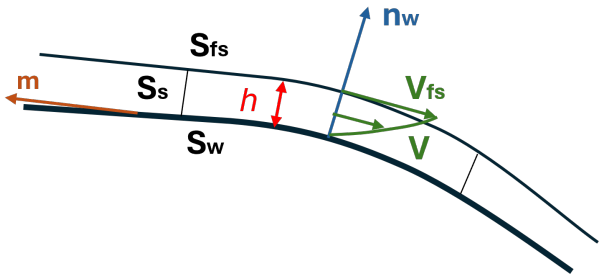



Figure 2.8: Finite area representation of a thin wall film.

This is done in order to reduce computational cost and mesh sensitivity of the solution, as resolving the wall film equations in 3D would lead to the need of an extremely refined grid at the wall [6]. With the hypothesis just explained, the generic liquid film conservation equation can be written as [17] [3]:

$$\int_{S_W} \frac{\partial h}{\partial t} dS + \oint_{\partial S_W} hm \cdot \overline{v} dL = \int_{S_W} \frac{\dot{m}_s}{\rho_L} dS + \int_{S_W} \frac{\dot{m}_V}{\rho_L} dS \tag{2.7}$$

Where the first term on the left represents an unsteady term, the second is a convection term computed with the film velocity tangential to the wall surface ($m$ direction) and averaged in the normal ($n_w$) direction (see Figure 2.8). The two terms on the right side of the equation are the mass source due to impingement and the mass sink due to evaporation. The film continuity equation of the single face area can be expressed as a function of film thickness given the incompressibility of the fluid:

$$\frac{\partial h}{\partial t} = \nabla \cdot (h\mathbf{U}_f) = S_M + S_V \tag{2.8}$$

In an analogous way the momentum equation can be expressed as:

$$\frac{\partial h\mathbf{U}_f}{\partial t} + \nabla \cdot (h\mathbf{U}_f\mathbf{U}_f) = -\frac{1}{\rho_f}\nabla_s(hp_f) + \tau_g - \tau_w + hg_t + S_U \tag{2.9}$$

Where $\tau_g$ and $\tau_w$ represent the shear force at the interface with the gas and the wall respectively and $g_t$ is the tangential component of gravity. $S_U$ accounts for the momentum

given by the incoming droplets and $\nabla_s(hp_f)$ is the surface gradient of the pressure. If the modelled spray is multicomponent, also a system of $N-1$ conservation equations must be solved, where $N$ is the number of chemical species in the spray. For instance, in an SCR case, the spray is composed by water and urea, so one species tracking equation is needed. Finally, the energy equation, expressed as sensible enthalpy can be given:

$$\frac{\partial hH_{s,f}}{\partial t} + \nabla \cdot (h\mathbf{u}_f H_{s,f}) = j_g - j_w + S_H \tag{2.10}$$

The source terms are due to heat exchanged with the gas ($j_g$), with the wall ($j_w$) and with the droplets entering the film ($S_H$). It must be noted that in ($j_g$) there is also a contribution due to the latent heat of vaporization of the mass fraction that evaporates.

### 2.2.2. Finite Volume Method

This approach is similar to the one just described, but the first layer of cell extrusion is used to apply the conservation laws of the film. To apply this method, a good boundary layer extrusion in the meshing process is needed. This is not trivial to obtain with snappyHexMesh or other non boundary based meshing tools. The main advantage of this approach with respect to the finite-area formulation described above lies in the possibility of exploiting all the data structures and numerical infrastructure already available for finite-volume simulations within the software.

## 2.3. Lagrangian wall film sub-model

As already anticipated, the scope of the present work is to develop a Lagrangian wall film sub-model for OpenFOAM. A particle-based Lagrangian approach has been developed in [19] and extended in [20] for KIVA-3. The same references have also been used by Ansys Fluent developers to write their model [2]. This model is based on the hypothesis of thin film, which implies a series of assumptions:

- film thickness is much smaller than radii of curvature of the walls and the characteristic distances along the wall surface over which mean properties vary;
- the flow of the film is laminar, velocity is always tangent to the wall $(\vec{U}_f - \vec{U}_w)\cdot\vec{n} = 0$ and vary linearly in the direction normal to the wall;
- air velocity is assumed much greater than film velocity, so for the computation of air velocity the film will be assumed to be still.

### 2.3.1. Wall film equations

As already seen in subsection 1.4.2, in discrete particle methods mass is conserved and the only variation is due to evaporation. To compute the wall film velocity $\overline{U_f}$, the following momentum equation must be solved for each particle inside the film:

$$\underbrace{\rho_d h \frac{d\vec{U}_d}{dt}}_{\text{(a)}} + \underbrace{h \nabla_s p_f}_{\text{(b)}} = \underbrace{\tau_w \vec{t}}_{\text{(c)}} - \underbrace{\mu_d \frac{\vec{U}_p - \vec{U}_w}{h/2}}_{\text{(d)}} + \underbrace{\dot{P}_{\text{imp}}}_{\text{(e)}} - \underbrace{\dot{M}_{\text{imp}} \vec{U}_p}_{\text{(f)}} + \underbrace{\vec{F}_n \vec{n}}_{\text{(g)}} + \underbrace{\rho h g}_{\text{(h)}} . \tag{2.11}$$

The terms in Equation 2.11 represent:

(a) Inertia term.

(b) Tangential force due to surface pressure gradient acting on the film.

(c) Tangential shear stress applied by the wall on the film.

(d) Viscous resistance in the film.

(e) Momentum input from impinging droplets.

(f) Momentum exchange associated with mass added by impingement.

(g) All normal forces contributions, determined by the constraint $(\vec{U}_f - \vec{U}_w) \cdot \vec{n} = 0$.

(h) Body force due to gravity.

In [19], the energy equation is solved considering the mean film temperature, $\overline{T}_l$ and the temperature profile is considered to be piecewise linear from the wall temperature $\overline{T}_w$ to $\overline{T}_l$ and from $\overline{T}_l$ to the gas surface temperature $\overline{T}_g$. With this approximation, the resulting energy equation to be solved for each particle in the film is the following:

$$\underbrace{\dot{Q}}_{\text{(a)}} = \underbrace{\lambda_d \left( \frac{T_s - \overline{T_l}}{h/2} \right)}_{\text{(b)}} + \underbrace{\dot{M}_{\text{vap}} \mathcal{L}}_{\text{(c)}} \tag{2.12}$$

The terms in Equation 2.12 represent:

(a) Gas side heat transport.

(b) Heat conduction through the liquid film.

(c) Latent heat of vaporization.

## 2.4. OpenFOAM lagrangian particle model

The wall-film formulation developed in this thesis is implemented within the *OpenFOAM* 8 framework, extending its Lagrangian infrastructure. For this reason, before introducing the specific model and algorithm proposed in the present work, it is necessary to briefly examine the structure of the Lagrangian library provided by *OpenFOAM*.

In OpenFOAM, Lagrangian parcels are represented through templated parcel classes that store the particle state and are organised into cloud objects, responsible for their evolution and coupling with the Eulerian fields. Additional physical phenomena such as forces, injection, and interaction with boundaries are handled by modular sub-models that are selected at run time and instantiated within the cloud. This architecture, based on a combination of compile-time templating and run-time polymorphism, provides both flexibility and computational efficiency, but it also determines how new models must be integrated into the solver.

Since the wall-film model developed in this thesis builds upon and modifies this existing infrastructure, a concise overview of the relevant components of the OpenFOAM Lagrangian framework is provided in the following. The focus is placed on the hierarchy of parcel and cloud classes and on the instantiation of the principal sub-models, as these elements constitute the basis on which the proposed implementation is constructed.

This section therefore serves as a bridge between the review of existing modelling strategies and the numerical methodology presented in chapter 3, where the Lagrangian wall-film model developed for this work is described in detail.

At the lowest level of the Lagrangian library lies the `particle` class, which is responsible for purely geometric tracking. This class stores the fundamental topological and geometric information required to advance a particle through the computational mesh, such as its position, the index of the cell containing it, and the face through which it may cross during tracking. Further details on the tracking procedure are reported in section 3.2.

One level above `particle`, *OpenFOAM* introduces the `KinematicParcel` class, which is templated on the base particle type and extends it by adding the physical quantities required for momentum computation. In particular, this class stores the particle diameter, velocity and density and provides the formulation of the equations of motion. Within this level, the main kinematic sub-models such as drag forces and basic wall-interaction models are instantiated and applied during the particle evolution. In the OpenFOAM framework, a parcel represents a Lagrangian computational entity that corresponds to multiple identical physical particles. This is accounted for through the variable `nParticle`, which

specifies how many real particles are represented by a single computational parcel.

Using the same templated approach, additional layers can be introduced to account for further physics. The `ThermoParcel` class extends the kinematic description by including thermodynamic properties and heat-transfer models, while the `ReactingParcel` class adds the treatment of species transport and chemical reactions. At the top of this hierarchy, the `SprayParcel` class incorporates atomization and breakup models typically required in spray simulations. This layered structure allows different levels of physical fidelity to be composed at compile time while maintaining a common tracking infrastructure.

Parcels are not evolved individually but are grouped within a `Cloud` object, which represents the main container and manager of all Lagrangian elements. The cloud class stores and organises the list of parcels, advances them in time, and handles their coupling with the Eulerian fields. It is also responsible for reading the Lagrangian properties from the case dictionaries and for instantiating the various sub-models required for the simulation, such as injection, forces, dispersion, and patch-interaction models. During the simulation loop, the cloud handles the injection of new parcels, the tracking and update of existing ones, and the accumulation of source terms exchanged with the carrier phase in the `evolve` function.

Similarly to parcels, cloud classes follow a hierarchical structure:

Starting from a generic `Cloud` template, specialised classes such as `KinematicCloud`, `ThermoCloud`, `ReactingCloud`, and `SprayCloud` are obtained by combining the appropriate parcel type with the corresponding set of sub-models. The resulting cloud type therefore defines both the physical behaviour of the parcels and the collection of models acting on them. This modular and templated architecture is central to the extensibility of the OpenFOAM Lagrangian framework and determines how new models such as the wall-film formulation proposed in this thesis must be integrated within the existing infrastructure [7]. Figure 2.9 show the `SprayParcel` and Cloud classes implemented through this nested template definition.

```
namespace Foam
{
    typedef SprayParcel
    <
        ReactingParcel
        <
            ThermoParcel
            <
                KinematicParcel
                <
                    particle
                >
            >
        >
    > basicSprayParcel;

    template<>
    inline bool contiguous<basicSprayParcel>()
    {
        return false;
    }
}
```

(a) Definition of spray parcel

```
namespace Foam
{
    typedef SprayCloud
    <
        ReactingCloud
        <
            ThermoCloud
            <
                KinematicCloud
                <
                    Cloud
                    <
                        basicSprayParcel
                    >
                >
            >
        >
    > basicSprayCloud;
}
```

(b) Definition of spray cloud

Figure 2.9: Spray Parcel and Cloud classes implemented through nested template definitions [7]

# 3 | Methodology

## 3.1. Patch interaction model

As a starting point for the creation of the Lagrangian wall film, the patch interaction model had to be modified. In *OpenFOAM*, a patch interaction model defines how Lagrangian particles interact with boundary patches of the computational domain. In other words, the patch interaction model is called when a Lagrangian particle reaches a boundary patch and describes the interaction law with that patch. The first version of this custom patch interaction model was done by modifying LocalInteraction.C, available in *OpenFOAM* 8. This model requires a specific interaction type input for each patch and it is possible to choose among:

- stick: the parcel is stopped at the contact point.
- rebound: the normal velocity of the parcel (with respect to the patch plane), $Un$, is reflected.
- escape: the parcel is removed from the computational domain.

The goal was to create and add a new type of interaction called film, used to keep the particles on the boundary, but letting them free to move on the plane of the patch. To achieve this, it was necessary to create two new parcel variables, the first one being the particle index regime (which was set to film for particles inside the wall film, "p.pIRegime") and the boundary face on which the particle was ("p.faceWF"). Operatively, this was done as explained in Algorithm 3.1.

The setup just explained is enough to handle kinematics, but as explained in section 2.1, the impingement of a droplet on a surface is a complex phenomenon which involves also thermodynamic aspects. In *OpenFOAM*, all patch interaction models are Kinematic sub-models, meaning they can't access to thermodynamic quantities. Thermodynamic sub-models are instantiated within a thermodynamic cloud (e.g. ThermoCloud), which extends the kinematic cloud by introducing energy and mass-transfer capabilities. For this reason a thermodynamic patch interaction model had to be created and instantiated on

Algorithm 3.1 Film case in Local Interaction Model

Retrieve mesh and film thickness field on boundary faces
Set dry threshold $h_{\text{dry}}$
Obtain patch normal $\vec{n}_w$ and patch velocity $\vec{U}_p$
Compute parcel velocity relative to patch and decompose parcel velocity:

$$\vec{U} \leftarrow \vec{U} - \vec{U}_p \qquad U_n = \vec{U} \cdot \vec{n}_w, \qquad \vec{U}_t = \vec{U} - U_n \vec{n}_w$$

Set $U_n = 0$
**if** parcel regime is *film* **then**
    **if** parcel changed wall face **then**
Update stored wall face index
        **if** film thickness $h_f > h_{\text{dry}}$ (wet face) **then**
Maintain tangential motion in film
        **else**
Apply capillarity friction:

$$\vec{U} = \vec{U} - \mu \vec{U}_t$$

        **end if**
    **end if**
**else**
Parcel impinging for the first time: store current face as wall-film face
    **if** film thickness $h_f > h_{\text{dry}}$ (wet impingement) **then**
Deposit into film without friction loss
    **else**
Apply capillarity friction:

$$\vec{U} = \vec{U} - \mu \vec{U}_t$$

    **end if**
Accumulate normal momentum flux If $U_n > 0$
Set parcel regime to *film*
**end if**
Transform velocity back to global frame:

$$\vec{U} \leftarrow \vec{U} + \vec{U}_p$$

the ThermoCloud, in this way it has access to temperature, viscosity and surface tension of the parcel. If the parcel is already in Film regime, the steps are the same as for local Interaction (so that the patch attraction force doesn't generate fictitious atomization, see section 3.2). If the parcel is not a film parcel, the Bai and Gosman splash model is applied [4], as was already done in *OpenFOAM* for the lagrangian-eulerian wall film model described in section 2.2. In this model, the parcel may follow two paths according to the thickness of film, $\delta$, already present on the face: if $\delta < \delta_{threshold}$, the path is the one described in Algorithm 3.2, else the path is for wet impact, as described in Algorithm 3.3.

Algorithm 3.2 Dry splash interaction for Thermo Lagrangian surface film

Compute parcel velocity relative to patch and decompose parcel velocity:

$$\vec{U} \leftarrow \vec{U} - \vec{U}_p \qquad U_n = \vec{U} \cdot \vec{n}_w, \qquad \vec{U}_t = \vec{U} - U_n \vec{n}_w$$

Compute Laplace number:

$$La = \frac{\rho, \sigma, d}{\mu^2}$$

Compute Weber number:

$$We = \frac{\rho, |\vec{U}_n|^2, d}{\sigma}$$

Compute critical Weber number:

$We_c = A_{dry}, La^{-0.183}$ where $A_{dry} = 2630$: dry surface roughness coefficient [4]

1: **if** $We < We_c$ (Adhesion regime) **then**
Set parcel velocity to zero:

$$\vec{U} \leftarrow 0$$

Return to global frame:

$$\vec{U} \leftarrow \vec{U} + \vec{U}_p$$

Call absorption interaction

2: **else**
(Splash regime)
Sample splashing mass ratio:

$m_r = 0.2 + 0.6\,\xi$ where $\xi \in [0, 1]$ is a unIform random number

Call splash interaction function and pass:

$$(m_r,\ We,\ We_c,\ \sigma)$$

3: **end if**

Algorithm 3.3 Wet splash interaction for Thermo Lagrangian surface film

Compute parcel velocity relative to patch and decompose velocity:

$$\vec{U} \leftarrow \vec{U} - \vec{U}_p \qquad U_n = \vec{U} \cdot \vec{n}_w, \qquad \vec{U}_t = \vec{U} - U_n \vec{n}_w$$

Compute Laplace number:

$$La = \frac{\rho, \sigma, d}{\mu^2}$$

Compute Weber number:

$$We = \frac{\rho, |\vec{U}_n|^2, d}{\sigma}$$

Compute critical Weber number:

$$We_c = A_{wet}, La^{-0.183}$$

1: **if** $We < 2$ (Adhesion regime) **then**
Call absorption interaction
2: **else if** $2 \leq We < 20$ (Bounce regime) **then**
Compute incident angle:

$$\theta = \frac{\pi}{2} - \text{arccos!} \left( \frac{\vec{U}}{|\vec{U}|} \cdot \vec{n}_w \right)$$

Compute restitution coefficient:

$$\epsilon = 0.993 - \theta \left(1.76 - \theta \left(1.56 - 0.49, \theta\right)\right)$$

Update parcel velocity:

$$\vec{U} = -\epsilon, \vec{U}_n + \frac{5}{7}, \vec{U}_t$$

3: **else if** $20 \leq We < We_c$ (Spread regime) **then**
Call absorption interaction
4: **else**
(Splash regime)
Sample splashing mass ratio:

$$m_r = 0.2 + 0.8, \xi \qquad \text{where } \xi \in [0, 1] \text{ is a unIform random number}$$

Call splash interaction function and pass:

$$(m_r, ; We, ; We_c, ; \sigma)$$

5: **end if**

Algorithm 3.4 Splash interaction function

Determine two orthonormal tangential vectors to the wall:

$$\vec{t}_1 = \text{unit!}\left(\text{perp}(\vec{n}_w)\right), \qquad \vec{t}_2 = \vec{n}_w \times \vec{t}_1$$

Compute splashing mass:

$$m_{splash} = m \cdot m_r$$

Compute number of secondary droplets and mean splash diameter:

$$N_s = 5\left(\frac{We}{We_c} - 1\right) \qquad \bar{d}_s = \frac{1}{\sqrt[3]{6}}\sqrt[3]{\frac{m_r}{N_s}} \cdot d$$

Define diameter distribution bounds and normalization constant:

$$d_{max} = 0.9, \sqrt[3]{m_r}, d, \qquad d_{min} = 0.1, d_{max} \qquad K = e^{-d_{min}/\bar{d}_s} - e^{-d_{max}/\bar{d}_s}$$

1: **for** each secondary parcel $i = 1 \dots N_{parcels}$ **do**
Sample new diameter and Compute multiplicity:

$$d_i = -\bar{d}_s \,\text{ln!}\left(e^{-d_{min}/\bar{d}_s} - \xi K\right) \qquad \text{where } \xi \text{ random} \in [0,1] \qquad n_{p,i} = \frac{m_r \cdot n_p \cdot d^3}{d_i^3 \cdot N_{parcels}}$$

Accumulate surface energy of secondary parcels:

$$E_{\sigma,sec} \mathrel{+}= n_{p,i} \cdot \sigma \cdot A(d_i)$$

2: **end for**
Compute incident kinetic, surface and dissipated energy:

$$E_{K,in} = \tfrac{1}{2} m |\vec{U}_n|^2 \qquad E_{\sigma,in} = n_p, \sigma, A(d) \qquad E_d = \text{max!}\left(0.8E_{K,in}, \frac{n_p, We_c}{12}\pi\sigma d^2\right)$$

Compute available splash energy:

$$E_K = E_{K,in} + E_{\sigma,in} - E_{\sigma,sec} - E_d$$

3: **if** $E_K \leq 0$ **then**
Insufficient energy: call absorption interaction and terminate splash routine
4: **end if**

Algorithm 3.5 Splash interaction function (continued)

Compute auxiliary coefficients for velocity scaling:

$$\log d, \qquad c_2 = \log(d_1) - \log d \qquad c_1 = \sum_i \left(\log(d_i) - \log d\right)^2$$

Compute reference normal velocity magnitude:

$$|\vec{U} * n, 0| = \sqrt{\frac{2N * parcelsE_K}{m_{splash}\,(1 + c_1/c_2^2)}}$$

5: **for** each secondary parcel $i$ **do**
Sample splash direction:

$$\vec{e_i} = \text{splashDirection}(\vec{t_1}, \vec{t_2}, -\vec{n}_w)$$

Create new parcel by copying the incident parcel and assign:

$$d \leftarrow d_i, \qquad n_p \leftarrow n_{p,i}$$

Set velocity:

$$\vec{U} = \vec{e_i}\left(|C_f, \vec{U} * t| + |\vec{U} * n, 0|\frac{\log(d_i) - \log d}{c_2}\right)$$

Apply dimensional constraints If required
Insert parcel into cloud
6: **end for**
Transfer remaining mass to film:

$$m' = m - m_{splash}$$

Call absorption interaction with $m'$

Algorithm 3.6 Absorption interaction

Compute parcel velocity relative to patch and decompose parcel velocity:

$$\vec{U} \leftarrow \vec{U} - \vec{U}_p \qquad U_n = \vec{U} \cdot \vec{n}_w, \qquad \vec{U}_t = \vec{U} - U_n \vec{n}_w$$

Set $U_n = 0$

1: Set p.faceWF and p.pIRegime:

$$p.faceWF \leftarrow p.face() \qquad p.pIRegime \leftarrow \text{film}$$

2: Update film statistics:

$$n_{film} \mathrel{+}= 1$$

3: Keep parcel active in cloud:

$$keepParticle \leftarrow \text{true} \qquad p.active \leftarrow \text{true}$$

## 3.2. Patch attraction Force

In [19], the motion of film parcels is handled by exploiting the logical coordinate system adopted in KIVA-3. Each computational cell in KIVA-3 is defined as the image, under a trilinear mapping, of a unit cube in logical space. Accordingly, each cell is associated with its own mapping and local logical coordinate system. The logical coordinates of a point $x$ within a cell, $x(\xi, \eta, \zeta)$, are the ordered triplet $(\xi, \eta, \zeta)$ of coordinates whose image under trilinear mapping correspond to $x$. As it is shown in Figure 3.1, when advancing in time, firstly a new provisional particle position $\widetilde{x}_p^{n+1}$ is computed by advancing the parcel in physical space: this movement is unconstrained. Then, the provisional point is mapped to the corresponding logical coordinates, $(\widetilde{\xi}, \widetilde{\eta}, \widetilde{\zeta})$: If all the values of the triplet are between 0 and 1, the parcel has remained in the same cell and the new position $x_p^{n+1}$ is simply the projection of $\widetilde{x}_p^{n+1}$ on the face on which the parcel was at $x_p^n$. Else, according to the values of the triplet, four cases may arise:

1. the particle moves to the same logical face of a neighbouring cell (like in Figure 3.1),
2. it turns an inside corner and changes face of the same cell,
3. it turns an outside corner and is assigned to a different face of a neighbouring cell,
4. crosses an open boundary and escapes the system.

In the first three cases, the cell and face indices are updated accordingly, and the logical coordinates are adjusted to remain consistent with the new cell mapping. In the last case, the parcel is removed from the simulation.

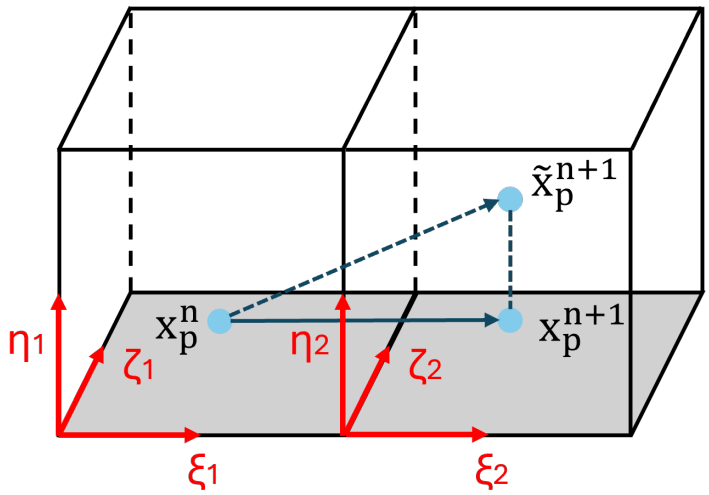



Figure 3.1: Representation of the projection algorithm implemented in [19].

In *OpenFOAM*, the tracking algorithm of the particles works differently: it tracks particles on an implicit decomposition of each cell into tetrahedra as shown in Figure 3.2. This choice was done by the developers of the software in order to provide more robustness with respect to mesh quality [9].

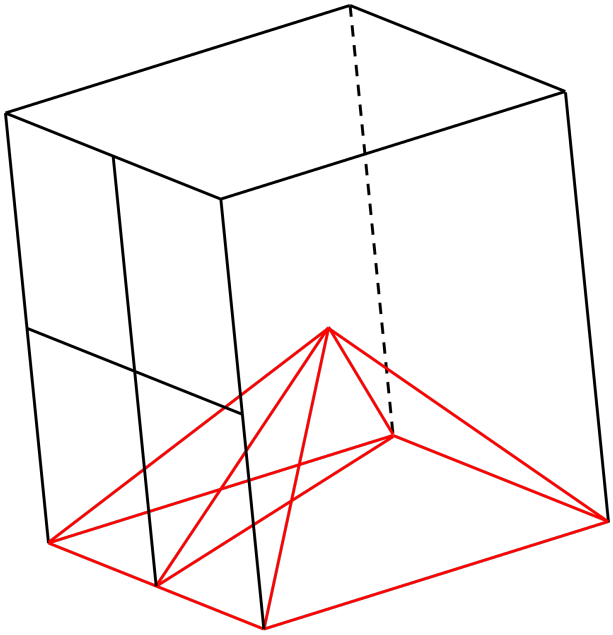


Figure 3.2: Tracking algorithm of *OpenFOAM*.

In this framework, trying to apply the "projection" algorithm of O'Rourke as it is, would raise portability issues for the library and generate a code very difficult to maintain. To overcome this, the patch attraction force has been created: in this way it was possible

to obtain the same result by simply adding a new force model instead of acting inside the "basic" lagrangian class and the tracking algorithm. The algorithm Algorithm 3.7 describes the patch attraction force. Moreover, this force is needed for computational purposes: it is possible to know on which boundary face a parcel is only If it collides with it and calls the patch interaction model (which updates p.faceWF).

Algorithm 3.7 Patch attraction force

1: **if** *p.pIRegime* is in *film* **and** not already located on a boundary face **then**
IdentIfy the nearest boundary face to the particle position
Get $\vec{n}$ outward unit normal of the boundary face
Get $\vec{U}$ particle velocity
Compute the normal distance:

$$d = (\vec{x}_{face} - \vec{x}_p) \cdot \vec{n}$$

Compute the normal velocity component:

$$U_n = \vec{U} \cdot \vec{n}$$

Correct the distance:

$$d_c = d - U_n \Delta t$$

Evaluate the corrective force ensuring parcel–patch impact at the end of the time step:

$$\vec{F}_{pa} = m \, \frac{2 d_c}{\Delta t^2} \, \vec{n}$$

**return** $\vec{F}_{pa}$
2: **end if**

## 3.3. Update Film

The main purpose of the function Update Film is to sum the mass of the parcels that are in film regime and to compute a representative tangential velocity on each boundary face. First, the function resets the film thickness increment field and then loops over all parcels, identifying those whose interaction regime corresponds to film. For each parcel on a boundary face, it updates the local film thickness contribution based on parcel mass, density, and face area, while simultaneously accumulating the tangential momentum on that face. In a second stage, the function computes a mass-weighted mean tangential velocity for each face with film parcels on it. This averaged velocity is then imposed back onto all parcels residing on that face, ensuring consistency between parcel motion and the evolving wall-film layer. Overall, the function enforces a localized momentum redistribution and thickness update for parcels transitioning into a surface film, providing a bridge between discrete Lagrangian tracking and surface film dynamics. To perform this localized accumulation, the algorithm constructs a face-based occupancy structure that associates each boundary face with the parcels currently residing on it in film regime. Specifically, a list of dynamic pointer lists is created, where each outer list entry corresponds to a boundary face and stores the pointers to the parcels attached to that face. This data structure, referred to as filmOccupancy, is rebuilt at every time step to ensure consistency with the instantaneous parcel distribution. The adoption of this architecture provides several advantages. First, it enables direct face-wise access to parcels without requiring repeated global searches over the entire cloud, thereby reducing computational overhead. Second, by storing pointers rather than copies, the structure avoids memory duplication and guarantees that any modification applied to a parcel is immediately reflected in the cloud. Finally, reconstructing the structure at every time step ensures robustness in cases of parcel deletion, regime switching, or mesh updates.

Algorithm 3.8 Film update and parcel velocity redistribution

Set film thickness field to zero on all boundary faces

1: **for** each parcel $p$ in the cloud **do**

2: **if** *p.pIRegime* corresponds to *film* **then**

Determine boundary patch index and local face index

Compute deposited film thickness:

$$\Delta h_f \mathrel{+}= \frac{m}{A_f \, \rho_p}$$

Compute face unit normal $\vec{n}$

Decompose parcel velocity:

$$U_n = \vec{U} \cdot \vec{n}, \qquad \vec{U}_t = \vec{U} - U_n \vec{n}$$

Accumulate tangential momentum on face:

$$\vec{P}_t \mathrel{+}= \vec{U}_t \, m_p \, n_p$$

3: **end if**

4: **end for**

5: **if** tangential momentum field $\vec{P}_t$ exists **then**

Retrieve list of parcels located on each boundary face

6: **for** each boundary face **do**

Compute total parcel mass on face and mean tangential velocity:

$$M_f = \sum m_p \, n_p, \qquad \vec{U}_{f,\text{mean}} = \frac{\vec{P}_t}{M_f}$$

Assign $\vec{U}_{f,\text{mean}}$ to all parcels on that face

7: **end for**

8: **end if**

## 3.4. Drag force

When inside the film, particles interact both with the carrier phase and the boundary, as shown in Figure 3.3. So, the sphere drag model which is designed for dispersed particles inside the carrier is not representative for the film parcels. The film drag model accounts for the two contributions, making the hypothesis of linear velocity profile between the patch and the centroid of the cell. Each parcel receives a contribution of these forces which is proportional to its mass, as described in Algorithm 3.9.

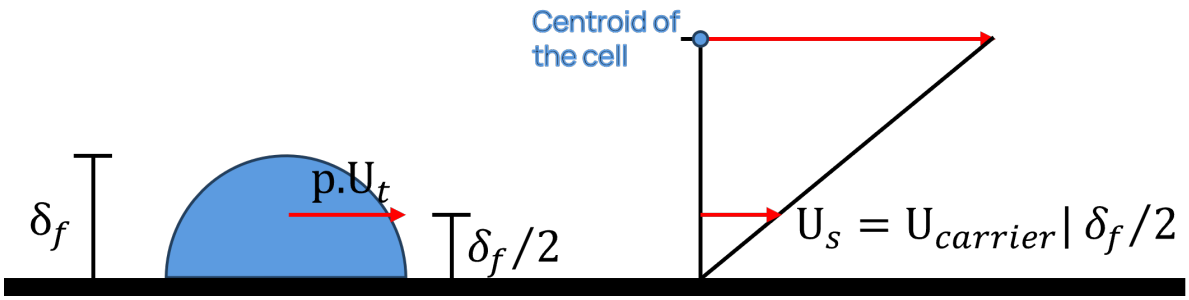



Figure 3.3: Drag force scheme.

$$F_g = -\mu_p \frac{\mathrm{p.U}_t - \mathrm{U}_s}{\frac{\delta_f}{2}} \cdot mr \cdot A \tag{3.1}$$

$$F_w = -\mu_p \frac{\mathrm{p.U}_t}{\frac{\delta_f}{2}} \cdot mr \cdot A \tag{3.2}$$

Algorithm 3.9 Film drag force

**if** *p.pIRegime* is *film* **then**
Retrieve wall face associated with the parcel and film thickness field, $\delta_f$
Compute face area $A_f$ and outward unit normal $\vec{n}$
Get parcel velocity $\vec{U}_p$ and decompose it:

$$U_n = \vec{U}_p \cdot \vec{n}, \qquad \vec{U}_t = \vec{U}_p - U_n \vec{n}$$

Compute total parcel mass already present on the face and parcel mass ratio:

$$m_f = \sum_{i \in \text{p on face}} m_i, \qquad mr = \frac{m}{m_f}$$

Retrieve gas velocity $\vec{U}_c$ in the owner cell of the wall face and decompose it:

$$U_{cn} = \vec{U}_c \cdot \vec{n}, \qquad \vec{U}_{ct} = \vec{U}_c - U_{cn} \vec{n}$$

Compute normal distance between cell center and face center:

$$d = (\vec{x}_f - \vec{x}_c) \cdot \vec{n}$$

Estimate slip velocity at the film mid-plane and compute relative gas–parcel tangential velocity:

$$\vec{U}_s = \vec{U}_{ct} \frac{\delta}{2d}, \qquad \vec{U}_{rel} = \vec{U}_s - \vec{U}_t$$

Compute viscous shear coefficient:

$$K_\mu = mr \frac{\mu_l A_f}{\delta/2}$$

**if** $K_\mu < \frac{m}{\Delta t}$ **then**
Apply viscous drag using film shear model

$$\vec{F}_{gas} = K_\mu \vec{U}_{rel}, \qquad \vec{F}_{wall} = -K_\mu \vec{U}_t$$

**else**
Limit drag to avoid velocity reversal (overshoot) within one time step

$$\vec{F}_{gas} = \frac{m}{\Delta t} \vec{U}_{rel}, \qquad \vec{F}_{wall} = -\frac{m}{\Delta t} \vec{U}_t$$

**end if**
Compute total film drag force:

$$\vec{F}_{fd} = \vec{F}_{gas} + \vec{F}_{wall}$$

**return** $\vec{F}_{fd}$
**end if**

## 3.5. Heat exchange

When a parcel enters the surface film, its heat exchange model must be modified both in terms of equations and in the definition of the effective heat-transfer area. In the free stream, the parcel exchanges heat only with the carrier phase and the characteristic surface area is evaluated from the parcel diameter. Once the parcel is deposited onto a wall film, however, it participates in a collective interaction over the wall face on which it resides. The relevant exchange area is therefore no longer the particle surface but a fraction of the wall-face area, proportional to the ratio between the parcel volume and the total volume of parcels present on that face. Within the film, the parcel simultaneously exchanges heat with the carrier phase through convection across the film interface and with the wall through conduction across the liquid layer. The resulting thermal balance becomes:

$$p.mass \cdot C_p \cdot \frac{dT}{dt} = h_g A_w \left(T_g - p.T\right) + \frac{k_l A_w}{\frac{\delta_f}{2}} \left(T_w - p.T\right) + Q_s \tag{3.3}$$

where $A_w$ denotes the effective wall-face area assigned to the parcel, $h_g = Nu \cdot \frac{K_g}{\frac{\delta_f}{2}}$ is the convective coefficient evaluated with half of the film half-thickness as characteristic length, and $Q_s$ represents any additional heat source (e.g. radiation or user-defined terms). The temperature evolution implemented in the function is written in a linear form that can be handled efficiently by the parcel temperature integrator. After dividing the energy balance by the parcel thermal inertia $mC_p$, the governing equation is rearranged as

$$\frac{dT}{dt} = a - b\left(p.T\right) \tag{3.4}$$

In this formulation, the coefficient $b$ collects all contributions that are proportional to the parcel temperature $p.T$. In practice, $b$ is obtained by summing the heat–transfer coefficients multiplied by the effective exchange area and dividing by $mC_p$. For a parcel in the carrier phase,

$$b = \frac{h_{tc} A_s}{mC_p}, \tag{3.5}$$

while for a parcel inside the wall film,

$$b = \frac{h_{tc} A_w}{mC_p} + \frac{k_l A_w}{\left(\delta_f/2\right) mC_p}. \tag{3.6}$$

The coefficient $a$ gathers all terms that do not multiply the parcel temperature. These include the same heat–transfer mechanisms weighted by the external temperatures (carrier temperature $T_c$ and wall temperature $T_w$), together with any additional source terms such

as radiation or user-defined heating. In the carrier phase,

$$a = b\,T_c + \frac{Q_s + Q_{\text{rad}}}{mC_p}, 16 \tag{3.7}$$

whereas inside the film,

$$a = b_{\text{conv}} T_c + b_{\text{cond}} T_w + \frac{S_h + Q_{\text{rad}}}{mC_p}. \tag{3.8}$$

The temperature integrator, which is kept the same for both the models, advances the solution of the linear ordinary differential equation

$$\frac{dT}{dt} = a - bT \tag{3.9}$$

over the time step $\Delta t$.The integrator returns the temperature increment $\Delta T$ over the time step, which is then decomposed into a coupled contribution (associated with convection and conduction) and a non-coupled contribution (from independent sources). This splitting allows the code to update consistently both the parcel temperature and the enthalpy source terms exchanged with the Eulerian fields.

Algorithm 3.10 calcHeatTransfer function

**if** filmHeatTransfer inactive **then return** $T$
**end if**
**if** $faceWF = -1$ **then** ▷ Parcel in carrier phase
  $A_s \leftarrow \text{surfaceArea}(d)$
  $h_{tc} = \text{convectiveHTC}(d, Re, Pr, k)$
  $b_{cp} = \dfrac{h_{tc} A_s}{m C_p}$
  $a_{cp} = b_{cp} T_c$
  **if** radiation active **then**
    Add radiative source to $a_{ncp}$
  **end if**
  $a_{ncp} = \dfrac{a_{ncp}}{m C_p}$
  $\Delta T = \text{Integrator}(T, dt, a_{cp} + a_{ncp}, b_{cp})$
  $\Delta T_{ncp} = a_{ncp}\, dt$
  $\Delta T_{cp} = \Delta T - \Delta T_{ncp}$
  $T_{new} \leftarrow T + \Delta T$
  Limit $T_{new}$ to $[T_{min}, T_{max}]$
  Update enthalpy source terms **return** $T_{new}$
**else** ▷ Parcel inside wall film
  Retrieve film thickness $\delta_f$
  Retrieve wall temperature $T_w$
  $A_w = \dfrac{p.Volume}{\delta_f}$ ▷ $A_w$ is fraction of face area proportional to parcel volume
  $h_{tc} = \text{filmConvectiveHTC}(k, \delta_f)$
  $k_l =$ liquid conductivity
  ▷ Convection with carrier phase
  $b_{\text{conv}} = \dfrac{h_{tc} A_w}{m C_p}$
  ▷ Conduction with wall
  $b_{\text{cond}} = \dfrac{k_l A_w}{(\delta_f/2)\, m C_p}$
  $b_{tot} = b_{\text{conv}} + b_{\text{cond}}$
  $a_{tot} = b_{\text{conv}} T_c + b_{\text{cond}} T_w$
  **if** radiation active **then**
    Add radiative source using $A_w$
  **end if**
  $a_{ncp} \leftarrow \dfrac{a_{ncp}}{m C_p}$
  $\Delta T = \text{Integrator}(T, dt, a_{tot} + a_{ncp}, b_{tot})$
  $\Delta T_{ncp} = a_{ncp}\, dt$
  $\Delta T_{cp} = \Delta T - \Delta T_{ncp}$
  $T_{new} \leftarrow T + \Delta T$
  Limit $T_{new}$ to $[T_{min}, T_{max}]$
  Update enthalpy source terms **return** $T_{new}$
**end if**

## 3.6. Film Cloud

All the models presented previously require the algorithm to determine whether a parcel is in the free stream or in the film regime. Performing this check repeatedly at every time step for every Lagrangian element introduces a non-negligible overhead and can slow down the simulation. For this reason, a dedicated library has been developed specifically for the Lagrangian wall–film treatment.

The main feature of this library is the introduction of a `FilmCloud` and a corresponding `FilmParcel` class, implemented at the same abstraction level as the `SprayCloud` and `Parcel` classes (see Figure 2.9). This separation allows each cloud to manage its own physics and avoids repeated regime checks within individual submodels.

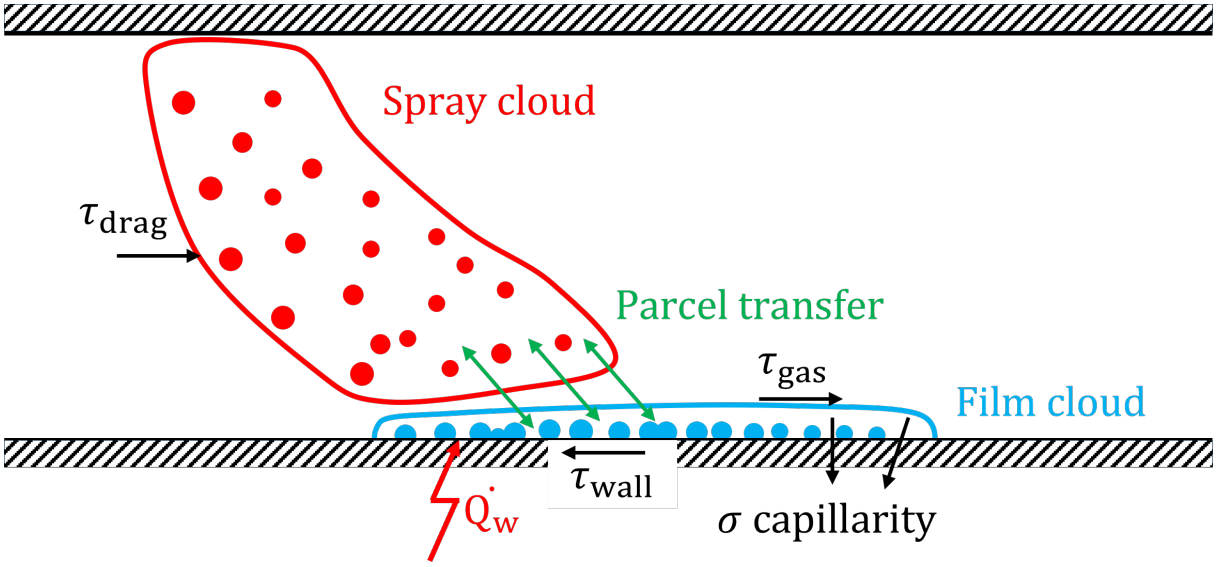



Figure 3.4: Film cloud.

Operationally, the solver, derived from a slightly modified version of `sprayFoam`, advances the solution as follows at each time step: first the `SprayCloud` is evolved, then the `FilmCloud`, and finally a transfer routine is invoked. This transfer function loops over all parcels and performs the following operations:

- if a spray parcel has `p.pIRegime == film`, it is transferred to the `FilmCloud`;
- if a film parcel has `p.pIRegime == freeStream`, it is transferred back to the `SprayCloud`.

The details of the transfer procedure from `SprayCloud` to `FilmCloud` are described in Algorithm 3.11.

This architecture eliminates the need to perform regime checks within each force model

and patch-interaction model. Instead, model selection is handled independently for each cloud through their respective properties dictionaries, ensuring a clear separation at the configuration level. As an additional safeguard, the `patchAttraction` and `filmDrag` forces have been implemented as film-specific sub-models, preventing their accidental application to the `SprayCloud`.

The Bai-Gosman patch interaction model described in section 3.1 has likewise been decomposed into two distinct implementations, one dedicated to the spray cloud and the other to the film cloud.

Moreover, the introduction of the `FilmCloud` allows the film update routine to be encapsulated within that class. As a consequence, the update procedure operates exclusively on film parcels, avoiding unnecessary iterations over free-stream parcels.

Algorithm 3.11 transferSpraytoFilm

$nTransferred = 0$
Create empty dynamic list $parcelsToRemove$
**for all** parcels $p_A$ in $SprayCloud$ **do**
▷ Transfer criterion: parcel ready to move from A to B
    **if** $p_A.pIRegime() = film$ **then**
        $particleMass \leftarrow p_A.mass() \cdot p_A.nParticle()$ ▷ Mass used for statistics
        Create a new particle $p_B$ in $FilmCloud$
        Copy all the properties of $p_A$ in $p_B$
        $massTransferredAtoB \leftarrow massTransferredAtoB + particleMass$
        Append $p_A$ to $parcelsToRemove$
        $nTransferred \leftarrow nTransferred + 1$
    **end if**
**end for**
▷ Remove transferred parcels from source cloud
**for all** $p$ in $parcelsToRemove$ **do**
    Delete $p$ from $SprayCloud$
**end for**
$nTransferredAtoB \leftarrow nTransferredAtoB + nTransferred$
**return** $nTransferred$

### 3.6.1. Template specialization

The introduction of the second cloud enables a template specialization strategy. As shown in Figure 3.5, and in contrast with Figure 2.9, the basicFilmParcel type is templated on a FilmThermoParcel. This design allows the heat transfer model described in section 3.5 to be applied directly and exclusively to film parcels at compile time, while the base heat transfer model remains associated with free-stream parcels. Consequently, no runtime check on pIRegime() is required to determine which thermal model should be applied, eliminating conditional branching and improving both clarity and computational efficiency.

Similarly, the Film Cloud is templated on a FilmKinematicCloud. This ensures that operations specific to the wall-film phase—such as the construction of filmOccupancy are inherently restricted to film parcels. As a result, the occupancy-building procedure can iterate directly over film parcels without repeatedly verifying their interaction regime before inserting them into the face-based pointer lists.

```
namespace Foam
{
    typedef FilmParcel
    <
        ReactingParcel
        <
            FilmThermoParcel
            <
                KinematicParcel
                <
                    particle
                >
            >
        >
    > basicFilmParcel;

    template<>

    inline bool contiguous<basicFilmParcel>()
    {
        return false;
    }
}
```

(a) Definition of film parcel

```
namespace Foam
{
    typedef FilmCloud
    <
        ReactingCloud
        <
            ThermoCloud
            <
                FilmKinematicCloud
                <
                    Cloud
                    <
                        basicFilmParcel
                    >
                >
            >
        >
    > basicFilmCloud;
}
```

(b) Definition of film cloud

Figure 3.5: Film Parcel and Cloud classes implemented through nested template definitions

# 4 | Model testing

In the first section of this chapter, a test case is introduced. This configuration was specifically conceived for showing the capabilities of this model and does not rely on experimental validation; its purpose is to highlight the effects of heat transfer and gravity on the liquid film. To test the accuracy and efficiency of this model, it has been applied to a well-known experiment, first investigated in [22] and later reproduced numerically in [1]. The simulations were performed using both the Eulerian wall film model available in OpenFOAM and the model developed in the present work.

## 4.1. Cylinder test case

As already mentioned at the beginning of this chapter, the following section presents a test case specifically created for this study, aimed at isolating and showing the effects of heat transfer and gravitational forces on the liquid film model.

### 4.1.1. Domain discretization and initial conditions

The computational domain consists of a cylinder whose geometrical dimensions are reported in Figure 4.1. The mesh is fully structured and was generated through a dedicated *blockMesh* script; the main mesh statistics are summarized in Table 4.1. The principal boundary conditions are listed in Table 4.2.

The $k$–$\omega$ *SST* turbulence model was adopted for the reasons previously discussed, and the boundary conditions for the turbulent quantities are consistent with those used in the previous test case.

To emphasize the effect of heat transfer within the liquid film, the cylinder wall temperature was fixed at 450 K, while the injected parcels were initialized at a temperature of 320 K.

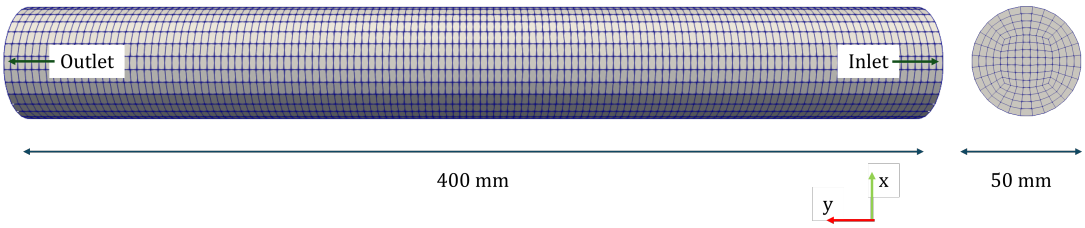



Figure 4.1: Cylinder domain dimensions and mesh.

| n. cells | non orthogonality (max/avg) | max skewness |
|---|---|---|
| 2,15e4 | 26,06° / 5,80° | 0,44 |

Table 4.1: Cylinder Mesh statistics

| Patch name | BC for U | BC for p | BC for T |
|---|---|---|---|
| Inlet | fixedValue (5 m/s) | zeroGradient | fixedValue (450 K) |
| Outlet | zeroGradient | fixedValue (1e6 Pa) | zeroGradient |
| Wall | noSlip | zeroGradient | zeroGradient |

Table 4.2: Boundary conditions for p, U and T

### 4.1.2. Results

In the following images, the particles are scaled according to their diameter and coloured by temperature. The snapshots are taken in the XY plane. A sharp temperature variation can be observed when the parcels impact the wall, due to the conductive heat transfer term. Furthermore, as time progresses, the liquid film accumulates at the bottom of the cylinder under the action of gravity and gradually evaporates because of the elevated wall temperature.

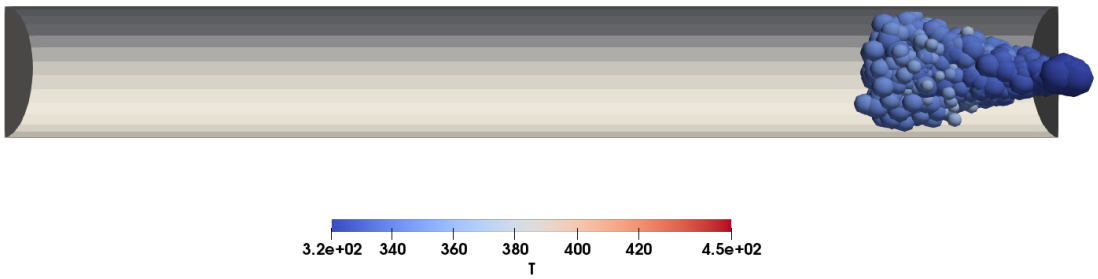



Figure 4.2: Particles in XY plane at $t = 0,005s$, parcels are coloured by temperature and scaled by diameter

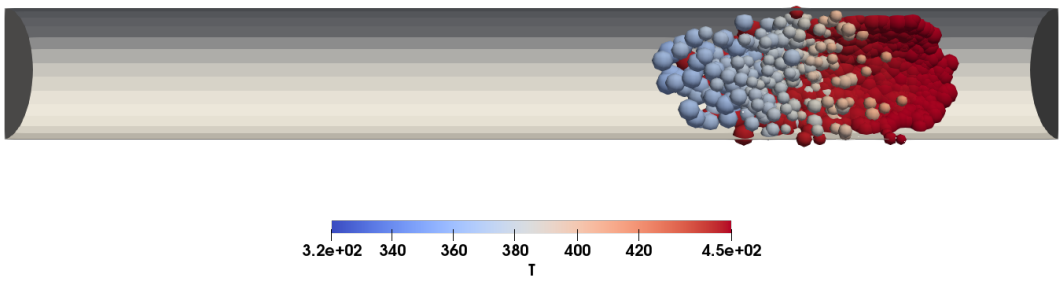



Figure 4.3: Particles in XY plane at $t = 0,016s$, parcels are coloured by temperature and scaled by diameter

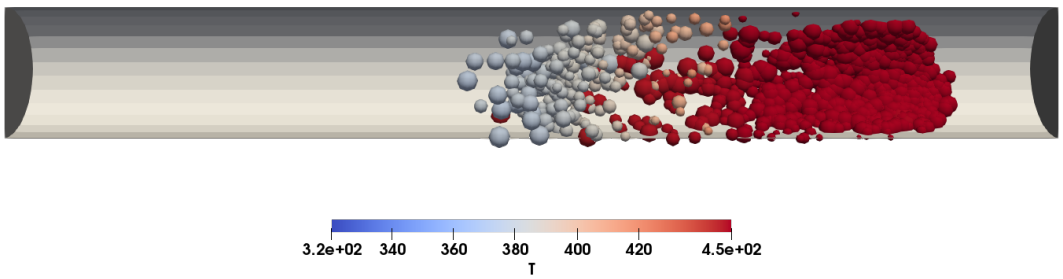



Figure 4.4: Particles in XY plane at $t = 0,030s$, parcels are coloured by temperature and scaled by diameter

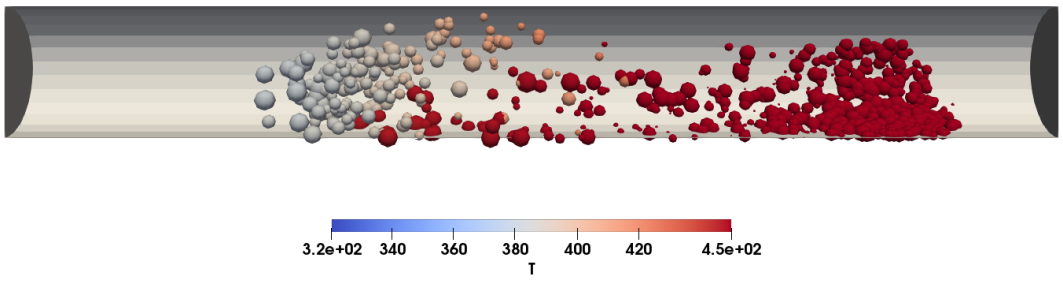



Figure 4.5: Particles in XY plane at $t = 0,045s$, parcels are coloured by temperature and scaled by diameter

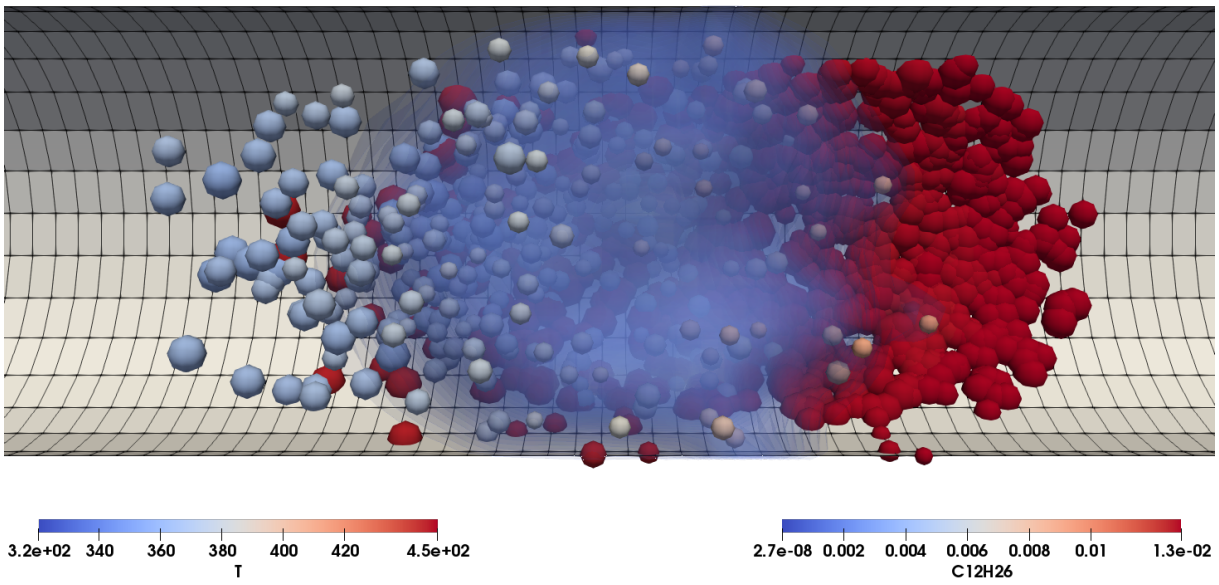



Figure 4.6: Particles in XY plane at $t = 0,016s$, parcels are coloured by temperature and scaled by diameter. Iso-countours of the mass concentration of C12H26.

In Figure 4.6, corresponding to the same time step shown in Figure 4.3, iso-contours of the C12H26 mass fraction are reported. Since the initial condition consists of a gas phase composed purely of air, these contours can be interpreted as an indicator of spray evaporation. The spatial distribution clearly shows that the wall film evaporates more rapidly than the free-stream spray. Moreover, the generated vapour is transported downstream by the convective motion of the flow.

## 4.2. Experimental adiabatic validation

### 4.2.1. Domain discretization and initial conditions

The mesh is the same for both the cases: its dimensions are represented in Figure 4.7 and the most relevant statistics can be found in Table 4.3. The experimental setup is shown in Figure 4.8.

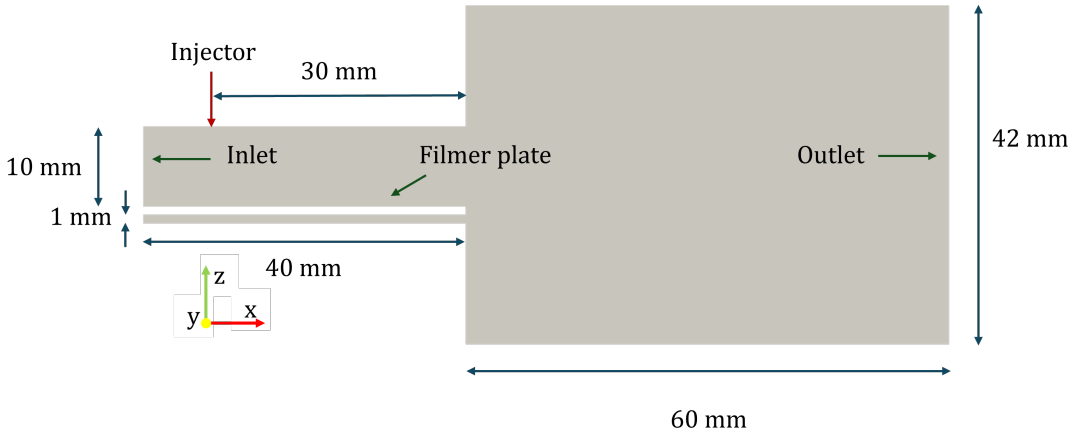



Figure 4.7: Domain dimensions.

| n. cells | non orthogonality (max/avg) | max skewness |
|---|---|---|
| 6,73e6 | 25,24° / 2,87° | 0,33 |

Table 4.3: Mesh statistics

In order to provide consistent initial conditions for the transient Lagrangian simulations, a preliminary steady-state computation of the single phase carrier flow was performed using the solver rhoSimpleFoam. This step allowed the carrier phase velocity, pressure, and thermodynamic fields to converge to a physically meaningful solution within the duct geometry. The converged steady-state fields were subsequently mapped as initial conditions for the transient simulations, thereby reducing numerical transients and improving solution stability during the early stages of particle injection. The $k$–$\omega$ $SST$ turbulence model was selected due to its robustness with respect to near-wall treatment. The boundary conditions applied in this preliminary simulation are summarized in Table 4.4 and Table 4.5. An assessment of the evolution of the flow variables indicated that a steady-state regime was effectively reached after 3500 pseudo time-steps. The velocity field ($U$) after those iterations is presented in Figure 4.9.

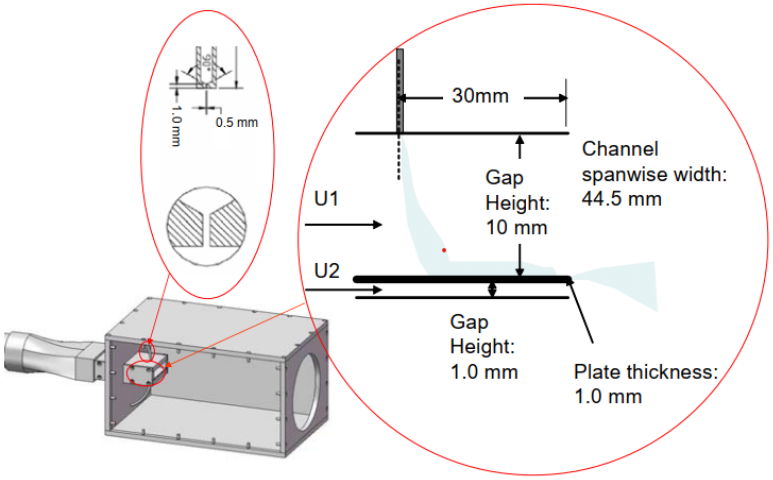



Figure 4.8: Experimental setup from [22].

| Patch name | BC for U | BC for p | BC for T |
|---|---|---|---|
| Inlet | fixedValue (82 m/s) | zeroGradient | fixedValue (288 K) |
| Outlet | zeroGradient | fixedValue (1e6 Pa) | zeroGradient |
| Walls | noSlip | zeroGradient | zeroGradient |
| Front & Back | symmetry | symmetry | symmetry |

Table 4.4: Boundary conditions for p, U and T (SS case)

| Patch name | BC for $\alpha_t$ | BC for k | BC for $\nu_t$ |
|---|---|---|---|
| Inlet | calculated (0,02 $Pa \cdot s$) | TIKEInlet (0,05 $m^2/s^2$) | inletOutlet (0,05 $m^2/s$) |
| Outlet | zeroGradient | zeroGradient | zeroGradient |
| Walls | alphatWallFunction | kqRWallFunction | nutkWallFunction |
| Front & Back | symmetry | symmetry | symmetry |

Table 4.5: Boundary conditions for turbulent quantities (SS case)

### 4.2.2. Lagrangian cloud properties

To enable a consistent comparison between the two wall-film models, both simulations employed the same injection configuration, as summarised in Table 4.6. Droplet breakup was disabled in all cases. Heat transfer was modelled only while parcels were in the free stream using the Ranz–Marshall correlation, whereas it was deactivated once the liquid entered the surface film. Reaction and combustion models were also turned off. Finally, both simulations were executed using a coupled solution approach in order to improve numerical accuracy. The comparison between the two models was carried out using this setup as a reference (among the 20 presented in the paper), because it is the only case in which the film thickness is reported at multiple locations on the filmer plate.

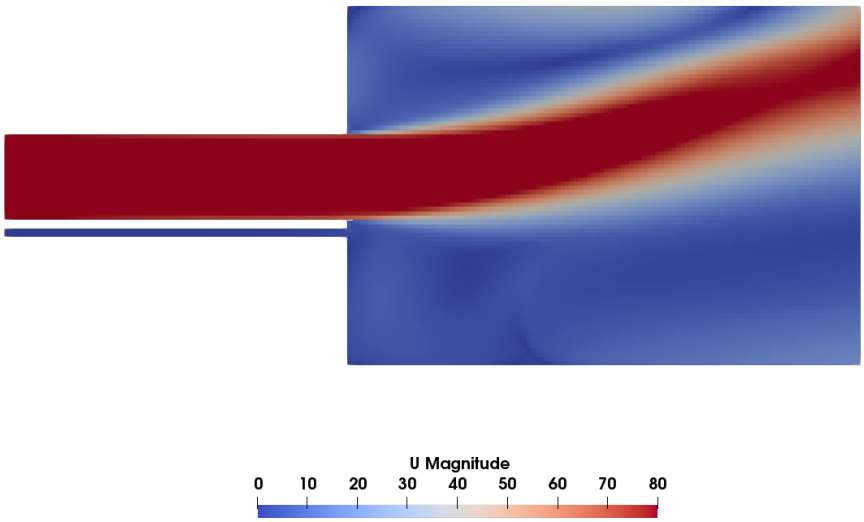



Figure 4.9: Steady state velocity field.

| Category | Value |
|---|---|
| *Injection model* | |
| Model type | coneInjection |
| Total injected mass | $2.5 \times 10^{-6}$ kg |
| Duration | $1.25 \times 10^{-3}$ s |
| Parcels per second | $2.0 \times 10^{6}$ $s^{-1}$ |
| Flow rate profile | constant |
| Discharge coefficient | 1 |
| *Injector geometry and orientation* | |
| Outer diameter | $5.0 \times 10^{-4}$ m |
| Direction | -Z |
| *Spray cone properties* | |
| Inner cone angle | 0° |
| Outer cone angle | 4° |
| *Droplet size distribution* | |
| Distribution type | uniform |
| Minimum diameter | $30 \times 10^{-6}$ m |
| Maximum diameter | $100 \times 10^{-6}$ m |

Table 4.6: injection model parameters.

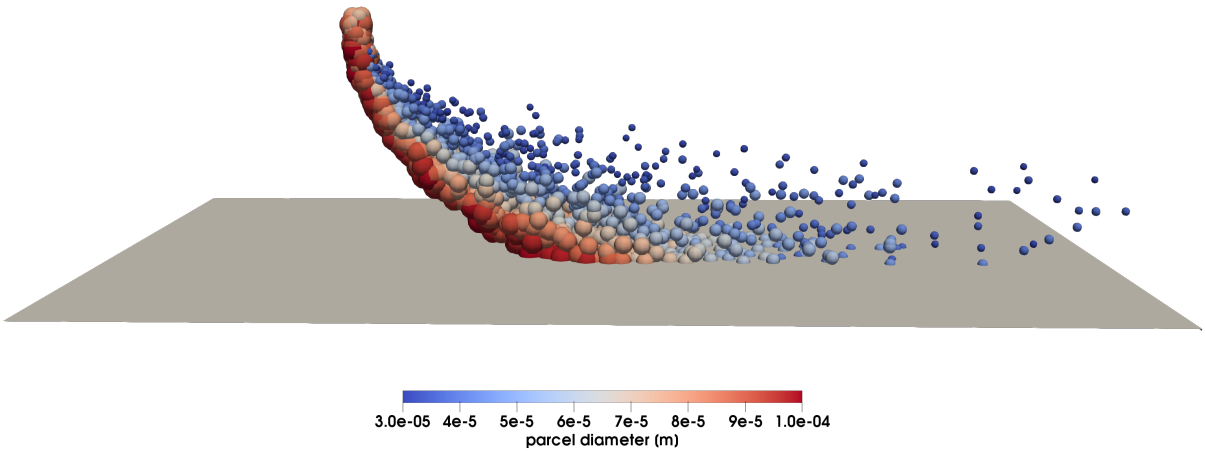



Figure 4.10: Snapshot taken at $t = 0,0008s$, the parcels are scaled and coloured by diameter.

### 4.2.3. Eulerian Wall film model

In this case, the chosen solver was `reactingParcelFoam`, a transient solver for compressible, turbulent flows that supports a reacting multiphase cloud coupled with an Eulerian surface-film model [7]. To enable wall-film modelling with this solver, a dedicated surface-film region must be generated and coupled to the primary flow region. This is achieved by selecting the wall faces on which the film is allowed to develop and extruding them into a thin computational mesh representing the film domain.

In the present setup, the faces belonging to the filmer plate were first collected into a `faceSet` and then converted into a `faceZone`. This face zone was subsequently used to generate the film region through an extrusion procedure normal to the wall. The resulting region, named `wallFilmRegion`, is therefore a secondary mesh attached to the selected wall faces and used to solve the two-dimensional surface-film transport equations.

The film mesh was generated using a linear normal extrusion with a single layer and uniform thickness. Although only one cell of $1 \times 10^{-4}$, m is extruded across the thickness, this region does not represent the physical liquid height; instead, it provides the computational domain in which the film thickness field is solved and evolved. The `nearestPatchFace` sampling mode was adopted to couple the film region with the primary flow, allowing momentum, heat, and mass exchange between the two regions.

This setup requires new boundary conditions for this region: these are all zeroGradient conditions apart from the velocity at the filmer plate which is set to zero (no slip condition).

### 4.2.4. Lagrangian Wall film model

For the Lagrangian wall-film formulation adopted in this work, it is sufficient to select the patch-interaction model described in section 3.1 and to include a 'filmCloudProperties' dictionary. This dictionary can be obtained by copying the 'sprayCloudProperties' file and adapting it to the selected film sub-models, namely to add film drag and patch attraction as forces and to select the lagrangian heat transfer model.

### 4.2.5. Results

In Figure 4.11 the film parcels are scaled and coloured by diameter, as expected bigger parcels reach the filmer plate closer to the injector point, while smaller particles are more affected by the drag of carrier phase. Figure 4.12 is the corresponding image in terms of filmDelta.

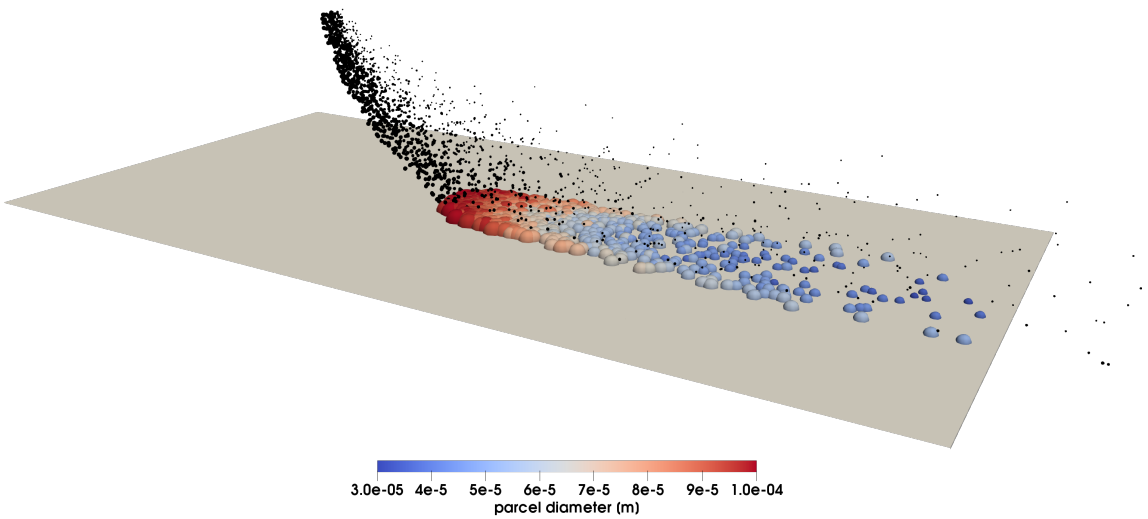



Figure 4.11: Snapshot taken at $t = 0,0014s$, the film parcels are scaled and coloured by diameter, the black parcel are spray (free stream) parcels.

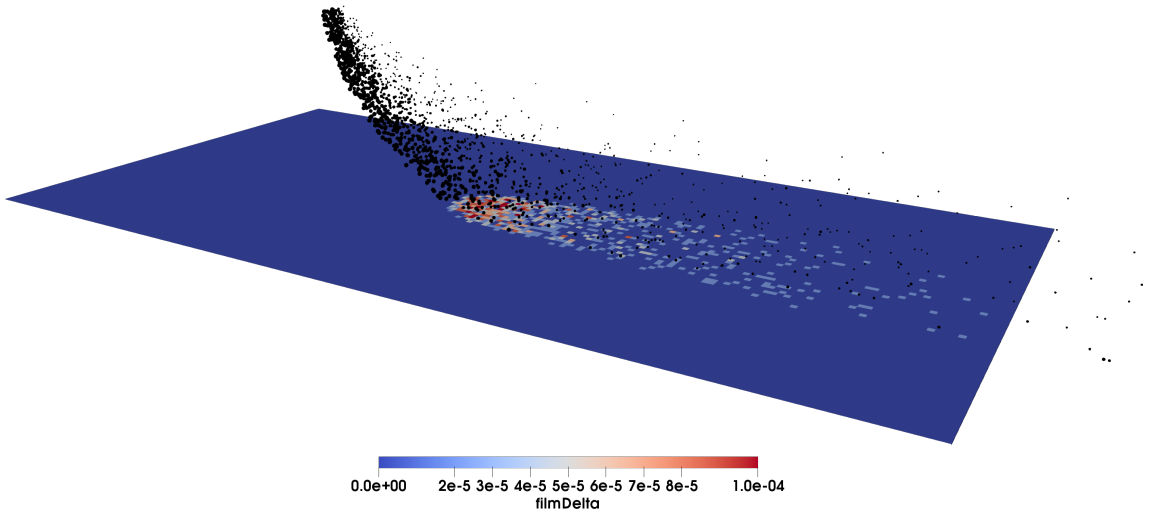



Figure 4.12: Snapshot taken at $t = 0,0014s$, film parcels are hidden and the plate faces are colored by filmDelta

From Figure 4.13 to Figure 4.17, the evolution of the wall film viewed from above is shown at time intervals of $(0.001s)$. The parcels are coloured according to their streamwise velocity component. It is possible to observe the initial accumulation of liquid following spray impingement and its subsequent downstream transport driven by the kinetic energy of incoming parcels and aerodynamic drag. As expected, parcels located near the central region of the film exhibit higher velocities, while those closer to the edges move more slowly.

It should be noted that, due to the Lagrangian formulation of the wall-film model, it was not possible to include a mechanism accounting for hydrostatic pressure gradients that would promote a continuous diffusion and redistribution of the film thickness, $(\delta)$. Furthermore, the discrete parcel-based representation makes it difficult to define a criterion preventing the complete drying of individual faces: once parcels are advected to neighbouring faces, the originating face may instantaneously become dry. Furthermore, in the present work it was not possible to include a curvature based separation criterion for the wall film, and this limitation is visible in Figure 4.17, where the film remains artificially attached to the surface despite the sharp edge and flow conditions that would promote liquid detachment.

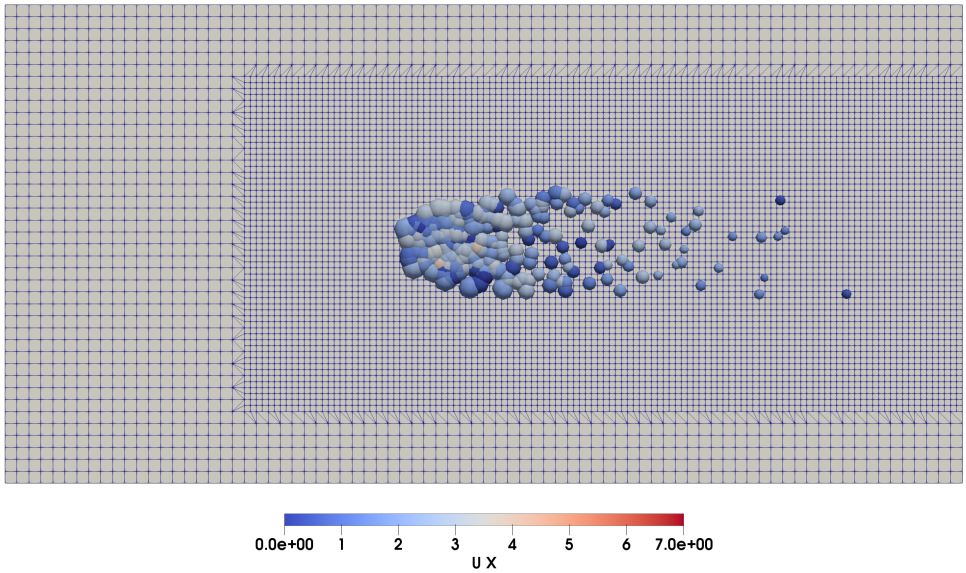



Figure 4.13: Film cloud seen from the top at $t = 0,001s$, parcels are coloured by velocity in X direction and scaled by diameter

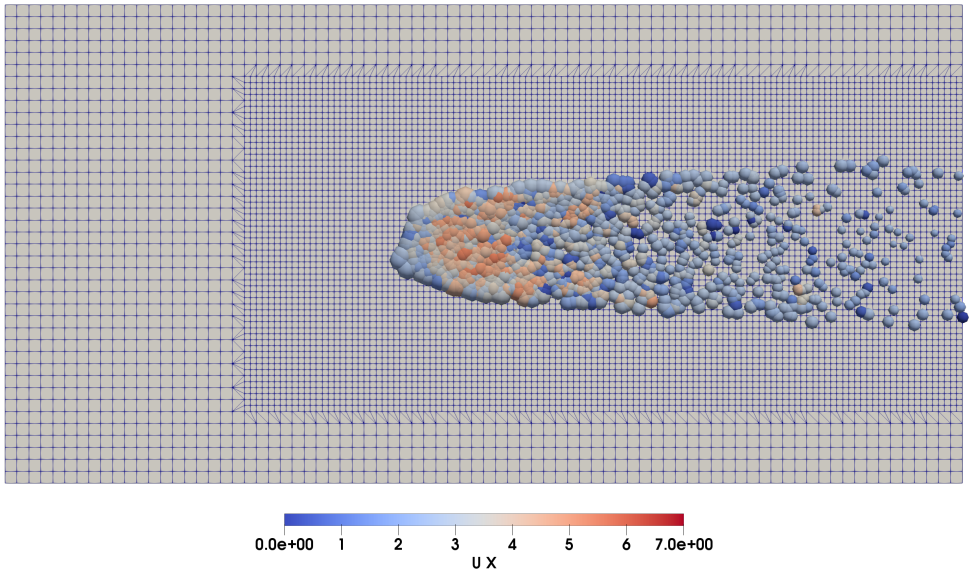



Figure 4.14: Film cloud seen from the top at $t = 0,002s$,, parcels are coloured by velocity in X direction and scaled by diameter

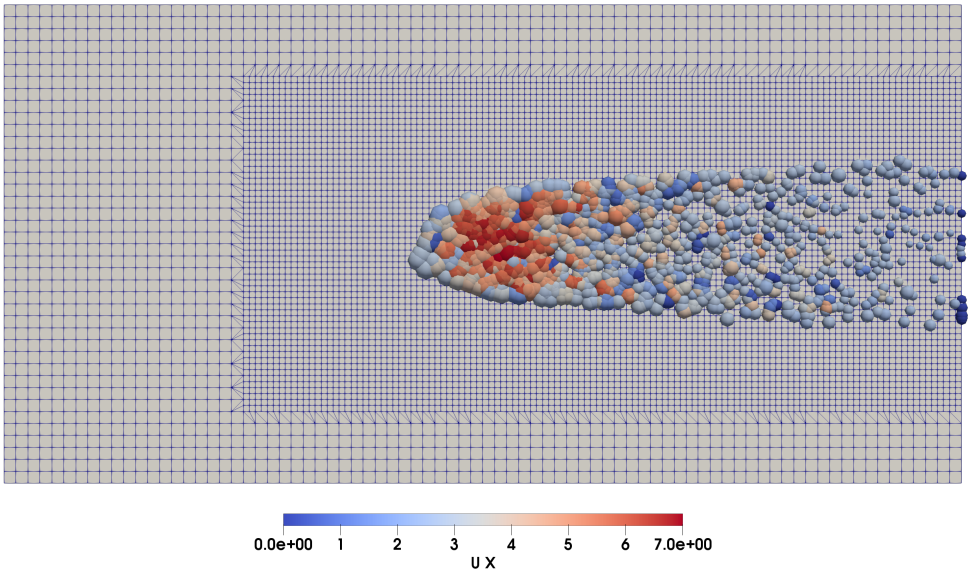



Figure 4.15: Film cloud seen from the top at $t = 0,003s$,, parcels are coloured by velocity in X direction and scaled by diameter

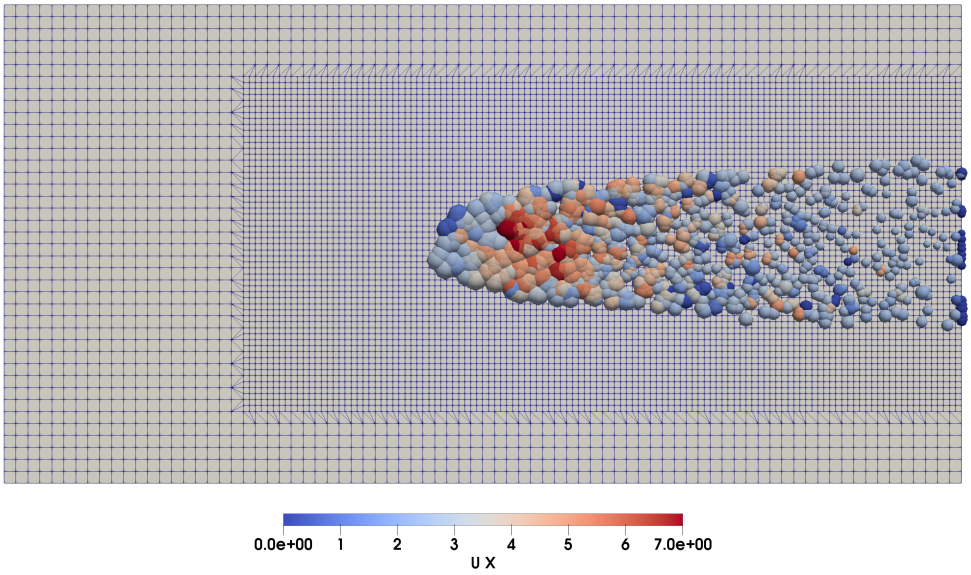



Figure 4.16: Film cloud seen from the top at $t = 0,004s$,, parcels are coloured by velocity in X direction and scaled by diameter

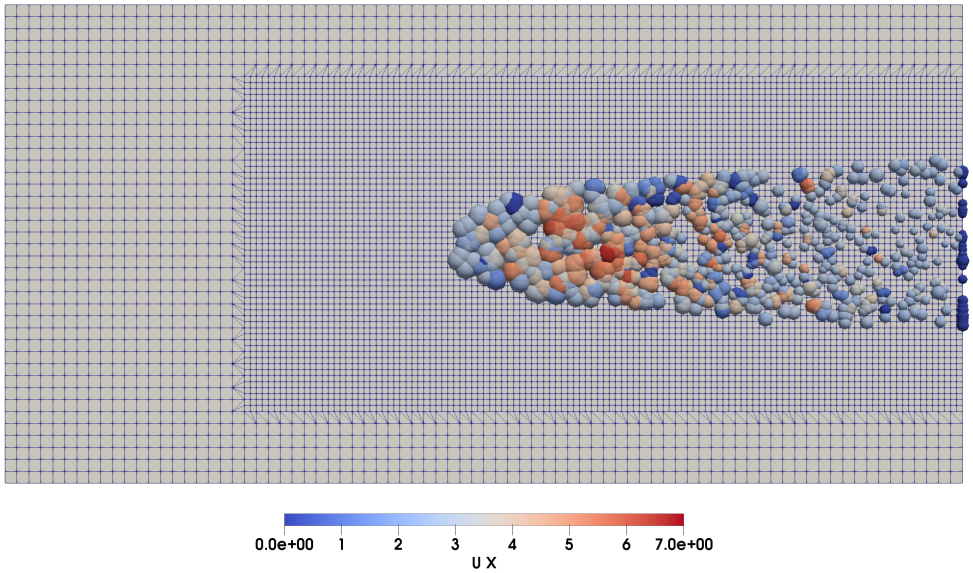



Figure 4.17: Film cloud seen from the top at $t = 0,005s$,, parcels are coloured by velocity in X direction and scaled by diameter

In Figure 4.18, the film thickness at various points along the centreline is shown. In the referenced paper, the measurements were performed under steady-flow conditions. The following figures compare the experimental results with the numerical predictions.

The numerical values were obtained by integrating the film thickness, ($\delta$), over an area of ($1, \mathrm{mm}^2$) centered at the measurement location and dividing by the same area, thus yielding a local area-averaged thickness. Furthermore, the numerical data points were temporally averaged using a running-average window of ($0.001, \mathrm{s}$), estimated by dividing the distance between measurement points (approximately (3mm)) by a representative film velocity (approximately (3m/s)). The use of a running average was motivated by the inherently unsteady nature of the simulation, for which reporting a single mean value would not adequately represent the local film dynamics. Instead, the running average provides a temporally resolved but smoothed representation of the thickness evolution while filtering high-frequency fluctuations. All time histories are reported starting from $t = 0s$, corresponding to the start of injection, and ending when the film thickness at that location decreases and the film effectively passes the probe position.

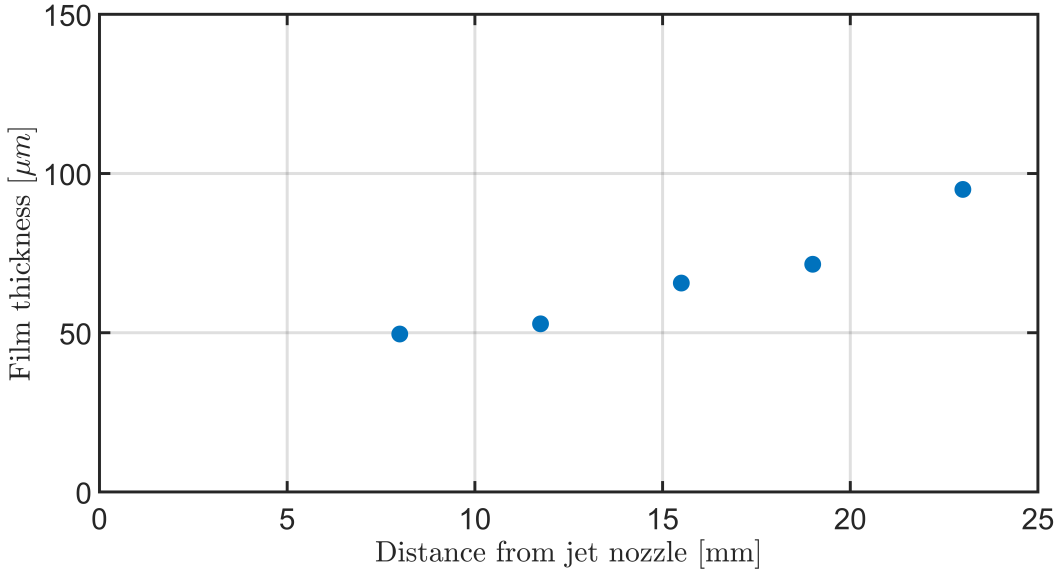



Figure 4.18: Experimental results, film thickness along the centreline.

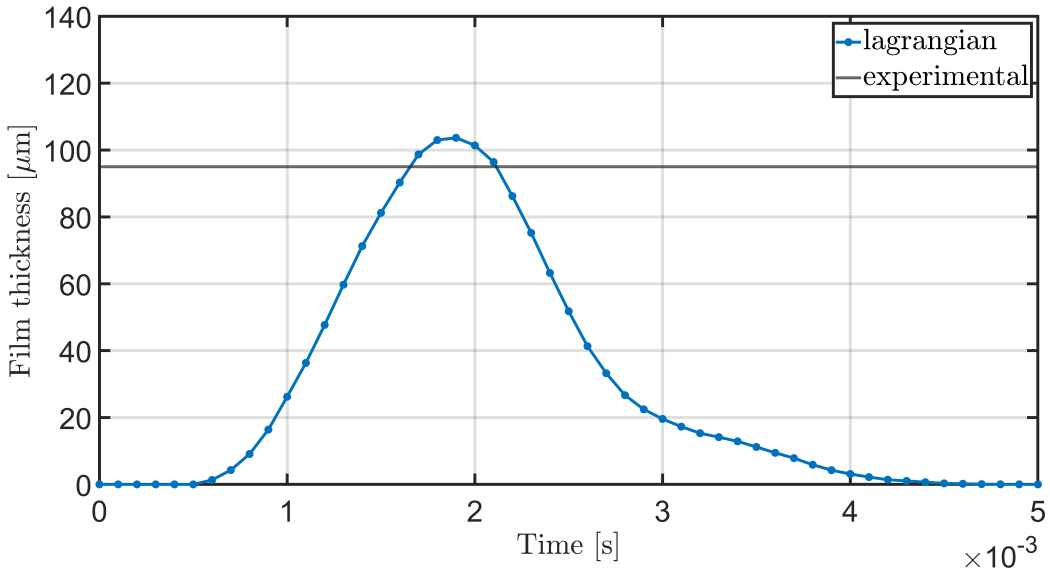



Figure 4.19: Comparison of the lagrangian model and the experimental result at $8mm$ from the jet nozzle.

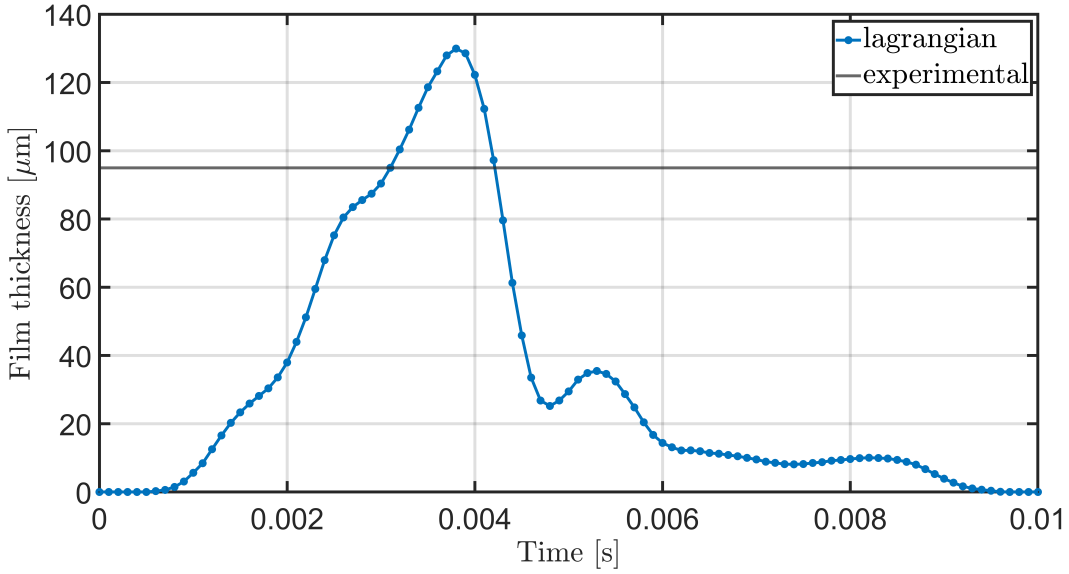



Figure 4.20: Comparison of the lagrangian model and the experimental result at $11,75mm$ from the jet nozzle.

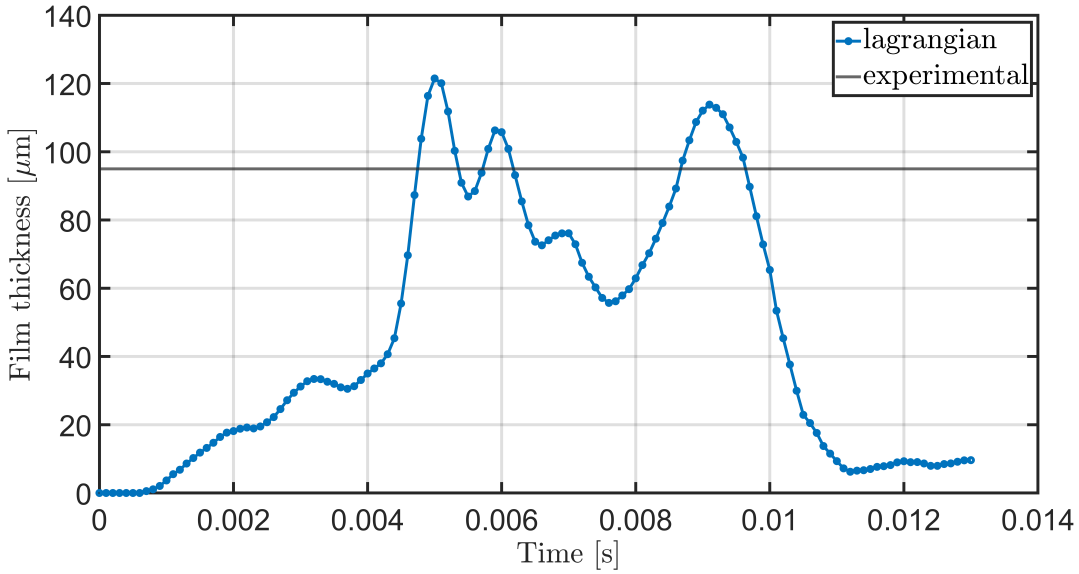



Figure 4.21: Comparison of the lagrangian model and the experimental result at $15,5mm$ from the jet nozzle.

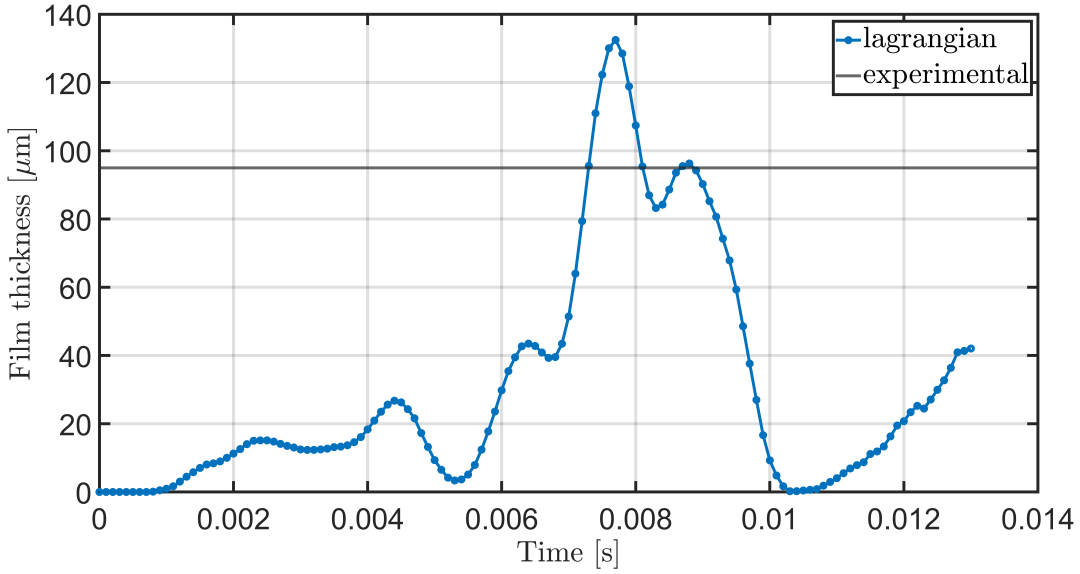



Figure 4.22: Comparison of the lagrangian model and the experimental result at 19$mm$ from the jet nozzle.

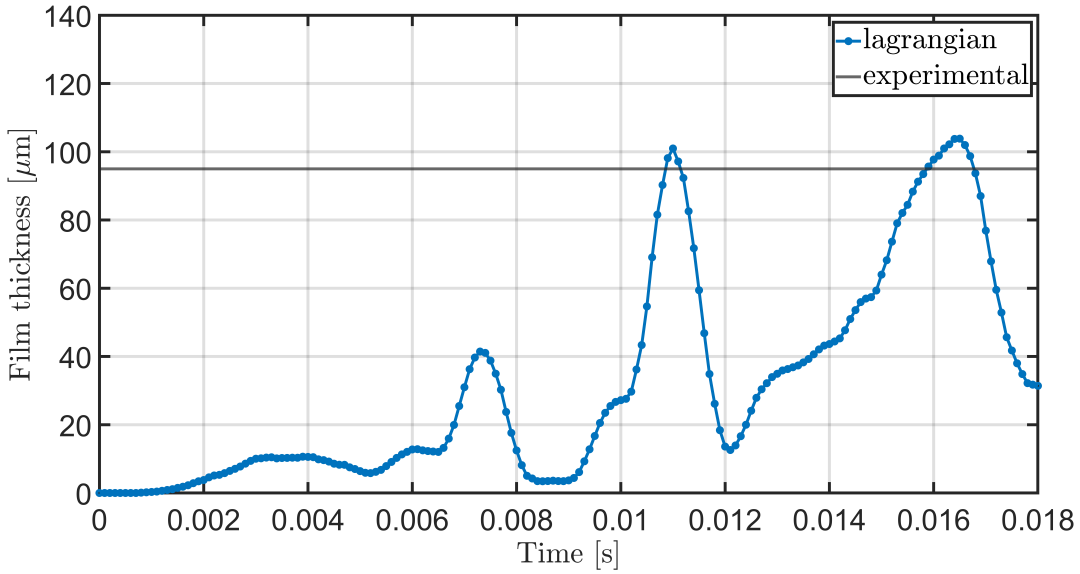



Figure 4.23: Comparison of the lagrangian model and the experimental result at 23mm from the jet nozzle.

Finally, in Figure 4.24, the computational times of the two models are compared. As expected, the Lagrangian wall-film model is faster, with a reduction in computational time of approximately (33.5%) compared to the eulerian approach.

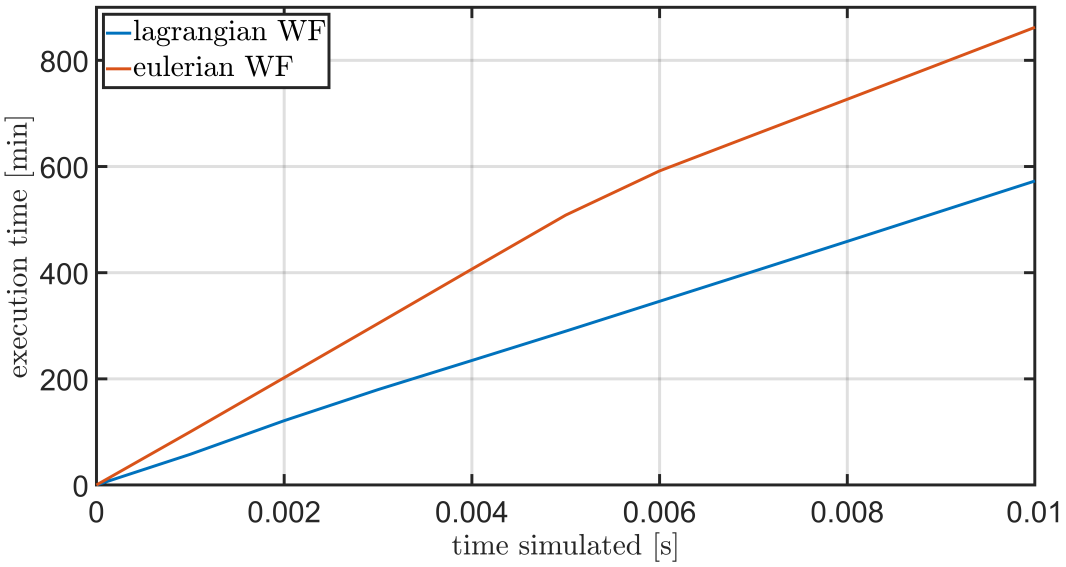



Figure 4.24: Computational times comparison.

# 5 | Conclusions and future developments

The motivation behind this work was the need for a fast and simple wall film model capable of simulating spray in Selective Catalytic Reduction (SCR) systems. SCR environments are characterized by high temperatures, high turbulence intensity, and high exhaust gas velocities. Accurately capturing wall film dynamics in such regimes is essential for predicting urea-water solution behaviour, deposition, evaporation, and overall system efficiency.

The development began with a detailed analysis of the wall film model already available in *OpenFOAM*. The existing implementation follows a lagrangian–eulerian formulation, where the spray is treated in a Lagrangian framework while the wall film is described as an eulerian thin film region coupled through an interface handling mass, momentum, and energy exchanges. Although physically comprehensive, this approach presents two major drawbacks:

1. High setup complexity, requiring careful configuration of the film region.
2. Limited mesh robustness, especially when using snappyHexMesh, which is not a boundary based meshing software by construction and may lead to inaccuracies in the thin-film region definition.

These limitations motivated the exploration of a fully Lagrangian wall film formulation as already adopted in established CFD solvers such as KIVA, CONVERGE, and ANSYS Fluent.

The core idea was to treat the wall film as a set of Lagrangian parcels constrained to move along the boundary surface. To achieve this, several key modelling components were developed:

- A new patch interaction model, initially designed to retain parcels on the wall after impact, mimicking the physical effect of capillarity and preventing rebound or detachment. The model was subsequently extended into a more complete formu-

lation capable of determining the impact outcome among deposition into the film, rebound or splash based on the local thermodynamic and Kinematic conditions of the droplet–wall interaction.

- An update film function designed to distribute the momentum inside the film parcels of the same face.
- A drag model, accounting for shear stress contributions from both the gas phase and the wall.
- A heat transfer model, incorporating conductive heat exchange with the solid wall, enabling the simulation of temperature evolution and thermally driven effects.

This formulation completely removes the need for an interface between a lagrangian spray cloud and an eulerian wall film region. Initially, sub-model activation was managed through conditional logic based on particle regime indices within the same cloud. However, this structure was later redesigned to improve clarity, modularity, and computational efficiency.

A dedicated filmCloud class was introduced, along with a cloud transfer function, allowing parcels to transition from a spray cloud to a film cloud. This architectural separation provided several advantages:

- Clear physical distinction between spray and film sub-models for the user.
- Removal of internal conditional branching based on particle indices.
- Possibility to specialize templates for each cloud type, resulting in improved computational performance.

The capabilities of the model were shown through a dedicated test case. Its response to external forces, such as gravity, was demonstrated, through correct motion of film parcels. The conductive heat transfer model showed the expected thermal behaviour. Furthermore, encouraging results were obtained from comparison with an established experimental benchmark case.

At the current stage, the model does not include the effect of the hydrostatic pressure gradient arising from spatial variations in film thickness between adjacent surface regions. Incorporating this effect in a physically consistent manner is particularly challenging within a discrete parcel framework constrained to a boundary surface. However, for SCR applications, this contribution is generally secondary compared to shear-driven transport and thermal effects.

Future developments should focus on:

- Implementation of a curvature-induced separation (stripping) model to account for film breakup in the presence of a geometric discontinuity.
- Implementation of a more advanced capillarity formulation to better represent surface tension driven spreading.
- Exploration of Smoothed Particle Hydrodynamics (SPH) hybridization, where the filmCloud could be treated using a meshless approach. This would enable the evaluation of hydrostatic pressure gradients through kernel based interpolation, potentially overcoming current limitations related to thickness-driven flow redistribution.

In conclusion, this work demonstrates that a fully Lagrangian wall film model within *OpenFOAM* is both feasible and advantageous for SCR simulations. By eliminating the eulerian thin-film region and its interface coupling, the proposed approach improves mesh robustness, reduces user setup complexity, and provides a flexible and extensible framework for future physical model enhancements.

# Bibliography

[1] G. Agati, A. Evangelisti, S. Gabriele, F. Rispoli, P. Venturini, and D. Borello. Liquid film formation: prediction accuracy of different numerical approaches. *Journal of Physics: Conference Series*, 2385(1):012138, Dec. 2022. doi: 10.1088/1742-6596/2385/1/012138.

[2] Ansys, Inc. *ANSYS Fluent Theory Guide, release 2024 R2*, 2025. URL `https://ansyshelp.ansys.com/public/account/secured?returnurl=/Views/Secured/corp/v242/en/flu_th/flu_th.html`.

[3] M. Arienti, L. Wang, M. Corn, X. Li, M. Soteriou, T. Shedd, and M. Herrmann. Modeling wall film formation and breakup using an integrated interface-tracking/discrete-phase approach. *Journal of Engineering for Gas Turbines and Power*, 133:031501–7, 02 2013. doi: 10.1115/GT2010-23381.

[4] C. Bai and A. D. Gosman. Mathematical modelling of wall films formed by impinging sprays. *SAE Transactions*, 105:782–796, 1996. ISSN 0096736X, 25771531. URL `http://www.jstor.org/stable/44736317`.

[5] J. M. Cohen and T. J. Rosfjord. Influences on the sprays formed by high-shear fuel nozzle/s wirler assemblies. *Journal of Propulsion and Power*, 9(1):16–27, Jan. 1993. doi: 10.2514/3.51351. URL `http://dx.doi.org/10.2514/3.51351`.

[6] H. Foucart, C. Habchi, J.-F. Le-coz, and T. Baritaud. Development of a three dimensional model of wall fuel liquid film for internal combustion engines. *SAE International Journal of Engines*, 107:16, 02 1998. doi: 10.4271/980133.

[7] O. Foundation. Openfoam v8 c++ source code documentation, 2020. URL `https://cpp.openfoam.org/v8/`. Accessed: 2026-02-11.

[8] S. M. Ghiaasiaan. *Two-phase flow, boiling, and condensation*. Cambridge University Press, Cambridge, England, 2 edition, Jan. 2017.

[9] C. Greenshields. Openfoam 2.0.0: Particle tracking, Aug 2011. URL `https://openfoam.org/release/2-0-0/particle-tracking/`.

[10] J. L. Hee, K. Simmons, B. Kakimpa, and D. Hann. Computationally efficient modelling of oil jet-breakup and film formation for bearing chamber applications. In *Volume 2C: Turbomachinery*, GT2018. American Society of Mechanical Engineers, June 2018. doi: 10.1115/gt2018-76172. URL `http://dx.doi.org/10.1115/gt2018-76172`.

[11] M. Kim and K. Min. Calculation of fuel spray impingement and fuel film formation in an hsdi diesel engine. *KSME International Journal*, 16(3):376–385, Mar. 2002. doi: 10.1007/bf03185235.

[12] M. Li, M. Yu, J. Liu, and X. Sheng. Analysis of water film distribution and aerodynamic performances of high-speed train under rainfall environment. *Chinese Journal of Mechanical Engineering*, 37(1), Nov. 2024. doi: 10.1186/s10033-024-01120-7.

[13] C. Mundo, M. Sommerfeld, and C. Tropea. Droplet-wall collisions: Experimental studies of the deformation and breakup process. *International Journal of Multiphase Flow*, 21(2):151–173, 1995. doi: https://doi.org/10.1016/0301-9322(94)00069-V.

[14] C. Mundo, M. Sommerfeld, and C. Tropea. On the modelling of liquid sprays impinging on surfaces. *Atomization and Sprays*, 8(6):625–652, 1998. doi: 10.1615/atomizspr.v8.i6.20.

[15] A. Nappi. *Modelling the liquid film evolution and spray wall interaction in SCR dosing units*. PhD thesis, Politecnico di Milano, 2022.

[16] L. Nocivelli. *CFD modelling and experimental characterization of urea/water solution injection inside SCR systems of diesel engines*. PhD thesis, Politecnico di Milano, 2017.

[17] I. Nova and E. Tronconi, editors. *Urea-SCR technology for deNOx after treatment of diesel exhausts*. Springer, 1 edition, Mar. 2014.

[18] I. Owen and D. Ryley. The flow of thin liquid films around corners. *International Journal of Multiphase Flow*, 11(1):51–62, Jan. 1985. doi: 10.1016/0301-9322(85)90005-9.

[19] P. O'Rourke and A. A. Amsden. A particle numerical model for wall film dynamics in port-injected engines. In *SAE Technical Paper Series*, FFL. SAE International, Oct. 1996. doi: 10.4271/961961.

[20] P. J. O'Rourke and A. A. Amsden. A spray/wall interaction submodel for the kiva-3 wall film model. In *SAE Technical Paper Series*, ANNUAL. SAE International, Mar. 2000. doi: 10.4271/2000-01-0271.

[21] M. Rein. Phenomena of liquid drop impact on solid and liquid surfaces. *Fluid Dynamics Research*, 12(2):61–93, Aug. 1993. doi: 10.1016/0169-5983(93)90106-k.

[22] T. Shedd, M. Corn, J. Cohen, M. Arienti, and M. Soteriou. Liquid film formation by an impinging jet in a high-velocity air stream. In *47th AIAA Aerospace Sciences Meeting*. American Institute of Aeronautics and Astronautics, Jan. 2009. doi: 10.2514/6.2009-998.

[23] D. W. Stanton and C. J. Rutland. Modeling fuel film formation and wall interaction in diesel engines. *SAE Technical Paper Series*, 1996. doi: 10.4271/960628.

[24] C. D. Stow and M. G. Hadfield. An experimental investigation of fluid flow resulting from the impact of a water drop with an unyielding dry surface. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 373(1755):419–441, 1981. URL `http://www.jstor.org/stable/2397009`.

[25] H. Versteeg and W. Malalasekera. *An Introduction to computational fluid dynamics: The finite volume method*. Pearson Education (US), 2024.

[26] P.-K. Wu, K. A. Kirkendall, R. P. Fuller, and A. S. Nejad. Breakup processes of liquid jets in subsonic crossflows. *Journal of Propulsion and Power*, 13(1):64–73, Jan. 1997. ISSN 1533-3876. doi: 10.2514/2.5151. URL `http://dx.doi.org/10.2514/2.5151`.

[27] A. L. Yarin and D. A. Weiss. Impact of drops on solid surfaces: self-similar capillary waves, and splashing as a new type of kinematic discontinuity. *Journal of Fluid Mechanics*, 283:141–173, Jan. 1995. doi: 10.1017/s0022112095002266.

[28] L. Zhao, R. Torelli, X. Zhu, R. Scarcelli, S. Som, H. Schmidt, J. Naber, and S.-Y. Lee. An experimental and numerical study of diesel spray impingement on a flat plate. *SAE International Journal of Fuels and Lubricants*, 10(2):407–422, Mar. 2017. doi: 10.4271/2017-01-0854.

# List of Figures

# List of Tables

# List of Symbols

| Variable | Description | SI unit |
|---|---|---|
| $t$ | time | $s$ |
| $\vec{x}$ | position | $m$ |
| $\phi$ | generic quantity | - |
| $\rho$ | density | $kg/m^3$ |
| $V$ | volume | $m^3$ |
| $M$ | mass | $kg$ |
| $U$ | velocity | $m/s$ |
| $p$ | pressure | $Pa$ |
| $T$ | temperature | $K$ |
| $Re$ | Reynolds number | - |
| $Co$ | CFL number | - |
| $\alpha$ | volume fraction | - |
| $\delta$ | film thickness | $m$ |

# Ringraziamenti

Vorrei ringraziare anzitutto il professor Gianluca Montenegro per i tantissimi insegnamenti ricevuti e per avermi seguito costantemente durante questo lungo percorso che è stata la tesi. Desidero estendere questo grazie a tutti coloro che negli anni hanno dedicato del tempo ad insegnarmi qualcosa: è grazie a voi se oggi posso compiere questo passo. In particolare vorrei ricordare il prof. Rigoli e la prof.ssa Corsi che durante i miei anni di liceo sono stati fonte di esempio costante.

Non avrei potuto intraprendere questo percorso senza mia mamma, che mi incoraggia costantemente a diventare una versione migliore di me stesso. Vorrei esprimere la mia più immensa gratitudine anche a Stefy per capirmi sempre ed essere la persona migliore che io conosca. Inoltre ringrazio tutto il resto della mia famiglia, in particolare mia sorella Enrica e mia nonna Carolina.

Vorrei ringraziare inoltre tutti i miei amici, a partire dalla "Congrega dei CM3": se questi ultimi sono stati i migliori anni della mia esperienza universitaria è soprattutto merito vostro. In particolare vorrei menzionare Elisa (detta Strange) e Luca (detto il Doc) per essere stati al mio fianco durante tutto questo percorso.

Tra gli altri amici vorrei menzionare Sara e Andrea per le ore (e le sventure) passate in treno e Alessandra per ascoltarmi sempre quando ne ho bisogno.