DEGREE PROJECT IN PHARMACEUTICAL FORMULATION - KLGM16

CFD Simulation and Experimental Validation of Air-Assisted Atomization in a Büchi Spray Dryer

Author: Serge Didier Atetelim
Supervisor: Jesús Enrique Campos Pacheco, Andreas Håkansson
Examiner: Anna Fureby
Department: Process and Life Science Engineering
Faculty of Engineering LTH
Date: May 2026

Abstract

This degree project combines computational fluid dynamics (CFD) simulations with experimental validation using a Büchi B-290 spray dryer to study particle deposition patterns during spray drying of a glycerol-based solution. The CFD simulations were performed using ANSYS Fluent with a monophasic air model, where the liquid phase was represented as discrete inert particles injected via the Discrete Phase Model (DPM). Particle deposition on heated walls was modeled using trap boundary conditions with erosion/accretion activation. Experimental validation was conducted using a solution containing glycerol, trehalose dihydrate, and colorant, processed under controlled conditions (inlet temperature 138°C, outlet temperature 80°C). The effects of particle number, particle size, mesh refinement, and aspirator rate on deposition patterns using the computational model were systematically investigated. Quantitative analysis using spectrophotometry revealed that the wall adjacent to the outlet captured the highest colorant concentration (10.70 ± 0.05 % at 50% aspirator). Temperature analysis of 10,000 particles showed that escaped particles experienced a mean temperature increase of 9 K, while trapped particles reached a maximum temperature of 380 K. A comparison with the high-fidelity 3D Large-Eddy Simulations of the same spray dryer by Pralits et al. [10] revealed that the primary deposition zones identified in our experiments (walls adjacent to outlet and connection sections) are consistent with their findings. However, our 2D planar RANS model could not capture the inherent three-dimensionality and unsteadiness of the turbulent flow. Increasing particle size from 1 μm to 6 μm revealed a critical transition at 5 μm, where the conical section began collecting more particles than the outlet tube, indicating an inertia-dominated deposition regime for larger particles. The study model provides a framework for understanding deposition mechanisms in spray drying processes.

Preface

This degree project was carried out at the Department of Process and Life Science Engineering, Lund University, between January and May 2026, as part of the Pharmaceutical Formulation program (KLGM16). The project was motivated by the need for reliable simulation tools to optimize spray drying processes used extensively in the food and pharmaceutical industries.

I would like to express my sincere gratitude to my supervisors, Jesús Enrique Campos Pacheco and Andreas Håkansson, for their invaluable guidance, patience, and expertise throughout this project. I also thank Anna Fureby for accepting the role of examiner and for her valuable feedback. I am particularly grateful to all three for their generosity in offering support and advice beyond scheduled meeting hours whenever it was needed.

Finally, I acknowledge the access to computational resources provided by Lund University

Table of Contents

1. Introduction

Spray drying is a critical unit operation in the food and pharmaceutical industries, used to convert liquid products into dry powders while preserving quality attributes such as flavor, color, and nutritional value [1]. The Büchi B-290 spray dryer is a laboratory-scale device widely used for research and development due to its flexibility and precise control of operating parameters [2]. It operates in co-current mode, where the drying air and the atomized liquid move in the same direction, ensuring gentle drying of heat-sensitive materials.

Understanding the fluid dynamics and particle behavior within spray dryers is essential for optimizing product quality and energy efficiency. Computational Fluid Dynamics (CFD) has emerged as a powerful tool for simulating these complex processes, enabling virtual experimentation and parametric studies [3]. Recent advances in CFD modeling have highlighted the importance of unsteady simulations to capture flow dynamics accurately. Pralits et al. [10] performed high-resolution Large-Eddy Simulations (LES) of a Büchi B-290 spray dryer, demonstrating that strong recirculation within the chamber significantly affects particle residence time and wall deposition. Their work, validated with experiments using a calcium carbonate suspension, showed that well-resolved simulations can predict product loss due to wall contamination with good accuracy.

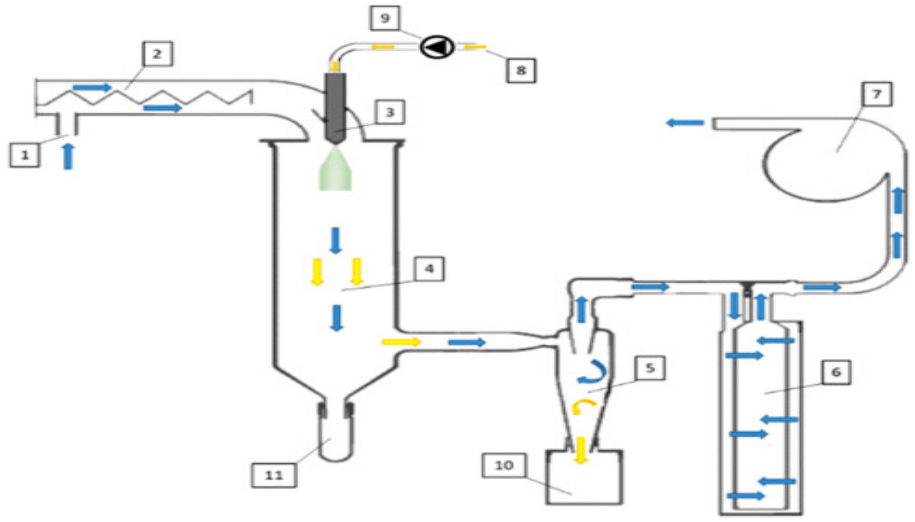

Fig. 1. Scheme of the mini spray dryer (BÜCHI- Model B-290) used for the experimental tests. In the figure, the paths of: gas stream (blue arrows) and product (yellow arrows) are reported. Numbers in the figure indicate, respectively: (1) gas inlet; (2) electric heater; (3) two fluid nozzle; (4) drying vessel; (5) cyclone to separate particles from gas stream; (6) outlet filter; (7) aspirator to pump gas through system; (8) feed solution inlet; (9) peristaltic pump; (10) right product collection vessel; (11) left product collection vessel. Adapted from BÜCHI mini spray dryer B-290 Operation manual.

The accurate prediction of particle deposition on spray dryer walls remains a challenge due to the complex interactions between the gas phase, droplets, and heated walls. Traditional

empirical approaches,such as correlations based on limited experimental data or simplified mass balance models, often fail to capture the underlying fluid dynamics and turbulence phenomena, leading to suboptimal process design. The primary aim of this project is therefore to develop and evaluate a 2D model of the Büchi spray dryer for predicting particle deposition patterns, using a glycerol-based solution with colorant for experimental validation.While a 2D model is not as accurate as a full 3D model, it offers a computationally efficient alternative for parametric studies and trend prediction. The results are intended to identify main deposition zones and relative effects of operating parameters, rather than to provide absolute quantitative predictions.

2. Theoretical Background

2.1 Governing Equations

The CFD simulations solve the Reynolds-Averaged Navier-Stokes (RANS) equations for the continuous air phase [4, 5]. These equations represent the conservation of mass, momentum, and energy in the fluid flow. The continuity equation expresses the conservation of mass, stating that the rate of mass accumulation in a control volume equals the net mass flow into it. For an incompressible flow, this reduces to the condition that the divergence of the velocity field is zero, meaning that the fluid is neither created nor destroyed within the domain. All symbols used in equations are defined in the Nomenclature (Section 2.4).

$$\frac{\partial \rho}{\partial t} + \nabla \bullet (\rho \bar{u}) = 0 \qquad (1)$$

The momentum equation, derived from Newton's second law, describes how the velocity field evolves under the influence of pressure gradients, viscous stresses, and body forces such as gravity. The left-hand side represents the rate of change of momentum of a fluid particle, while the right-hand side includes the pressure forces and viscous shear stresses. In turbulent flows, the instantaneous velocity is decomposed into a mean component and a fluctuating component, leading to the appearance of Reynolds stresses that must be modelled [4]:

$$\frac{\partial}{\partial t}(\rho \bar{u}) + \nabla.(\rho \bar{u}.\bar{u}) = -\nabla p + \nabla.\bar{\bar{\tau}} + \rho \bar{g} \qquad (2)$$

The energy equation governs the conservation of thermal energy in the flow. It accounts for heat transfer by conduction (through the thermal conductivity of the fluid), by convection (as the fluid moves), and by viscous dissipation (conversion of kinetic energy into heat). The source term represents additional heat sources or sinks, such as chemical reactions or radiative heat transfer [5]. In the present study, the energy equation is essential for capturing the heating of particles as they travel through the hot drying air.

$$\frac{\partial}{\partial t}(\rho E) + \nabla.(u(\rho E + p)) = \nabla.\left(k_{eff} \nabla T\right) + S_h \qquad (3)$$

2.2 Turbulence Modeling

The Realizable k-ε turbulence model was selected for this study due to its robustness for free-shear flows and boundary layers under adverse pressure gradients [6]. This model solves two transport equations: one for the turbulent kinetic energy (k), which represents the energy contained in the turbulent eddies, and one for the turbulent dissipation rate (ε), which represents the rate at which this turbulent energy is converted into heat. The term "realizable" refers to the mathematical constraints imposed on the model to ensure that the predicted Reynolds stresses remain physically realistic [6].

While Pralits et al. [10] demonstrated that Large-Eddy Simulation (LES) provides more detailed insight into unsteady flow features, the RANS approach remains widely used in industrial applications due to its significantly lower computational cost. RANS simulations solve only for the mean flow quantities, with the effects of turbulence modeled rather than explicitly resolved [5]. This approach is acceptable when the primary interest is in mean flow properties rather than instantaneous fluctuations. For the present study, which focuses on mean particle deposition patterns rather than detailed turbulent structures, the RANS approach is appropriate.

2.3 Discrete Phase Model (DPM)

Particle trajectories were computed using Lagrangian particle tracking [5], where individual particles are tracked through the flow field by integrating the equation of motion for each particle. This approach is fundamentally different from the Eulerian description of the continuous fluid phase, which solves for field variables at fixed points in space. In the Lagrangian framework, each particle is treated as a discrete entity whose position and velocity evolve in time under the influence of aerodynamic drag, gravity, and other forces [5].

Following the approach of Pralits et al. [10], particles are coupled with the continuous phase through momentum and heat transfer. In the present study, one-way coupling is assumed: the particles are small enough and dilute enough that they do not significantly affect the flow field, but they do exchange heat with the surrounding air. The drag force on each particle is modeled using the standard spherical drag law, which is appropriate for the near-spherical particles produced by the two-fluid nozzle [5]. Turbulent dispersion is accounted for using the discrete random walk model, which simulates the effect of turbulent eddies by adding random velocity fluctuations to the particle motion. Heat transfer between the particle and the surrounding air is modeled using the Ranz-Marshall correlation, which relates the Nusselt number (Nu) to the Reynolds (Re_p) and Prandtl numbers (Pr) [5].

$$Nu = 2 + 0.6 \cdot Re_p^{(1/2)} \cdot Pr^{(1/3)} \quad (4)$$

Where $Nu = (h \cdot d_p) / k_{air}$ represents the dimensionless heat transfer coefficient, $Re_p = (\rho \cdot |u - u_p| \cdot d_p) / \mu$ characterizes the relative motion between the particle and the gas, and $Pr = (c_p \cdot \mu) / k_{eff}$ accounts for the physical properties of the drying air. This correlation is widely used for spherical particles in a forced convection flow and is suitable for the range of particle sizes and velocities encountered in this study

2.4 Nomenclature

Symbol	Description	Units

ρ	Density	kg/m³
t	Time	s
$\overline{u}$	Velocity vector of air	m/s
u_p	Particles velocity	m/s
p	Pressure	Pa
$\overline{\overline{\tau}}$	Stress tensor	Pa
$\overline{g}$	Gravitational acceleration	m/s²
E	Total energy	J/kg
k_{eff}	Effective thermal conductivity	W/m·K
T	Temperature	K
S_h	Turbulent kinetic energy	kg. m/ s³
ε	Turbulent dissipation rate	m²/s³
Re_p	Particles Reynolds Number	dimensionless
Pr	Prandtl number	dimensionless
h	Convective heat transfer coefficient	W/m².K
c_p	Specific heat capacity of air	J/kg.K
d_p	Particles diameter	m
μ	Dynamic viscosity of the air	Pa.s

3. Materials and Methods

3.1 Laboratory Experimental Setup

3.1.1 The Büchi B-290 Spray Dryer

Fig.2. Photo of Büchi B-290 Mini Spray Dryer

The Büchi B-290 Mini Spray Dryer [2] consists of several key functional components that together enable precise control of the drying process. These include a heating system capable of raising the drying air to inlet temperatures up to 220 °C, a two-fluid nozzle that atomizes the liquid feed into fine droplets using compressed air, a cylindrical glass drying chamber where evaporation occurs, a cyclone separator that separates dried particles from the exhaust air, and a collection vessel for the final powder product. Pralits et al. [10] provided a detailed description of this apparatus in their CFD study, noting that the chamber has a height of approximately 0.6 m and a diameter of approximately 0.15 m. The instrument operates in co-current mode, meaning the drying air and atomized particles flow in the same direction, which is particularly suitable for heat-sensitive materials.

3.1.2 Solution Preparation

The solution for spray drying was prepared according to the following protocol. First, 4 mL of pure glycerol was measured. Glycerol serves as a humectant and plasticizer,its high boiling point (563 K) which prevents rapid evaporation make it an excellent model compound that influences droplet formation and deposition. Second, 80 µL of colorant was added using a micropipette; this colorant allowed for visual identification of deposition zones and quantitative analysis via spectrophotometry. Third, 0.78 g of trehalose dihydrate (a common excipient and stabilizer used in formulation development) was weighed and added to the mixture. Finally, distilled water was added to reach a total volume of 20 mL, and the mixture was stirred thoroughly until a homogeneous solution was obtained. The solution was kept at room temperature prior to spray drying

3.1.3 Experimental Conditions

The spray dryer was operated under controlled conditions. The feed flow rate was maintained at 2.5 mL/min. The inlet temperature was set to 138°C, and the outlet temperature was maintained at 80°C by adjusting the feed rate. The aspirator rate, which controls the air flow through the system, was varied among 100%, 75%, and 50% successively corresponding to

38.5 m^3/h, 31 m^3/h and 20 m^3/h to investigate its effect on particle deposition. The pump rate was fixed at 20% (5 ml/min), and the flow height was set to either 45 mm or 60 mm. The velocities at the four inlets (Figure 10) were set as follows: inlets 1 and 4 (slow inlets) at 0.1 m/s, and inlets 2 and 3 (fast inlets) at 611.067 m/s. This high velocity for the fast inlets represents the atomization air jet at the two-fluid nozzle exit, as derived from the nozzle geometry and the compressed air supply conditions used in the experimental setup. Although this value approaches sonic conditions, it reflects the highly localised nature of the nozzle jet, which decelerates rapidly upon entering the larger drying chamber. The use of an incompressible RANS solver in these near-sonic nozzle regions is acknowledged as a modelling limitation and is discussed in Section 4.5.

3.1.4 Definition of Deposition Zone

Before analyzing the deposition patterns, the spray dryer model was divided into four distinct zones (Figure 2). These zones correspond to the main structural regions of the drying chamber where particle deposition was visually observed.

- **Zone A**: The right vertical wall, extending from the inlet to the conical section. This region is directly exposed to the incoming air jets and is the first surface that particles encounter.

- **Zone B**: The left vertical wall on the site opposite to the outlet. This region is less directly exposed to the inlet jets but can receive particles carried by recirculating flows.

- **Zone C**: The cylindrical outlet tube section. This narrow tube is the exit path for particles that escape the chamber.

- **Zone D**: The conic down part, the angled conical wall connecting the main chamber to the outlet tube. This region experiences complex flow patterns due to the change in cross-section area.

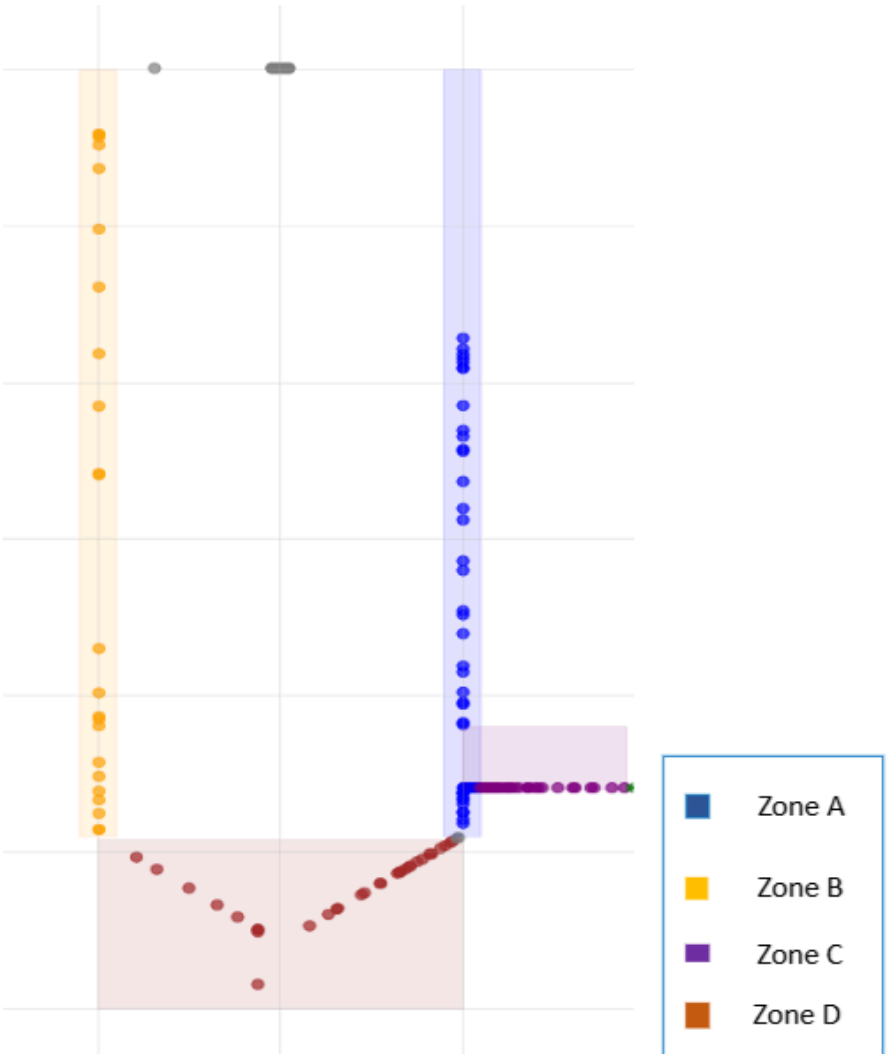

Fig 3: Schematic of the four deposition zones (A, B, C, D) on the clean spray dryer model before any experiment. Zone A: top wall. Zone B: bottom wall. Zone C: outlet tube. Zone D: conical section.

3.1.5 Quantitative Analysis by Spectrophotometry

Model Images: Before and After Spray Drying

To visualize the deposition patterns, photographs of the model were taken before and after the spray drying experiment figure 4 shows the clean model before any experiment, with no particle deposition on the walls. The four zones (A, B, C, D) are clearly visible.

Fig 4: Photograph of the clean spray dryer model before the experiment. No particle deposition is visible on the walls.

Figures 4, 5, 6, and 7 show the model after spray drying under different operating conditions (100%, 75%, and 50% aspirator rates, as well as different flow heights). The deposited colorant appears as visible stains on the walls, particularly in Zones A and C, confirming the visual observations reported in Section 3.1.

Fig 5: Photograph of the model after spray drying at 100% aspirator, flow height 60 mm. Deposition is visible in all zones, but with a different distribution compared to the 45 mm flow height case.

Fig 6: Photograph of the model after spray drying at 100% aspirator Flow height 45 mm Visible deposition is observed primarily in zone (right wall) and zone C (outlet tube)

Fig 7: Photograph of the model after spray drying at 50% aspirator, flow height 45 mm. Significant deposition is observed in Zone A (right wall), with visible accumulation in zones B, C, and D as well

Fig 8 : Photograph of the model after spray drying at 75% aspirator, flow height 45 mm. Deposition is visible in zone A (right wall) and in zone B, with some colorant accumulation in zone C

Spectrophotometry Protocol

After each spray drying run, each zone (A, B, C, D) was washed separately with the same volume of ethanol solution (10 mL per zone) to recover the deposited colorant. The ethanol solution containing the colorant from each zone was then analyzed using a spectrophotometer to measure the absorbance at the wavelength (450 nm), as determined from the glycerol calibration curve (Figure 9). The measured absorbance was converted to colorant concentration using the calibration curve, and the percentage of trapped colorant in each zone was calculated as the ratio of colorant mass in the zone to the total colorant mass injected. The areas of each zone were measured from the model geometry and are reported in Table 1.

Table1: Surface areas of each deposition zone

Zone	Area (m^2)
A	0.153
B	0.154
C	0.013
D	0.025

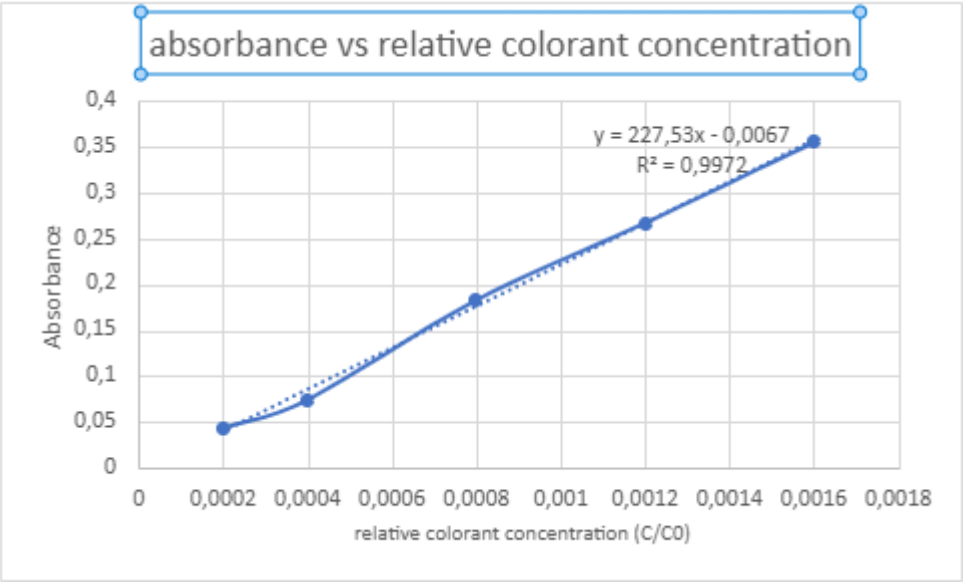

Fig 9: Glycerol standard calibration curve showing the linear relationship between concentration and absorbance at 405 nm

3.2 CFD Simulation Setup

3.2.1 Geometry and Mesh

The computational domain represents a 2D planar section of the Büchi spray dryer chamber. The geometry includes a central liquid injection point located at x = -0.26 m and y = 0, four air inlets (1 and 4 have a diameter of 1.625 cm; 2 and 3 have a diameter of 0.0085 cm), an outlet located at the right side of the chamber, and heated walls. Figure 10 shows the computational geometry with the injection point and air inlets indicated.

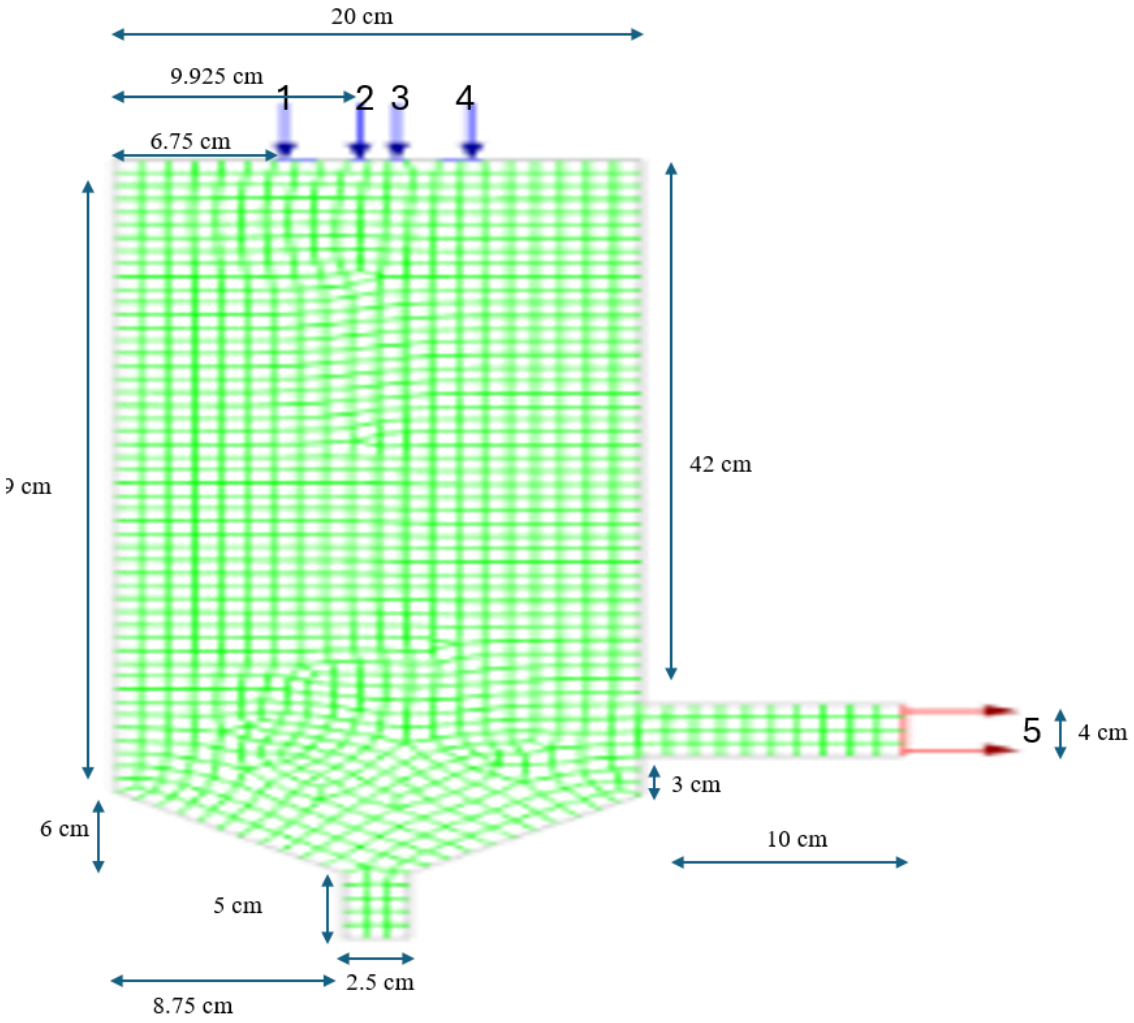

Fig 10: 2D original mesh planar domain representing a section of the Büchi B-290 spray dryer chamber showing the four air inlets and outlet.

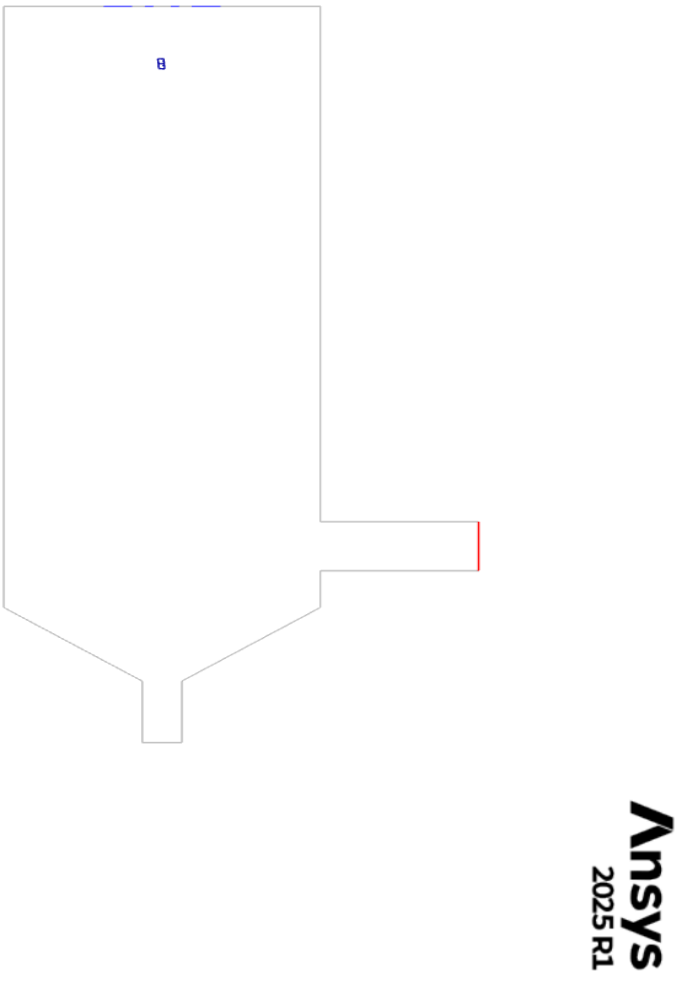

Fig 11: Computational geometry showing the central injection point (x = -0.26 m, y = 0)

A mesh independence study was performed with multiple refinement levels. The medium mesh of 109,636 cells was selected for all simulations. It was observed that excessive mesh refinement beyond this level (e.g., 429,076 cells) resulted in an unphysical disappearance of particle trapping. This is likely because the excessively fine cells near the wall artificially increased the local shear velocity, effectively 'sweeping' particles along the wall instead of allowing them to deposit. Such numerical artifacts prevent the particle-wall interaction required for the trap boundary condition to function properly. Furthermore, at very high resolutions (approaching 1.8 million cells), the flow behavior near Zone B (bottom wall) changed completely, contradicting experimental observations. This phenomenon may be attributed to the so-called flip-flop jet, a self-excited oscillatory phenomenon in which a confined jet periodically alternates between two directions in the absence of any moving parts

[16]. In highly resolved simulations of confined jets, this unsteady switching can appear spontaneously as a numerical artifact. Such behavior is not physically observed in the actual spray dryer under steady operating conditions. Therefore, a moderate refinement level was selected for the final simulations, balancing accuracy with physical realism and avoiding numerical artifacts.

3.2.2 Solver Settings and Boundary Conditions

The simulations employed a pressure-based, steady-state solver with second-order upwind spatial discretization and the SIMPLE scheme for pressure-velocity coupling [5]. The Realizable k-ε turbulence model [6] was used with enhanced wall treatment to resolve the boundary layer near the walls. The energy equation was enabled to account for heat transfer [5].

For the air inlets, the velocities and temperatures were set as follows: inlets 1 and 4 (slow inlets) at 0.1 m/s and 381 K, and inlets 2 and 3 (fast inlets) at 611.067 m/s and 293 K. The turbulence intensity was set to 5% for all inlets. The fast inlets (inlets 2 and 3) had a turbulent length scale of 8.5×10^{-5} m, while the slow inlets (inlets 1 and 4) had a turbulent length scale of 0.01625 m.

The walls were modeled with a thermal convection boundary condition using a heat transfer coefficient of $\alpha = 0.1$ W/m²·K and an ambient temperature of T = 298 K. This low value of the heat transfer coefficient reflects the insulating properties of the glass drying chamber, which minimizes heat loss to the surroundings. While the adiabatic condition would be ideal, the convection boundary condition provides numerical stability and a conservative estimate of heat transfer. The DPM trap condition was applied to all walls, meaning that any particle contacting the wall is considered deposited and removed from the simulation [5].

3.332.3 Particle Tracking Configuration

Multiple simulations were performed with different numbers of particles: 50, 100, 150, 200, 250, and finally 10,000 particles to ensure statistical convergence. The particles were injected at x = -0.26 m, y = 0 to simulate the atomization point at the exit of the two-fluid nozzle. This position corresponds to the location where the liquid feed is dispersed into fine droplets, approximately 0.04 m downstream of the inlet plane (x = -0.30 m), based on the geometry of the Büchi B-290 nozzle assembly as shown in Figure 11. The initial particle velocity was set to 0 m/s. The particle diameter was set to 1×10^{-6} m (1 μm) for the baseline simulations and varied from 1 μm to 6 μm for the parametric study. The particle density was set to 998 kg/m³ (water), and the specific heat capacity was set to 4180 J/kg·K.

3.2.4 Mesh Refinement Study

A systematic mesh refinement study was conducted to ensure that the numerical solution was independent of the mesh resolution. The refinement was performed in multiple stages, starting from a baseline mesh of approximately 6,500 cells. The final stable mesh for reliable results was 109,636 cells. As noted in Section 3.2.1, excessive refinement led to unphysical behavior, which guided the selection of this moderate refinement level for all final simulations.

4. Results

4.1 Experimental Results

4.1.1 Effect of Aspirator Rate on Deposition

The aspirator rate, which controls the air flow through the system, had a significant effect on particle deposition. The quantitative spectrophotometry analysis provided precise measurements of colorant deposition across the four zones. Table 2 presents the results for 100% aspirator rate at a flow height of 45 mm. Zone A exhibited an absorbance of 0.064, corresponding to a colorant volume of 3.11 µL and representing 3.89% of the total colorant. Zone B and Zone C both showed an absorbance of 0.037, with colorant volumes of 1.92 µL each, representing 2.40% of the total trapped colorant. Zone D showed an absorbance of 0.030, with a colorant volume of 1.61 µL, representing 2.01% of the total trapped colorant. When normalized by zone area, Zone C exhibited the highest percentage per area (191), reflecting its small surface area.

Table 2: Quantitative analysis at 100% aspirator, flow height 45 mm

Zone	**Absorbance**	**Volume colorant(µl)**	**area(m^2)**	**Percentage of trapped colorant (%)**	**Surface coverage (µl/m^2)**
A	0.064 ± 0.001	3.11 ± 0.04	0.153	3.89 ± 0.05	25.42 ± 0.32
B	0.037 ± 0.001	1.92 ± 0.03	0.154	2.40 ± 0.04	15.58 ± 0.26
C	0.037 ± 0.002	1.92 ± 0.07	0.013	2.40 ± 0.09	184.62 ± 6.92
D	0.030 ± 0.001	1.61 ± 0.03	0.025	2.01 ± 0.04	80.40 ± 1.60
Total	-	8.57 ± 0.04	-	10.70 ± 0.05	-

When the aspirator rate was reduced to 75% (Table 3), deposition increased across zones A and B but reduced across zones C and D. Zone A showed an absorbance of 0.081, with a colorant volume of 3.85 µL, representing 4.81% of the total trapped colorant. Zone B showed an absorbance of 0.062, with a colorant volume of 3.02 µL, representing 3.78%. Zone C showed an absorbance of 0.004, with a colorant volume of 0.47 µL, representing 0.59%. Zone D showed an absorbance of 0.012, with a colorant volume of 0.82 µL, representing 1.03%.

Table 3: Quantitative analysis at 75% aspirator, flow height 45 mm

Zone	Absorbance	Volume colorant(µl)	area(m^2)	Percentage of trapped colorant (%)	Surface coverage (µl/m^2)
A	0.081 ± 0.001	3.85 ± 0.03	0.153	4.81 ± 0.03	31.44 ± 0.2
B	0.062 ± 0.002	3.02 ± 0.08	0.154	3.78 ± 0.10	24.55 ± 0.65
C	0.004 ± 0.001	0.47 ± 0.04	0.013	0.59 ± 0.05	45.38 ± 3.85
D	0.012 ± 0.001	0.82 ± 0.02	0.025	1.02 ± 0.04	40.80 ± 1.6
Total	-	8.16 ± 0.04	-	10.20 ± 0.05	-

When the aspirator rate was further reduced to 50% (Table 4), deposition increased dramatically. Zone A showed an absorbance of 0.188, corresponding to a colorant volume of 8.56 µL, representing 10.70% of the total trapped colorant. Zone B showed an absorbance of 0.112, with a colorant volume of 5.22 µL, representing 6.52%. Zone C showed an absorbance of 0.012, with a colorant volume of 0.82 µL, representing 1.03%. Zone D showed an absorbance of 0.024, with a colorant volume of 1.35 µL, representing 1.69%.

Table 4: Quantitative analysis at 50% aspirator, flow height 45 mm

Zone	Absorbance	Volume colorant(µl)	area(m^2)	Percentage of trapped colorant (%)	Surface coverage (µl/m^2)
A	0.188 ± 0.01	8.56 ± 0.04	0.153	10.70 ± 0.05	69.93 ± 0.32
B	0.112 ± 0.01	5.22 ± 0.02	0.154	6.52 ± 0.04	42.34 ± 0.25
C	0.012 ± 0.01	0.82 ± 0.01	0.013	1.03 ± 0.01	79.23 ± 0.77
D	0.024 ± 0.01	1.35 ± 0.05	0.025	1.69 ± 0.06	67.60 ± 2.4
Total	-	15.95 ± 0.03	-	19.94 ± 0.04	-

The total trapped colorant increased from 10.70% at 100% aspirator to 19.93% at 50% aspirator, representing an 86% increase. This confirms that reducing the aspirator rate significantly increases particle deposition, consistent with the increased residence time hypothesis.

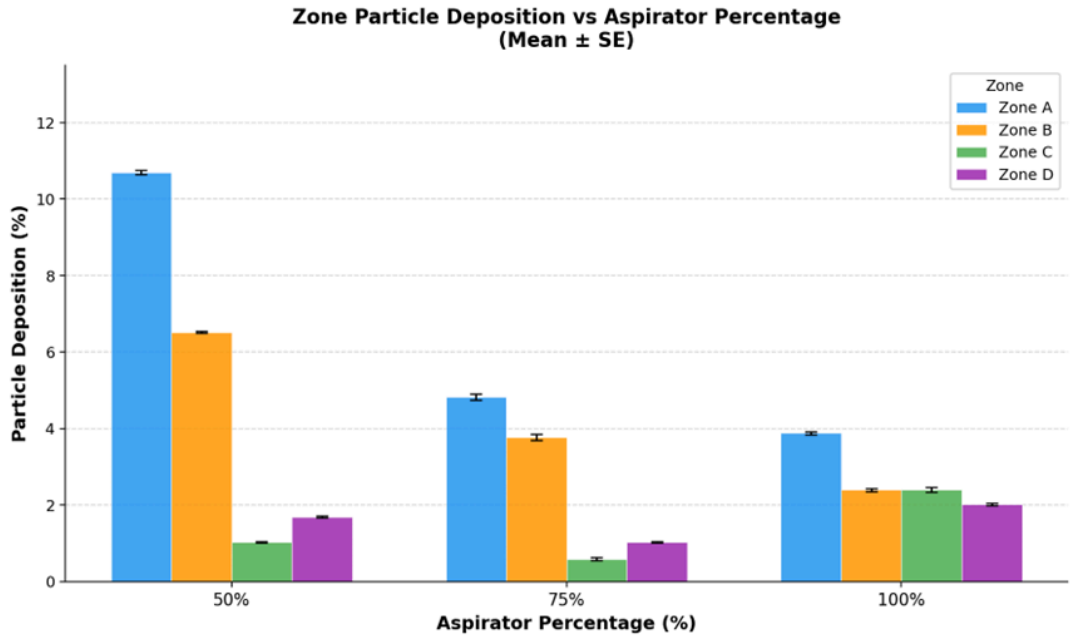

Fig 12: Mean (± SE) particle deposition (%) per zone at each aspirator percentage.

Statistical Analysis

To determine whether the observed differences in total colorant deposition across aspirator rates were statistically significant, a one-way analysis of variance (ANOVA) was performed. The assumptions of ANOVA (independence of observations, normality, and homogeneity of variance) were considered. The analysis was conducted using three replicates per aspirator condition.

Table5: Summary of colorant concentration trapped per zone under different aspirator rates (flow height 45 mm)

Aspirator rate (%)	Total trapped colorant (%)	Standard error (±)
100	10.70	0.05
75	10.20	0.05
50	19.94	0.04

The ANOVA results revealed a highly significant effect of aspirator rate on total colorant deposition. The F-statistic was 13,712.35 with a p-value less than 0.0001, indicating that the differences between the three aspirator conditions are extremely unlikely to have occurred by random chance alone.

To identify which specific aspirator rates differed from each other, a post-hoc Tukey Honestly Significant Difference (HSD) test was conducted at a significance level of $\alpha = 0.05$. The Tukey HSD threshold was calculated to be 0.118%. The results of the pairwise comparisons are summarized in Table 6 and illustrated in Figure 13.

Table 6: Tukey HSD pairwise comparisons for aspirator rates

Comparison	Mean difference (%)	Significant (p < 0.05)?	Tukey grouping
50% vs 75%	9.72	Yes	a vs b
50% vs 100%	9.23	Yes	a vs b
75% vs 100%	0.50	Yes	b vs c

The Tukey HSD test revealed that all three pairwise comparisons were statistically significant, despite the similar means between the 75% and 100% conditions (10.21% vs 10.70%). This indicates that even a relatively small difference of 0.50% between the 75% and 100% aspirator rates is statistically significant given the low variability within each group.

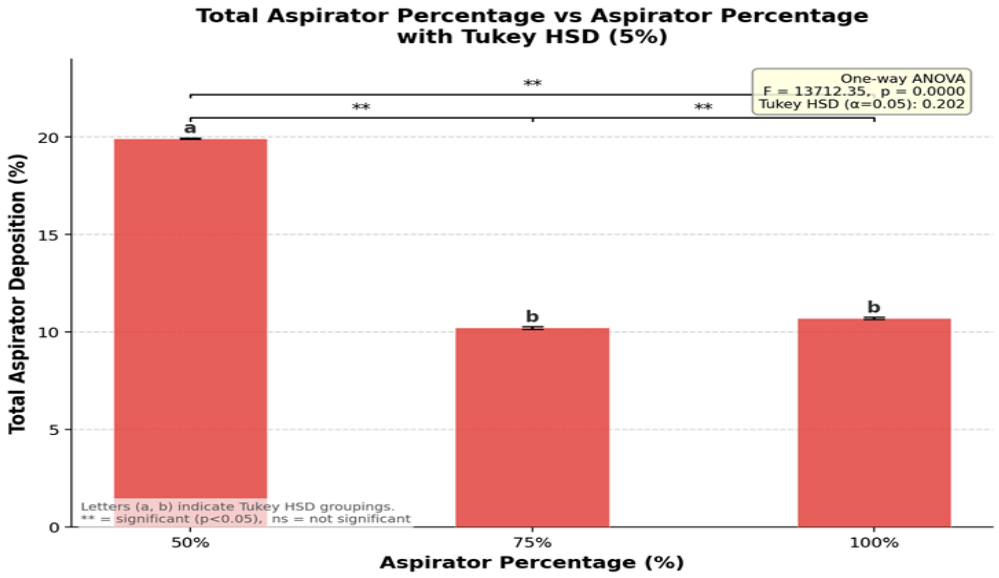

Fig 13: Total aspirator deposition (%) across the three settings with one-way ANOVA results and Tukey HSD (α = 0.05) pairwise comparisons. Shared letters indicate no significant difference. Error bars represent standard error. F-statistic = 13712.35, p < 0.0001

The statistical analysis thus confirms that the observed differences in particle deposition are not merely due to random variation but reflect a genuine physical effect. The aspirator rate has a clear and significant influence on how much colorant is deposited on the walls of the spray dryer.

4.1.2 Summary of Experimental Results

The experimental results consistently show that Zone A (top wall) is the primary deposition zone under all operating conditions, capturing between 3.9% and 10.7% of the total colorant depending on the aspirator rate. Zone C (outlet tube) and Zone D (conical section) capture significant portions of the remaining colorant. The total trapped colorant increases from 10.7% at 100% aspirator to 19.9% at 50% aspirator, demonstrating that lower air flow rates increase particle residence time and deposition.

4.2 CFD Simulation Results

4.2.1 Flow Field Characteristics

The velocity contours (Figure 13 at the left) reveal the development of the air jets from the four inlets. The fast inlets (inlets 2 and 3) produce jets with velocities approaching 611.067 m/s, which rapidly decelerate as they mix with the slower air from inlets 1 and 4. The flow field is characterized by strong recirculation zones near the walls, particularly in the region downstream of the inlets. These recirculation zones are the primary mechanism responsible for particle trapping, as they can entrain particles and carry them toward the walls. Figure 12 at the right shows the velocity vectors in the same domain, providing a clearer picture of the recirculation patterns. The vectors reveal that the flow separates at the inlet plane and forms a large recirculation bubble that occupies a significant portion of the chamber.

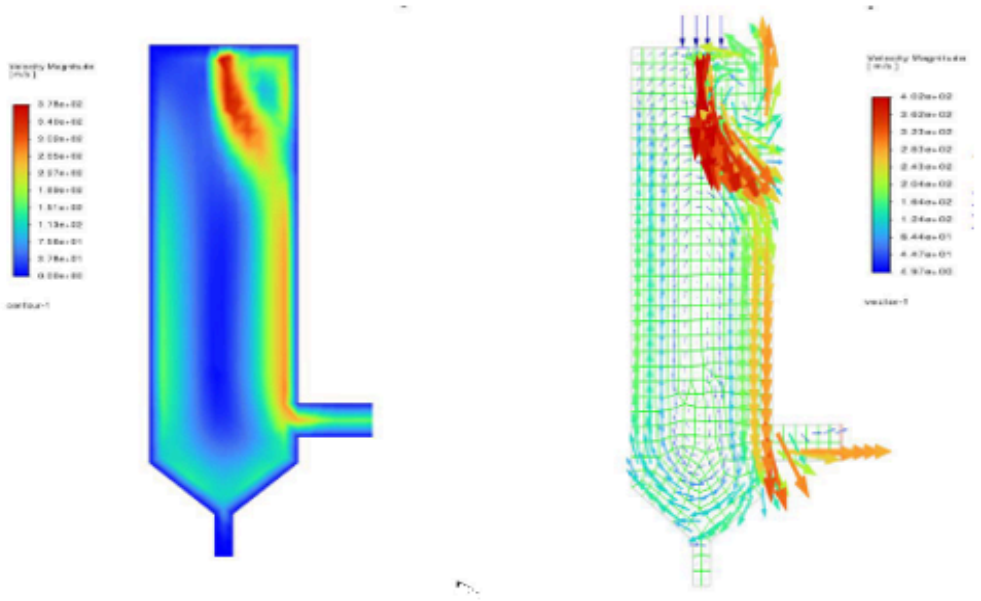

Fig 14: Velocity magnitude contour at the left and velocity vector showing direction and magnitude of the air flow from CFD simulation at the right

4.2.2 Residence Time Distribution

The residence time distribution (Figure 15) reveals a clear bimodal separation between escaped and trapped particles. Escaped particles exit the domain within a very narrow window of approximately 0.0008 to 0.005 seconds, forming a sharp peak. In contrast, trapped particles exhibit residence times spanning a wider range, from approximately 0.005 to 0.08 seconds, indicating complex recirculation paths before deposition.

While the bimodal separation is clear, it is worth noting that a small number of escaped particles (the "stragglers" visible in the green bars of Figure 15) also exhibit longer residence times, indicating that some particles that eventually exit still experience brief recirculation before escaping.

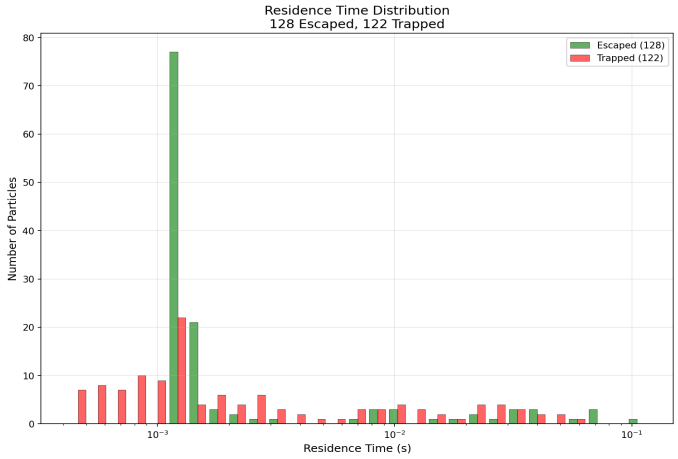

Fig15: Histogram of residence times for escaped (green) and trapped (red) particles. Escaped particles form a sharp peak at short residence times (0.0008–0.005 s), while trapped particles show a broader distribution across longer times (0.005–0.08 s).

4.2.3 Temperature Analysis

The temperature analysis was performed for 250 particles injected. Figure 16 shows the temperature evolution of all particles over time. The data reveals two distinct populations with very different behaviors.

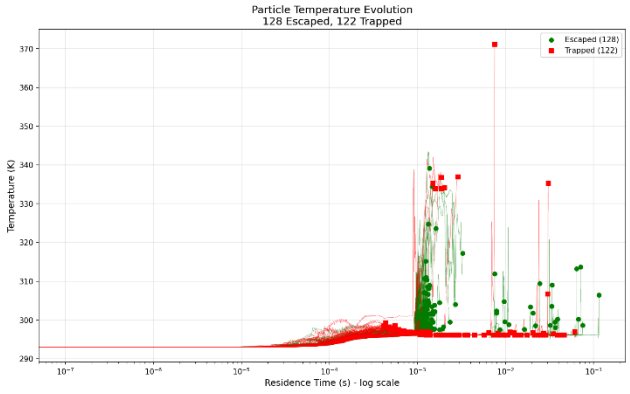

Fig 16: Temperature versus residence time for 250 particles. The green line represents escaped particles (128 particles, 51.2%), which maintain a constant temperature of approximately 298 K. The red line represents trapped particles (122 particles, 48.8%), whose temperature increases with residence time, reaching up to approximately 370 K.

The temperature analysis reveals two distinct particles' populations with fundamentally different thermal histories, as shown in Figure 16. For escaped particles, represented by the green line, the average temperature remains constant at approximately 298 K regardless of residence time. Out of the 250 particles injected, 128 particles (representing 51.2 %) escaped the domain. Their temperature remains constant because they exit the heated chamber quickly before significant heat transfer from the hot air can occur. The maximum temperature observed among escaped particles was approximately 342 K.

Of the 250 particles injected, 122 particles (representing 48.8 %) were trapped on the walls. The temperature of these trapped particles increases from approximately 298 K at short residence times up to approximately 370 K at longer residence times. This corresponds to a maximum temperature increase of up to 77 K above the escaped particle temperature, considering an injection temperature of 293 K. The positive correlation between residence

time and final temperature confirms that particles which remain longer in the drying chamber absorb more heat from the hot air, leading to progressively higher temperatures.

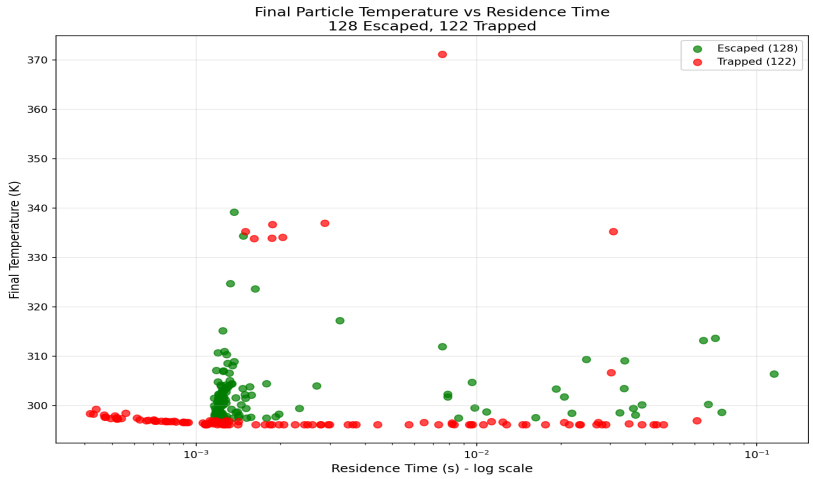

Fig 17: Scatter plot of final temperature versus residence time for all 250 particles. Escaped particles (green) cluster at low residence times with final temperatures around 298 K. Trapped particles (red) show a wider distribution, with final temperatures increasing with residence time up to approximately 342 K.

The scatter plot (Figure 17) confirms this bimodal distribution. Escaped particles form a tight cluster at short residence times (approximately 0.0008 to 0.005 seconds) with final temperatures around 342 K. Trapped particles are distributed across a wider range of residence times (0.005 to 0.08 seconds) with final temperatures ranging from approximately 298 K to 370 K.

Escaped particles show a sharp peak at approximately 298 K, showing that they exit the chamber before significant heating occurs. Trapped particles show a broader distribution, with a peak at lower temperatures (approximately 330-340 K) and a tail extending to higher temperatures (up to 370 K). This suggests that trapped particles experience a range of heating histories depending on their path and residence time.

4.2.4 Effect of Particle Size on Trapping

The effect of particle size on deposition patterns was investigated by varying the particle diameter from 1 µm to 6 µm while keeping the number of injected particles constant at 250. The results reveal a striking trend: as the particle size increased, the trapping rate increased dramatically.

For small particles (1 µm to 4 µm), Zones A and C consistently captured more particles than Zone D. At 1 µm, Zone A captured 49 particles while Zone D captured only 22. At 2 µm, Zone A captured 61 particles and Zone C captured 27, while Zone D captured 24. At 3.2 µm, Zone A captured 59 particles and Zone C captured 38, while Zone D captured 32.

A clear transition occurred at 5 µm. At this particle size, Zone C captured 27 particles while Zone D captured 28, marking the crossover point where the conical section begins to match

the outlet tube in deposition. At 6 µm, this reversal was fully established, with Zone D capturing 30 particles against only 18 in Zone C.

This transition suggests that larger particles (≥5 µm) have sufficient inertia to penetrate deeper into the conical section (Zone D) before being carried upward or escaping, whereas smaller particles are more easily entrained in the recirculating flows that deposit them on the upper walls (Zone A) or in the outlet tube (Zone C). This finding is consistent with Pralits et al. [10], who reported a Sauter mean diameter of 3.2 ± 0.25 µm for particles collected after spray drying.

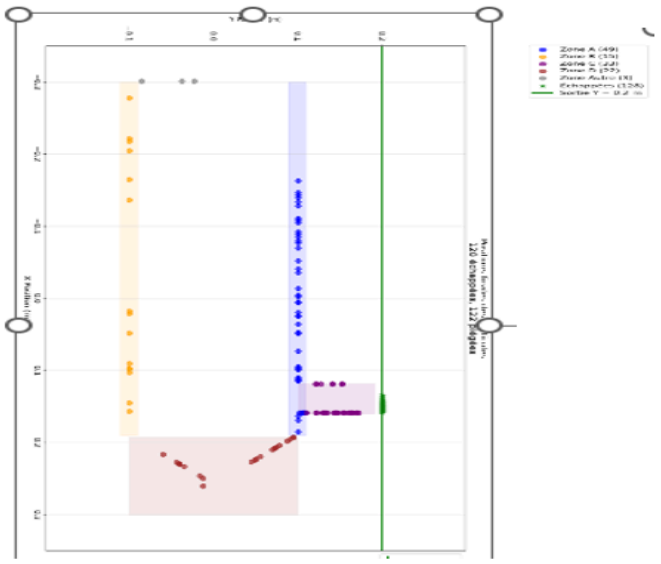

Fig 18: Particle deposition pattern for 250 injected particles. Zone A captured 49 particles, Zone B captured 15, Zone C captured 33, Zone D captured 22, and 3 particles were in other zones.

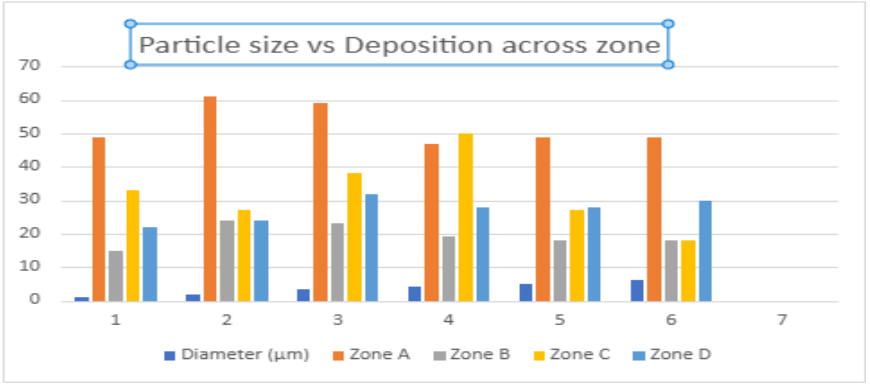

Fig 19: Bar chart showing the effect of particle size on x-axis deposition across zones A, B, C, and D for 250 particles. A clear transition is visible at 5 µm where Zone D begins to collect more particles than Zone C.

4.2.3 Temperature Analysis (10,000 particles)

The following temperature analysis describes the fate of solid particles without evaporation. In a real spray drying process, evaporative cooling would significantly reduce particle temperatures. Therefore, the temperatures reported here represent an upper bound and should be interpreted with caution.

To ensure statistical convergence and to capture the full range of thermal histories, a simulation was performed with 10,000 particles. Of these, 4,956 particles (49.6%) escaped the domain, while 5,044 particles (50.4%) were trapped on the walls. The deposition percentages were: Zone A captured 1,966 particles (approximately 20% of the total), Zone B captured 627 particles (approximately 6%), Zone C captured 1,030 particles (approximately 10%), and Zone D captured 595 particles (approximately 6%).

The temperature statistics for the 10,000-particle simulation are summarized in Table 7. For escaped particles, the initial mean temperature was 293 K, and the final mean temperature was 302.2 K, representing a mean increase of 9.2 K. The minimum temperature recorded among escaped particles was 296 K, while the maximum reached 352 K, with a standard deviation of 7 K.

For trapped particles, the initial mean temperature was also 293 K, but the final mean temperature was only 296 K, representing a mean increase of only 3 K. This finding, while initially counterintuitive, can be explained by the deposition mechanisms: trapped particles may impact and adhere to the walls early in their trajectory before they have been significantly heated by the hot air. The minimum temperature among trapped particles remained at the injection temperature of 293 K, while the maximum reached 380 K, a dramatic increase of 87 K. The standard deviation of 8 K indicates a wider distribution of thermal histories for trapped particles.

Table 7: Temperature statistics for escaped and trapped particles (10,000 particles)

Parameter	Escaped Particles	Trapped Particles
Number of particles	4,956 (49.56%)	5,044 (50.44%)
Initial mean temperature	293 K	293 K
Final mean temperature	302 K	296 K
Mean temperature increase	9 K	3 K
Minimum temperature	296 K	293 K
Maximum temperature	352 K	380 K
Standard deviation	7 K	8 K

4.2.4 Comparison: 250 vs 10,000 Particles

The consistency of the results between the 250-particle and 10,000-particle simulations (Table 8) confirms that the 250-particle simulation was statistically converged for mean deposition patterns. A chi-square test for independence showed no statistically significant difference between the deposition distributions of the two simulations ($p > 0.05$), indicating that 250 particles are sufficient for capturing mean deposition patterns.

Table 8: Comparison of particle statistics between 250-particle and 10,000-particle simulations

Parameter	250 particles	10,000 particles
Escaped particles	128 (51.2%)	4,956 (49.6%)
Trapped particles	122 (48.8%)	5,044 (50.4%)
Zone A (top wall)	49 (19.6%)	1,966 (19.7%)
Zone B (bottom wall)	15 (6%)	627 (6.3%)
Zone C (outlet tube)	33 (13.2%)	1,030 (10.3%)
Zone D (conical section)	22 (8.8%)	595 (6.0%)
Escaped particles max temperature	~342 K	352 K
Trapped particles max temperature	~370 K	380 K

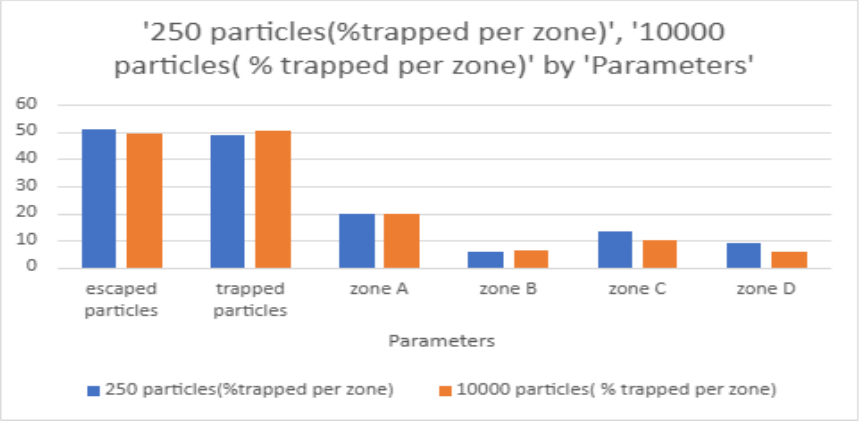

Fig 20: Bar chart comparing deposition patterns for 250 vs 10,000 particles, showing excellent agreement across all zones.

5. Discussion

5.1 Bimodal Residence Time Distribution

The clear separation between fast escaped particles and slow trapped particles observed in this study is a critical finding that illuminates the fundamental fluid dynamics of the Büchi B-290 spray dryer. Escaped particles exit the domain within a very narrow time window of approximately 0.0008 to 0.005 seconds, forming a sharp peak in the residence time distribution. In contrast, trapped particles exhibit residence times spanning a much wider range, from approximately 0.005 to 0.08 seconds, indicating complex recirculation paths before deposition. This bimodal distribution reveals that the spray dryer's flow field offers two distinct fates for injected particles. Particles that follow the main flow streamlines exit quickly, experiencing minimal heating and a low probability of wall impact. Conversely,

particles that enter recirculation zones remain in the chamber for extended periods, experiencing multiple passes through the hot drying air. These recirculating particles have a much higher probability of impacting a wall, either because they lose momentum during their extended journey or because they are carried toward the wall by the recirculating flow. The existence of such recirculation zones was confirmed by Pralits et al. [10] in their high-fidelity Large-Eddy Simulations, who noted that the strong mean recirculation in the drying chamber leads to a wide spread of the residence time of each particle. The present RANS simulations, despite their lower resolution, captured the same qualitative behavior, demonstrating that even simplified models can reveal essential features of the flow.

5.2 Particle Deposition Mechanisms

The identification of distinct trapping regions across the four zones suggests that multiple deposition mechanisms operate simultaneously within the spray dryer. In Zone A, the top wall near the inlet, particles are likely entrained in the primary recirculation zone that forms downstream of the air inlets. As these particles circulate, they gradually lose momentum and eventually settle on the top wall. The dominance of Zone A for particles of all sizes indicates that this recirculation zone is robust and captures particles regardless of their diameter. In Zone C, the outlet tube, particles appear to be carried by the flow toward the exit but deposit on the walls of this narrow passage.

The most striking deposition mechanism is observed in Zone D, the conical section, and Zone C, the outlet tube. For particles smaller than 5 μm, Zone C captured more particles than Zone D across all tested sizes. A clear transition appeared at 5 μm, marking the crossover point where the conical section begins to match, and ultimately dominate, the outlet tube in deposition. At 6 μm, this reversal was fully established, with Zone D capturing 30 particles against only 18 in Zone C. This transition reveals a critical size threshold at approximately 5 μm, above which particles have sufficient inertia to penetrate deeper into the conical section before being captured, while smaller particles are preferentially entrained by recirculating flows toward the outlet tube

5.3 Comparison with Pralits et al. (2022): 2D RANS vs 3D LES

The comparison between the present 2D RANS simulations and the Large-Eddy Simulations (LES) of Pralits et al. [10] provides valuable insights into the capabilities and limitations of each numerical approach. Their work, which used a calcium carbonate suspension and a realistic number of particles (10,000 particles per second), serves as a benchmark for understanding the limitations of our simplified 2D RANS model.

Flow characteristics. Pralits et al. [10] demonstrated that the flow in the drying vessel is fundamentally three-dimensional, unsteady, and lacks azimuthal symmetry. As they state in Section 5.1 of their paper, the azimuthal symmetry of the laminar inlet is broken almost immediately as the flow of the drying air undergoes transition to turbulence ($Re \approx 7500$), and the flow remains unsteady throughout the rest of the domain. The present RANS simulations, by contrast, assume a 2D planar geometry and steady-state flow. This simplification represents a straight channel rather than the cylindrical tube of the actual Büchi B-290 chamber. Consequently, the model imposes an artificial symmetry on the flow field and

cannot capture the curvature of the cylindrical walls, nor the three-dimensional azimuthal asymmetry that characterizes the real flow.

Deposition zones. Regarding deposition locations, Pralits et al. [10] reported that material losses in the equipment were approximately 20 ± 5% and were due to the deposition of solid particles on the wall in the upper part of the drying vessel as well as in different sections of the connection pipe and in the outlet filter [10, Section 4, Table 6]. This is qualitatively consistent with our experimental findings, where Zone A (top wall) captured 10.70% of the total colorant at 50% aspirator, and Zones C and D captured additional colorant, bringing the total trapped colorant to approximately 20% at 50% aspirator. Table 1 summarizes the key differences between the two modeling approaches.

Table 9: Comparison of deposition zones with Pralits et al. (2022)

Feature	**Present Study (2D RANS Model)**	**Pralits et al. (2022) [10] (3D LES Model)**
Dimensionality	2D axisymmetric	3D
Time dependence	Steady state	Unsteady
Turbulence model	RANS (k-ε)	LES (IDDES, WALE)
Grid resolution	~110,000 cells	~930,000 cells
Particle count	10,000 particles (total)	~10,000 particles per second
Dominant deposition	Zone A (top wall) ~20%	Upper part of drying vessel
Secondary deposition	Zones C and D	Connection pipe and outlet filter

Recirculation zones. Despite these limitations, the present RANS simulations show qualitative agreement with the LES results in terms of mean flow characteristics. Both approaches identify strong recirculation zones in the drying chamber. The RANS simulation predicts a large recirculation bubble near the top wall, extending approximately 0.07 m horizontally from $x \approx 0.05$ m to $x \approx 0.12$ m. Pralits et al. [10] report a similar recirculation feature in their LES velocity fields (see their Figure 8), though the exact dimensions are not explicitly stated. Based on visual comparison, the LES recirculation bubble appears slightly larger (≈ 0.09 m), likely due to the enhanced mixing from resolved turbulent fluctuations. This suggests that the RANS model captures the main recirculation feature but may underestimate its size.

Residence time distribution. The bimodal residence time distribution observed in the present study matches the physical mechanism described by Pralits et al. [10]: particles that follow the main flow streamlines exit quickly, while those that enter recirculation zones remain in the chamber for extended periods. This is a key finding that both modeling approaches capture.

Statistical convergence. The consistency of the deposition percentages between the 250-particle and 10,000-particle simulations demonstrates that the RANS approach, when properly configured, can provide statistically converged mean deposition patterns at a fraction of the computational cost of LES. However, the RANS approach is likely to underestimate the frequency of extreme events due to its steady-state nature and the lack of resolved turbulent fluctuations.

Conclusion from literature. Pralits et al. [10] emphasized the importance of using unsteady simulations with appropriate resolution. In their conclusion (Section 6), they state that "steady simulations, although they can still provide good agreement with experimental results over mean properties, cannot describe key flow features responsible for the large spread in particle residence time" [10]. This directly supports our observation that accurate CFD prediction of the spray drying process requires a three-dimensional, unsteady approach, but that simplified 2D RANS models can still provide useful qualitative insights and trend predictions at a much lower computational cost.

5.4 Limitations of the Study

Several limitations of this study should be acknowledged. First, the 2D axisymmetric simplification of the geometry cannot capture three-dimensional effects, particularly the azimuthal asymmetries that Pralits et al. [10] demonstrated to be significant. Second, the RANS turbulence model does not resolve the instantaneous turbulent fluctuations that affect particle dispersion; a Large-Eddy Simulation approach would provide more accurate predictions. Third, the assumption of monodisperse particles is a simplification, as real spray drying produces a distribution of particle sizes [10]. Fourth, the absence of an evaporation model means that the particles do not lose mass as they dry, which affects both their size and surface properties. Fifth, the particle count, while increased to 10,000 particles, is still lower than the 10,000 particles per second used by Pralits et al. [10] over multiple seconds. Sixth, the model does not account for the time to formation of a solid skin (i.e., the end of the constant rate period in droplet drying). In reality, a droplet remains sticky only during the initial constant-rate drying period; once a solid skin forms, the particle becomes rigid and is less likely to adhere to the wall upon impact. This critical transition is not captured in the present model, where all particles that reach the wall are assumed to be deposited (trap condition).

6. Conclusion

This degree project successfully combined experimental spray drying with computational fluid dynamics simulations to investigate particle deposition patterns in a Büchi B-290 spray dryer.

The experimental results demonstrated that Zone A, the top wall on the inlet side, captured the highest colorant concentration, reaching 10.70% at 50% aspirator rate. Lower aspirator rates resulted in higher deposition due to increased particle residence time, with total trapped colorant increasing from 10.70% at 100% aspirator to 19.93% at 50% aspirator.

CFD simulations suggest that particle size may influence trapping patterns, with a potential transition around 5 μm. However, this finding should be interpreted with caution. In the

simulations, particles larger than 5 μm tended to deposit in the conical section (Zone D) rather than the outlet tube (Zone C). This trend may reflect the increased inertia of larger particles, but the absolute deposition levels predicted in the outlet tube are higher than typically observed in experiments. This discrepancy likely arises from the absence of an evaporation model and the assumption of monodisperse particles, which may overestimate the number of particles reaching the outlet region.

The temperature analysis of 10,000 particles revealed that escaped particles experienced a mean temperature increase of 9 K, reaching a final mean temperature of 302 K, while trapped particles reached a maximum temperature of 380 K. However, the relatively low mean temperature increase of trapped particles (only 3.5 K) is surprising and likely due to the high particle velocities in the simulation (up to 611 m/s), which reduce residence time and limit heat transfer. Additionally, the absence of an evaporation model means that particles do not experience the cooling effect of evaporating water, which would further complicate the temperature evolution. Therefore, the temperature predictions should be considered as upper bounds rather than realistic estimates.

The consistency of the deposition percentages between the 250-particle and 10,000-particle simulations confirms statistical convergence for mean deposition patterns.

The comparison with the high-fidelity Large-Eddy Simulations of Pralits et al. [10] revealed that while the present 2D planar RANS model qualitatively captured the major deposition zones identified experimentally, it cannot reproduce the complex, time-dependent, asymmetric flow features that govern particle dispersion. The critical particle size threshold of approximately 5 μm identified in this study should be interpreted with caution, as it may be influenced by model simplifications (e.g., monodisperse particles, no evaporation, 2D geometry). Nevertheless, the observed trends, such as the shift from outlet tube deposition to conical section deposition with increasing particle size, are likely transferable to real spray drying processes, even if the absolute values require further validation. The model provides a useful framework for understanding deposition mechanisms and can guide optimization efforts, particularly for identifying critical size thresholds and deposition zones.

6. Future Work

Several avenues for further development are suggested based on the findings of this study and the recommendations of Pralits et al. [10]. First, a full three-dimensional model should be developed to capture the azimuthal asymmetries in the flow field. Second, the RANS turbulence model should be replaced with Large-Eddy Simulation to better capture unsteady flow features and turbulent dispersion. Third, an evaporation model should be included to account for mass transfer from the drying particles, following the two-stage evaporation model described by Mezhericher et al. [11, 12] and used by Pralits et al. [10]. Fourth, the particle size distribution should be modeled as polydisperse using a Rosin-Rammler distribution. Fifth, surface properties such as surface tension, elasticity, and protein adsorption kinetics should be incorporated into the model to capture the effects observed experimentally, particularly the critical size threshold at 5 μm. Sixth, the critical particle size threshold should be validated with additional experiments using different feed materials. Seventh, optimization studies should be conducted to determine optimal operating conditions that minimize particle deposition while maintaining adequate drying performance.

7. References

[1] Masters, K. (2002). *Spray Drying in Practice*. SprayDryConsult International ApS.

[2] Büchi Labortechnik AG. (2024). *B-290 Spray Dryer Operating Manual*.

[3] Keshani, S., Daud, W. R. W., Nourouzi, M. M., Namvar, F., & Ghasemi, M. (2015). Spray drying: An overview on wall deposition, process and modeling. *Journal of Food Engineering*, 146, 152–162.

[4] Versteeg, H. K., & Malalasekera, W. (2007). *An Introduction to Computational Fluid Dynamics: The Finite Volume Method* (2nd ed.). Pearson Education.

[5] ANSYS Inc. (2025). *ANSYS Fluent Theory Guide*. Release 2025R1.

[6] Shih, T. H., Liou, W. W., Shabbir, A., Yang, Z., & Zhu, J. (1995). A new k-ε eddy viscosity model for high Reynolds number turbulent flows. *Computers & Fluids*, 24(3), 227–238.

[7] Jaskulski, M., Wawrzyniak, P., & Zbicinski, I. (2020). CFD simulations of droplet and particle deposition in a spray dryer. *Drying Technology*, 38(1-2), 167–180.

[8] Woo, M. W., Daud, W. R. W., Mujumdar, A. S., Wu, Z. H., Talib, M. Z. M., & Tasirin, S. M. (2008). CFD simulation of temperature and air flow profiles in a pilot-scale spray dryer. *Drying Technology*, 26(5), 575–585.

[9] Tsotsas, E., & Mujumdar, A. S. (2014). *Modern Drying Technology, Volume 5: Process Intensification*. Wiley-VCH.

[10] Pralits, J. O., Atzori, M., Colli, M., Pettinato, M., Drago, E., & Perego, P. (2022). Unsteadiness and resolution effects in experimentally verified simulations of a spray drying process. *Powder Technology*, 404, 117482.

[11] Mezhericher, M., Levy, A., & Borde, I. (2008). Heat and mass transfer of single droplet/wet particle drying. *Chemical Engineering and Processing*, 47(5), 965–972.

[12] Mezhericher, M., Levy, A., & Borde, I. (2010). Spray drying modelling based on advanced droplet drying kinetics. *Chemical Engineering and Processing*, 49(11), 1205–1213.

[13] Langrish, T., & Fletcher, D. (2001). Spray drying of food ingredients and applications of CFD in spray drying. *Chemical Engineering and Processing*, 40(4), 345–354.

[14] Ziaee, A., et al. (2019). Spray drying of pharmaceuticals and biopharmaceuticals: Critical parameters and experimental process optimization approaches. *European Journal of Pharmaceutical Sciences*, 127, 300–318.

[15] Shishir, M. R. I., & Chen, W. (2017). Trends of spray drying: a critical review on drying of fruit and vegetable juices. *Trends in Food Science & Technology*, 65, 49–67.

[16] Hirata, K. et al. (2011). Empirical formula for oscillation frequency of flip-flop jet nozzle. Transactions of the Japan Society of Mechanical Engineers, Series B, 77(775), 567-576.

8. Appendices

Appendix A: Journal File for CFD Simulations

The complete journal file used for the CFD simulations is provided as a separate document. This journal file includes all commands for mesh reading, model setup, boundary condition configuration, particle injection, and post-processing.

```
/file/start-transcript transcript150_1.trn
;NB! Each time you run, you need to delete teh file "transcript.trn" before running!


; ============================================================
; 1 — CHARGER LE MAILLAGE
; ============================================================
/file/read-case "Essay2.msh"

; ============================================================
; 2 — MODELES PHYSIQUES
; ============================================================
; Energie — chaque reponse sur une ligne separee
; yes=activer, no=visc-dissip, no=press-work, no=kinetic, yes=diffusion-at-inlets
/define/models/energy? yes no no no yes

; k-epsilon realizable
/define/models/viscous/ke-realizable? yes

; ============================================================
; 3 — DENSITE AIR : incompressible-ideal-gas
```

```
; ============================================================
/define/materials/change-create air air yes incompressible-ideal-gas no no no no no no



; ============================================================
; 4 — CONDITIONS LIMITES
; ============================================================

; INLET 1 — 0.1 m/s, 381 K
; Turbulence : no=K-epsilon, yes=Intensity-Length, 5%, 0.01625 m
/define/boundary-conditions/velocity-inlet inlet1 yes yes no 0.1 no 0 no 1 no 0 no 381 no yes
5 0.01625

; INLET 2 — 611.067 m/s, 293 K
; Turbulence : no=K-epsilon, yes=Intensity-Length, 5%, 8.5e-5 m
/define/boundary-conditions/velocity-inlet inlet2 yes yes no 611.067 no 0 no 1 no 0 no 293 no
yes 5 8.5e-5

; INLET 3 — 611.067 m/s, 293 K
; Turbulence : no=K-epsilon, yes=Intensity-Length, 5%, 8.5e-5 m
/define/boundary-conditions/velocity-inlet inlet3 yes yes no 611.067 no 0 no 1 no 0 no 293 no
yes 5 8.5e-5

; INLET 4 — 0.1 m/s, 381 K
; Turbulence : no=K-epsilon, yes=Intensity-Length, 5%, 0.01625 m
/define/boundary-conditions/velocity-inlet inlet4 yes yes no 0.1 no 0 no 1 no 0 no 381 no yes
5 0.01625

; OUTLET — 353 K
/define/boundary-conditions/pressure-outlet outlet yes no 0 no 353 no yes no no yes 5 10
yes no no

; WALL — 300 K
(define alpha_wall 0.1)
(define T_wall 298)
/define/boundary-conditions/wall wall-surface_body n 0 n 0 n y convection n alpha_wall n
T_wall n n n 0 n 0.5 n 1

; ============================================================
; 5 — CRITERES DE CONVERGENCE (1e-14)
; ============================================================
/solve/monitors/residual/convergence-criteria 1e-14 1e-14 1e-14 1e-14 1e-14 1e-14

 ; ============================================================
; 12 — GRAVITE (X = 9.81 m/s2)
; ============================================================
```

```
/define/operating-conditions/gravity yes 9.81 0

; ============================================================
; 7 — INITIALISATION HYBRIDE
; ============================================================
/solve/initialize/hyb-initialization

; ============================================================
; 8 — CALCUL ECOULEMENT SEUL : 1500 iterations
; ============================================================
/solve/iterate 1500


; ============================================================
; 9 — LIGNES DE PROFIL DE VITESSE ET EXPORT
; Syntaxe confirmee en session interactive
; ============================================================

; Ligne a l'entree (X=0)
/surface/line-surface entree_line -0.2 -10 -0.2 10

; Ligne a la sortie (X=0.13)
/surface/line-surface sortie_line 0.13 -10 0.13 10

; Export profil entree
/plot/plot y velocity_entree.txt n y 0 1 n n velocity-magnitude (entree_line)

; Export profil sortie
/plot/plot y velocity_sortie.txt n y 0 1 n n velocity-magnitude (sortie_line)

; Sauvegarder image profil entree
/display/objects/create xy-plot velocity-xy-entree y-axis-function velocity-magnitude
plot-direction direction-vector x 0 y 1 q surfaces-list (entree_line) q
/display/objects/display velocity-xy-entree
/display/save-picture velocity-xy-entree.png y

; Sauvegarder image profil sortie
/display/objects/create xy-plot velocity-xy-sortie y-axis-function velocity-magnitude
plot-direction direction-vector x 0 y 1 q surfaces-list (sortie_line) q
/display/objects/display velocity-xy-sortie
/display/save-picture velocity-xy-sortie.png y

; ============================================================
; 10 — RAFFINEMENT GLOBAL
; Syntaxe confirmee en session interactive
; ============================================================
```

```
/mesh/adapt/set/maximum-refinement-level 200

/mesh/adapt/cell-registers/add region_0 type hexahedron min -10 -10 max 10 10 q q

/mesh/adapt/manual-refinement-criteria "region_0"
/mesh/adapt/adapt-mesh

/solve/iterate 500


/mesh/adapt/cell-registers/add region_1 type hexahedron min -10 -10 max 10 10 q q


/mesh/adapt/manual-refinement-criteria "region_1"
/mesh/adapt/adapt-mesh


/solve/iterate 800


/mesh/adapt/cell-registers/add region_2 type hexahedron min -10 -10 max 10 10 q q


/mesh/adapt/manual-refinement-criteria "region_2"
/mesh/adapt/adapt-mesh


/solve/iterate 800


/mesh/adapt/cell-registers/add region_3 type hexahedron min -10 -10 max 10 10 q q


/mesh/adapt/manual-refinement-criteria "region_3"
/mesh/adapt/adapt-mesh


/solve/iterate 800
/mesh/adapt/cell-registers/add region_4 type hexahedron min -10 -10 max 10 10 q q


/mesh/adapt/manual-refinement-criteria "region_4"
/mesh/adapt/adapt-mesh


/solve/iterate 800

; ============================================================
; 11 — RAFFINEMENT y+
; ============================================================
/mesh/adapt/cell-registers/add ystar type yplus-ystar min 0 max 1 q q
```

```
/solve/iterate 500
/mesh/adapt/manual-refinement-criteria "ystar"
/mesh/adapt/adapt-mesh


;============================================================
; 12 — CREER L'INJECTION
; Sequence exacte confirmee en session interactive
; ============================================================
(define Ntries 150) ;Here you choose nbr of particles released
(define x0 -0.26) ;Here you set x-coordinate of particle injection
(define y0 0) ;Here you set y-coordinate of particle injection
(define ux0 0) ;Here you set x-coordinate velocity of particle injection (for example 10 for 10 m/s downwards)
(define uy0 0) ;Here you set y-coordinate velocity of particle injection (always 0 if downward pointing injection)
(define d0 1e-6); Here you set size of drop formed after atomization
/define/models/dpm/injections/create-injection injection-0 no no no yes no Ntries 0.15 no no no x0 y0 ux0 uy0 d0 1e-06 1e-12


;/define/materials/change-create anthracite anthracite yes constant 1300 no no no no no no no
/define/materials/change-create anthracite anthracite yes constant 998 y constant 4180
; ============================================================
; 13 — CONDITION DPM PAROIS : TRAP
; ============================================================
/define/boundary-conditions/set/wall wall-surface_body () dpm-bc-type y trap ()
/display/set/particle-tracks/report-variables particle-x-position particle-y-position particle-id particle-x-velocity particle-y-velocity color-by quit
/display/set/particle-tracks/report-to file
/display/set/particle-tracks/history-filename "particles-temp.txt"
/display/particle-tracks/particle-tracks temperature "injection-0" () 0. 1e12


/display/set/particle-tracks/report-type summary
/display/set/particle-tracks/history-filename "particles-temp-sum.txt"
/display/particle-tracks/particle-tracks temperature "injection-0" () 0. 1e12
; ============================================================
; 14 — SAUVEGARDE
; ============================================================
/file/write-case-data "simulation-buchi-150-1-31pct.cas.h5"
/file/stop-transcript
```

Appendix B: Python Data Analysis Scripts

The Python scripts developed for data analysis, including residence time distribution calculation, temperature analysis, and zone classification, are provided as separate documents. These scripts utilize the pandas, numpy, and matplotlib libraries for data manipulation and visualization.

a) Code for particles trapped zone

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import os


# ==========================================
# 📁 PARAMÈTRES UTILISATEUR
# ==========================================


FILE_PATH = "H:/particles-temp-50-1.txt"


# Paramètres de détection
OUTLET_Y = 0.20          # Position Y de la sortie (m)
OUTLET_TOLERANCE = 0.01   # Tolérance pour détecter la sortie (m)


# Limites des zones de piégeage (VERSION CORRIGÉE)
ZONE_A = {
    'name': 'A (paroi supérieure)',
    'x_min': -0.30,
    'x_max': 0.19,
    'y_min': 0.09,   # Y ≈ 0.1 m avec tolérance
    'y_max': 0.11,
    'color': 'blue'
}


ZONE_B = {
    'name': 'B (paroi inférieure)',
    'x_min': -0.30,
    'x_max': 0.19,
    'y_min': -0.11,
    'y_max': -0.09,
    'color': 'orange'
}
```

```
ZONE_C = {
    'name': 'C (cylindre de sortie)',
    'x_min': 0.1186964,
    'x_max': 0.16,
    'y_min': 0.10,
    'y_max': 0.19,   # ← CORRIGÉ : Y < 0.2 pour ne pas compter les sorties
    'color': 'purple'
}


ZONE_D = {
    'name': 'D (zone conique)',
    'x_min': 0.192,   # ← CORRIGÉ
    'x_max': 0.30,    # ← CORRIGÉ
    'y_min': -0.10,
    'y_max': 0.10,
    'color': 'brown'
}


# Mode d'affichage
SAVE_PLOT = True
PLOT_PATH = "H:/positions_finales_par_zone.png"


# ============================================
# 📥 FONCTION 1 : CHARGEMENT DES DONNÉES
# ============================================


def load_particle_data(file_path):
    """Charge le fichier particles-temp.txt généré par Fluent"""

    if not os.path.exists(file_path):
        raise FileNotFoundError(f"❌ Fichier non trouvé : {file_path}")

    with open(file_path, 'r') as f:
        lines = f.readlines()

    # Trouver où commencent les données (après la ligne "-----")
    start_idx = 0
    for i, line in enumerate(lines):
        if '-----' in line:
            start_idx = i + 2
            break

    # Lire les données
    data_lines = []
    for line in lines[start_idx:]:
```

```
            if line.strip():
                values = line.strip().split('\t')
                if len(values) >= 7:
                    data_lines.append(values)

    # Créer le DataFrame
    columns = ['ResidenceTime', 'XPosition', 'YPosition', 'ParticleID',
               'XVelocity', 'YVelocity', 'ColorBy']

    df = pd.DataFrame(data_lines, columns=columns)

    # Convertir en numérique
    for col in columns:
        df[col] = pd.to_numeric(df[col], errors='coerce')

    # Nettoyer
    df = df.dropna()
    df = df[df['ParticleID'] >= 0]

    print(f"✅ Chargé : {df['ParticleID'].nunique()} particules, {len(df)} points")
    return df


# ============================================
# 📊 FONCTION 2 : DÉTECTION DES PARTICULES PAR ZONE (VERSION CORRIGÉE)
# ============================================


def get_zone(x, y):
    """
    Détermine dans quelle zone se trouve une particule piégée
    Retourne le nom de la zone ou None
    """
    # Zone A : paroi supérieure
    if (ZONE_A['x_min'] <= x <= ZONE_A['x_max'] and
        ZONE_A['y_min'] <= y <= ZONE_A['y_max']):
        return 'A'

    # Zone B : paroi inférieure
    if (ZONE_B['x_min'] <= x <= ZONE_B['x_max'] and
        ZONE_B['y_min'] <= y <= ZONE_B['y_max']):
        return 'B'

    # Zone C : cylindre de sortie (Y < 0.2)
    if (ZONE_C['x_min'] <= x <= ZONE_C['x_max'] and
        ZONE_C['y_min'] <= y <= ZONE_C['y_max']):
        return 'C'
```

```
    # Zone D : zone conique
    if (ZONE_D['x_min'] <= x <= ZONE_D['x_max'] and
        ZONE_D['y_min'] <= y <= ZONE_D['y_max']):
        return 'D'

    return None


def detect_particles_by_zone(df, outlet_y=OUTLET_Y, outlet_tol=OUTLET_TOLERANCE):
    """
    Détecte les particules :
    - Échappées : celles qui atteignent la sortie Y = outlet_y
    - Piégées : les autres, avec zone attribuée
    """

    escaped = []
    trapped_by_zone = {'A': [], 'B': [], 'C': [], 'D': []}
    other_trapped = []

    for pid, group in df.groupby('ParticleID'):
        last = group.iloc[-1]
        x_final = last['XPosition']
        y_final = last['YPosition']
        v_final = (last['XVelocity']**2 + last['YVelocity']**2)**0.5
        t_final = last['ResidenceTime']

        # Vérifier si la particule est échappée (atteint la sortie)
        if abs(y_final - outlet_y) < outlet_tol:
            escaped.append({
                'ParticleID': pid,
                'X_final': x_final,
                'Y_final': y_final,
                'V_final': v_final,
                'Time_final': t_final
            })
        else:
            # Déterminer la zone de piégeage
            zone = get_zone(x_final, y_final)
            if zone and zone in trapped_by_zone:
                trapped_by_zone[zone].append({
                    'ParticleID': pid,
                    'X_final': x_final,
                    'Y_final': y_final,
                    'V_final': v_final,
                    'Time_final': t_final,
                    'Zone': zone
                })
```

```
            else:
                other_trapped.append({
                    'ParticleID': pid,
                    'X_final': x_final,
                    'Y_final': y_final,
                    'V_final': v_final,
                    'Time_final': t_final,
                    'Zone': 'Autre'
                })

    if other_trapped:
        trapped_by_zone['Autre'] = other_trapped

    return {
        'escaped': escaped,
        'trapped_by_zone': trapped_by_zone,
        'escaped_count': len(escaped),
        'trapped_count': sum(len(v) for v in trapped_by_zone.values()),
        'total_count': df['ParticleID'].nunique()
    }


# ============================================
# 📊 FONCTION 3 : AFFICHAGE DES RÉSULTATS
# ============================================


def print_results(detection):
    """Affiche les résultats de manière claire"""

    total = detection['total_count']

    print("\n" + "="*60)
    print("📊 RÉSULTATS DE L'ANALYSE")
    print("="*60)

    print(f"\n📌 DÉTECTION DES PARTICULES :")
    print(f"   Sortie à Y = {OUTLET_Y} m (tolérance ±{OUTLET_TOLERANCE} m)")
    print(f"   ✅ PARTICULES ÉCHAPPÉES : {detection['escaped_count']} / {total}
({detection['escaped_count']/total*100:.1f}%)")
    print(f"   🛑 PARTICULES PIÉGÉES : {detection['trapped_count']} / {total}
({detection['trapped_count']/total*100:.1f}%)")

    print(f"\n📌 RÉPARTITION DES PARTICULES PIÉGÉES PAR ZONE :")
    zone_names = {
        'A': 'Zone A (paroi supérieure)',
        'B': 'Zone B (paroi inférieure)',
        'C': 'Zone C (cylindre de sortie)',
```

```
        'D': 'Zone D (zone conique)',
        'Autre': 'Autre zone'
    }

    for zone, particles in detection['trapped_by_zone'].items():
        if particles:
            zone_display = zone_names.get(zone, zone)
            pct_of_trapped = len(particles) / detection['trapped_count'] * 100 if
detection['trapped_count'] > 0 else 0
            pct_of_total = len(particles) / total * 100
            print(f"   {zone_display} : {len(particles)} particules ({pct_of_trapped:.1f}% des
piégées, {pct_of_total:.1f}% du total)")

    # Temps de résidence des échappées
    if detection['escaped']:
        times_escaped = [p['Time_final'] for p in detection['escaped']]
        print(f"\n⏱️ TEMPS DE RÉSIDENCE (ÉCHAPPÉES) :")
        print(f"   Min : {min(times_escaped):.6f} s")
        print(f"   Max : {max(times_escaped):.6f} s")
        print(f"   Moyen : {np.mean(times_escaped):.6f} s")


# ============================================
# 📊 FONCTION 4 : STATISTIQUES DÉTAILLÉES PAR ZONE
# ============================================


def print_zone_statistics(detection):
    """Affiche des statistiques détaillées par zone"""

    print("\n" + "="*60)
    print("📊 STATISTIQUES DÉTAILLÉES PAR ZONE")
    print("="*60)

    zone_names = {
        'A': 'Zone A (paroi supérieure)',
        'B': 'Zone B (paroi inférieure)',
        'C': 'Zone C (cylindre de sortie)',
        'D': 'Zone D (zone conique)',
        'Autre': 'Autre zone'
    }

    for zone, particles in detection['trapped_by_zone'].items():
        if particles:
            print(f"\n📍 {zone_names.get(zone, zone)} :")
            print(f"   Nombre de particules : {len(particles)}")

            # Temps de résidence
```

```
        times = [p['Time_final'] for p in particles]
        print(f"   Temps de résidence - Min : {min(times):.6f} s, Max : {max(times):.6f} s,
Moyen : {np.mean(times):.6f} s")

        # Vitesses finales
        velocities = [p['V_final'] for p in particles]
        print(f"   Vitesse finale - Min : {min(velocities):.1f} m/s, Max : {max(velocities):.1f} m/s,
Moyenne : {np.mean(velocities):.1f} m/s")

        # Positions X et Y
        x_positions = [p['X_final'] for p in particles]
        y_positions = [p['Y_final'] for p in particles]
        print(f"   Position X moyenne : {np.mean(x_positions):.3f} m")
        print(f"   Position Y moyenne : {np.mean(y_positions):.3f} m")


# ==========================================
# 📊 FONCTION 5 : GRAPHIQUE AVEC ZONES
# ==========================================


def plot_with_zones(detection, save_path=None):
    """Crée un graphique des positions finales avec zones colorées"""

    plt.figure(figsize=(14, 10))

    # Définir les couleurs par zone
    zone_colors = {
        'A': 'blue',
        'B': 'orange',
        'C': 'purple',
        'D': 'brown',
        'Autre': 'gray'
    }

    # Tracer les particules piégées par zone
    for zone, particles in detection['trapped_by_zone'].items():
        if particles:
            df_zone = pd.DataFrame(particles)
            color = zone_colors.get(zone, 'gray')
            plt.scatter(df_zone['X_final'], df_zone['Y_final'],
                        c=color, label=f'Zone {zone} ({len(particles)})', alpha=0.7, s=50)

    # Tracer les particules échappées
    if detection['escaped']:
        df_escaped = pd.DataFrame(detection['escaped'])
        plt.scatter(df_escaped['X_final'], df_escaped['Y_final'],
```

```
            c='green', label=f'Échappées ({len(detection["escaped"])})', alpha=0.7, s=50,
marker='*')

    # Ligne de sortie
    plt.axhline(y=OUTLET_Y, color='green', linestyle='-', linewidth=2, label=f'Sortie Y =
{OUTLET_Y} m')

    # Dessiner les rectangles des zones pour référence
    # Zone A
    rect_a = plt.Rectangle((ZONE_A['x_min'], ZONE_A['y_min']),
                           ZONE_A['x_max'] - ZONE_A['x_min'],
                           ZONE_A['y_max'] - ZONE_A['y_min'],
                           alpha=0.1, color='blue')
    plt.gca().add_patch(rect_a)

    # Zone B
    rect_b = plt.Rectangle((ZONE_B['x_min'], ZONE_B['y_min']),
                           ZONE_B['x_max'] - ZONE_B['x_min'],
                           ZONE_B['y_max'] - ZONE_B['y_min'],
                           alpha=0.1, color='orange')
    plt.gca().add_patch(rect_b)

    # Zone C
    rect_c = plt.Rectangle((ZONE_C['x_min'], ZONE_C['y_min']),
                           ZONE_C['x_max'] - ZONE_C['x_min'],
                           ZONE_C['y_max'] - ZONE_C['y_min'],
                           alpha=0.1, color='purple')
    plt.gca().add_patch(rect_c)

    # Zone D
    rect_d = plt.Rectangle((ZONE_D['x_min'], ZONE_D['y_min']),
                           ZONE_D['x_max'] - ZONE_D['x_min'],
                           ZONE_D['y_max'] - ZONE_D['y_min'],
                           alpha=0.1, color='brown')
    plt.gca().add_patch(rect_d)

    plt.xlabel('X Position (m)')
    plt.ylabel('Y Position (m)')
    plt.title(f'Positions finales des particules\n{detection["escaped_count"]} échappées,
{detection["trapped_count"]} piégées')
    plt.legend(loc='upper left', bbox_to_anchor=(1, 1))
    plt.grid(True, alpha=0.3)
    plt.axis('equal')
    plt.xlim(-0.35, 0.35)
    plt.ylim(-0.15, 0.25)

    if save_path:
```

```
        plt.savefig(save_path, dpi=150, bbox_inches='tight')
        print(f"\n💾 Graphique sauvegardé : {save_path}")

    plt.show()


# ============================================
# 🚀 MAIN
# ============================================


if __name__ == "__main__":

    print("\n🔍 MODE : Analyse d'un seul fichier avec détection par zone")
    print(f"   Fichier : {FILE_PATH}")

    # Charger les données
    df = load_particle_data(FILE_PATH)

    if df is not None:
        # Détecter les particules par zone
        detection = detect_particles_by_zone(df)

        # Afficher les résultats
        print_results(detection)

        # Afficher les statistiques par zone
        print_zone_statistics(detection)

        # Générer le graphique
        plot_with_zones(detection, save_path=PLOT_PATH if SAVE_PLOT else None)

        # Conclusion
        print("\n" + "="*50)
        print("🔍 CONCLUSION :")
        print("="*50)
        print(f"✅ Particules échappées : {detection['escaped_count']} / {detection['total_count']}
({detection['escaped_count']/detection['total_count']*100:.0f}%)")
        print(f"✅ Particules piégées : {detection['trapped_count']} / {detection['total_count']}
({detection['trapped_count']/detection['total_count']*100:.0f}%)")

        # Résumé par zone
        print(f"\n📌 RÉSUMÉ PAR ZONE :")
        zone_names = {
            'A': 'A (paroi supérieure)',
            'B': 'B (paroi inférieure)',
            'C': 'C (cylindre de sortie)',
            'D': 'D (zone conique)',
```

```
        'Autre': 'Autre'
    }
    for zone, particles in detection['trapped_by_zone'].items():
        if particles:
            zone_display = zone_names.get(zone, zone)
            print(f"   Zone {zone_display} : {len(particles)} particules")
```

b) **Code for histogram of escaped, trapped vs residence time**

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import os


# ==========================================
# 📂 USER PARAMETERS (modify according to your case)
# ==========================================

FILE_PATH = "H:/particles-temp-100.txt"


# Detection parameters
OUTLET_Y = 0.20           # Outlet Y position (m)
OUTLET_TOLERANCE = 0.01   # Tolerance for detecting outlet (m)


# Histogram parameters
N_BINS = 30               # Number of bins (intervals)
LOG_SCALE = True          # Logarithmic scale for X-axis
DENSITY = False           # False = count, True = probability density
SAVE_PLOT = True
PLOT_PATH = "H:/histogram_residence_time.png"


# ==========================================
# 👍 FUNCTION 1: LOAD DATA
# ==========================================


def load_particle_data(file_path):
    """Loads the particles-temp.txt file generated by Fluent"""

    if not os.path.exists(file_path):
        raise FileNotFoundError(f"❌ File not found: {file_path}")
```

```
    with open(file_path, 'r') as f:
        lines = f.readlines()

    # Find where data starts (after the "-----" line)
    start_idx = 0
    for i, line in enumerate(lines):
        if '-----' in line:
            start_idx = i + 2
            break

    # Read data
    data_lines = []
    for line in lines[start_idx:]:
        if line.strip():
            values = line.strip().split('\t')
            if len(values) >= 7:
                data_lines.append(values)

    # Create DataFrame
    columns = ['ResidenceTime', 'XPosition', 'YPosition', 'ParticleID',
               'XVelocity', 'YVelocity', 'ColorBy']

    df = pd.DataFrame(data_lines, columns=columns)

    # Convert to numeric
    for col in columns:
        df[col] = pd.to_numeric(df[col], errors='coerce')

    # Clean
    df = df.dropna()
    df = df[df['ParticleID'] >= 0]

    print(f"✅ Loaded: {df['ParticleID'].nunique()} particles, {len(df)}
points")
    return df


# ============================================
# 📊 FUNCTION 2: DETECT PARTICLES
# ============================================


def detect_particles(df, outlet_y=OUTLET_Y, outlet_tol=OUTLET_TOLERANCE):
    """
    Detects escaped and trapped particles with their residence time
    """
```

```
    escaped_times = []
    trapped_times = []

    for pid, group in df.groupby('ParticleID'):
        last = group.iloc[-1]
        y_final = last['YPosition']
        t_final = last['ResidenceTime']

        if abs(y_final - outlet_y) < outlet_tol:
            escaped_times.append(t_final)
        else:
            trapped_times.append(t_final)

    return escaped_times, trapped_times


# ==========================================
# 📊 FUNCTION 3: SIDE-BY-SIDE HISTOGRAM
# ==========================================

def plot_histogram_side_by_side(escaped_times, trapped_times,
save_path=None):
    """
    Creates a side-by-side histogram with two series
    """

    # Determine common limits for bins
    all_times = escaped_times + trapped_times
    min_time = min(all_times)
    max_time = max(all_times)

    if LOG_SCALE:
        # Logarithmic bins
        bins = np.logspace(np.log10(max(min_time, 1e-6)),
np.log10(max_time), N_BINS)
    else:
        bins = np.linspace(min_time, max_time, N_BINS)

    plt.figure(figsize=(12, 8))

    # Bar width
    width = 0.35

    # Bar positions
```

```
    x = np.arange(len(bins) - 1)

    # Histograms
    escaped_hist, _ = np.histogram(escaped_times, bins=bins)
    trapped_hist, _ = np.histogram(trapped_times, bins=bins)

    # Bin centers for display
    if LOG_SCALE:
        bin_centers = np.sqrt(bins[:-1] * bins[1:])  # Geometric mean
    else:
        bin_centers = (bins[:-1] + bins[1:]) / 2

    # Side-by-side bars
    plt.bar(bin_centers - width/2, escaped_hist, width, label=f'Escaped
({len(escaped_times)})',
            color='green', alpha=0.7)
    plt.bar(bin_centers + width/2, trapped_hist, width, label=f'Trapped
({len(trapped_times)})',
            color='red', alpha=0.7)

    # Formatting
    if LOG_SCALE:
        plt.xscale('log')
    plt.xlabel('Residence Time (s)', fontsize=12)
    plt.ylabel('Number of Particles', fontsize=12)
    plt.title(f'Residence Time Distribution\n{len(escaped_times)} Escaped,
{len(trapped_times)} Trapped', fontsize=14)
    plt.legend()
    plt.grid(True, alpha=0.3)

    if save_path:
        plt.savefig(save_path.replace('.png', '_side_by_side.png'),
dpi=150, bbox_inches='tight')
        print(f"💾 Graph saved: {save_path.replace('.png',
'_side_by_side.png')}")

    plt.show()


# ==========================================
# 📊 FUNCTION 4: OVERLAY HISTOGRAM
# ==========================================


def plot_histogram_overlay(escaped_times, trapped_times, save_path=None):
    """
```

```
    Creates an overlay histogram with two series (transparency)
    """

    all_times = escaped_times + trapped_times
    min_time = min(all_times)
    max_time = max(all_times)

    if LOG_SCALE:
        bins = np.logspace(np.log10(max(min_time, 1e-6)),
np.log10(max_time), N_BINS)
    else:
        bins = np.linspace(min_time, max_time, N_BINS)

    plt.figure(figsize=(12, 8))

    # Overlay histograms with transparency
    plt.hist([escaped_times, trapped_times], bins=bins,
             label=[f'Escaped ({len(escaped_times)})', f'Trapped
({len(trapped_times)})'],
             color=['green', 'red'], alpha=0.6, edgecolor='black',
linewidth=0.5)

    if LOG_SCALE:
        plt.xscale('log')

    plt.xlabel('Residence Time (s)', fontsize=12)
    plt.ylabel('Number of Particles', fontsize=12)
    plt.title(f'Residence Time Distribution\n{len(escaped_times)} Escaped,
{len(trapped_times)} Trapped', fontsize=14)
    plt.legend()
    plt.grid(True, alpha=0.3)

    if save_path:
        plt.savefig(save_path.replace('.png', '_overlay.png'), dpi=150,
bbox_inches='tight')
        print(f"💾 Graph saved: {save_path.replace('.png',
'_overlay.png')}")

    plt.show()


# ============================================
# 📊 FUNCTION 5: RESIDENCE TIME STATISTICS
# ============================================
```

```
def print_time_statistics(escaped_times, trapped_times):
    """Displays residence time statistics"""

    print("\n" + "="*60)
    print("📊 RESIDENCE TIME STATISTICS")
    print("="*60)

    print(f"\n🟢 ESCAPED PARTICLES ({len(escaped_times)}) :")
    print(f"   Min : {min(escaped_times):.6f} s")
    print(f"   Max : {max(escaped_times):.6f} s")
    print(f"   Mean : {np.mean(escaped_times):.6f} s")
    print(f"   Median : {np.median(escaped_times):.6f} s")
    print(f"   Std Dev : {np.std(escaped_times):.6f} s")

    print(f"\n🔴 TRAPPED PARTICLES ({len(trapped_times)}) :")
    print(f"   Min : {min(trapped_times):.6f} s")
    print(f"   Max : {max(trapped_times):.6f} s")
    print(f"   Mean : {np.mean(trapped_times):.6f} s")
    print(f"   Median : {np.median(trapped_times):.6f} s")
    print(f"   Std Dev : {np.std(trapped_times):.6f} s")


# ============================================
# 📊 FUNCTION 6: BOX PLOT
# ============================================


def plot_boxplot(escaped_times, trapped_times, save_path=None):
    """
    Creates a comparative box plot
    """

    plt.figure(figsize=(10, 8))

    data = [escaped_times, trapped_times]
    labels = [f'Escaped (n={len(escaped_times)})', f'Trapped
(n={len(trapped_times)})']
    colors = ['green', 'red']

    bp = plt.boxplot(data, patch_artist=True, labels=labels,
showfliers=True)

    for patch, color in zip(bp['boxes'], colors):
        patch.set_facecolor(color)
        patch.set_alpha(0.6)
```

```
    plt.yscale('log')
    plt.ylabel('Residence Time (s) - log scale', fontsize=12)
    plt.title('Comparison of Residence Times\nEscaped vs Trapped',
fontsize=14)
    plt.grid(True, alpha=0.3, axis='y')

    if save_path:
        plt.savefig(save_path.replace('.png', '_boxplot.png'), dpi=150,
bbox_inches='tight')
        print(f"💾 Graph saved: {save_path.replace('.png',
'_boxplot.png')}")

    plt.show()


# ============================================
# 🚀 MAIN
# ============================================


if __name__ == "__main__":

    print("\n🔍 RESIDENCE TIME ANALYSIS")
    print(f"   File: {FILE_PATH}")

    # Load data
    df = load_particle_data(FILE_PATH)

    if df is not None:
        # Detect particles
        escaped_times, trapped_times = detect_particles(df)

        # Display statistics
        print_time_statistics(escaped_times, trapped_times)

        # Create graphs
        plot_histogram_side_by_side(escaped_times, trapped_times,
save_path=PLOT_PATH if SAVE_PLOT else None)
        plot_histogram_overlay(escaped_times, trapped_times,
save_path=PLOT_PATH if SAVE_PLOT else None)
        plot_boxplot(escaped_times, trapped_times, save_path=PLOT_PATH if
SAVE_PLOT else None)

        # Conclusion
        print("\n" + "="*50)
        print("🔍 CONCLUSION :")
```

```
        print("="*50)
        print(f"✅ Escaped particles: {len(escaped_times)} /
{len(escaped_times) + len(trapped_times)}
({len(escaped_times)/(len(escaped_times)+len(trapped_times))*100:.0f}%)")
        print(f"✅ Trapped particles: {len(trapped_times)} /
{len(escaped_times) + len(trapped_times)}
({len(trapped_times)/(len(escaped_times)+len(trapped_times))*100:.0f}%)")
```

c) Code for particles – temperature analysis

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import os


# ============================================
# 📂 USER PARAMETERS (modify according to your case)
# ============================================


FILE_PATH = "H:/particles-temp-100.txt"


# Detection parameters
OUTLET_Y = 0.20           # Outlet Y position (m)
OUTLET_TOLERANCE = 0.01   # Tolerance for detecting outlet (m)


# Display parameters
SAVE_PLOT = True
PLOT_PATH = "H:/temperature_analysis.png"


# ============================================
# 🔧 FUNCTION 1: LOAD DATA
# ============================================


def load_particle_data(file_path):
    """Loads the particles-temp.txt file generated by Fluent"""

    if not os.path.exists(file_path):
        raise FileNotFoundError(f"❌ File not found: {file_path}")

    with open(file_path, 'r') as f:
        lines = f.readlines()
```

```
    # Find where data starts (after the "-----" line)
    start_idx = 0
    for i, line in enumerate(lines):
        if '-----' in line:
            start_idx = i + 2
            break

    # Read data
    data_lines = []
    for line in lines[start_idx:]:
        if line.strip():
            values = line.strip().split('\t')
            if len(values) >= 7:
                data_lines.append(values)

    # Create DataFrame
    columns = ['ResidenceTime', 'XPosition', 'YPosition', 'ParticleID',
               'XVelocity', 'YVelocity', 'ColorBy']

    df = pd.DataFrame(data_lines, columns=columns)

    # Convert to numeric
    for col in columns:
        df[col] = pd.to_numeric(df[col], errors='coerce')

    # Clean
    df = df.dropna()
    df = df[df['ParticleID'] >= 0]

    # ColorBy is particle temperature
    df = df.rename(columns={'ColorBy': 'Temperature'})

    print(f"✅ Loaded: {df['ParticleID'].nunique()} particles, {len(df)}
points")
    return df


# ============================================
# 📊 FUNCTION 2: DETECT PARTICLES WITH TEMPERATURE
# ============================================


def detect_particles_with_temperature(df, outlet_y=OUTLET_Y,
outlet_tol=OUTLET_TOLERANCE):
    """
```

```
    Detects escaped and trapped particles with their temperature data
    """

    escaped_data = []
    trapped_data = []

    for pid, group in df.groupby('ParticleID'):
        last = group.iloc[-1]
        y_final = last['YPosition']
        t_final = last['ResidenceTime']
        temp_final = last['Temperature']

        # Get entire trajectory for evolution
        times = group['ResidenceTime'].values
        temps = group['Temperature'].values

        particle_info = {
            'ParticleID': pid,
            'Time_final': t_final,
            'Temp_final': temp_final,
            'Time_series': times,
            'Temp_series': temps
        }

        if abs(y_final - outlet_y) < outlet_tol:
            escaped_data.append(particle_info)
        else:
            trapped_data.append(particle_info)

    return escaped_data, trapped_data


# ==========================================
# 📊 FUNCTION 3: TEMPERATURE VS TIME PLOT (ALL PARTICLES)
# ==========================================


def plot_temperature_vs_time(escaped_data, trapped_data, save_path=None):
    """
    Temperature vs time plot for all particles
    """

    plt.figure(figsize=(14, 8))

    # Plot escaped particles (green)
    for p in escaped_data:
```

```
        plt.plot(p['Time_series'], p['Temp_series'],
                 color='green', linewidth=0.5, alpha=0.5)

    # Plot trapped particles (red)
    for p in trapped_data:
        plt.plot(p['Time_series'], p['Temp_series'],
                 color='red', linewidth=0.5, alpha=0.5)

    # Mark final points
    escaped_final_times = [p['Time_final'] for p in escaped_data]
    escaped_final_temps = [p['Temp_final'] for p in escaped_data]
    trapped_final_times = [p['Time_final'] for p in trapped_data]
    trapped_final_temps = [p['Temp_final'] for p in trapped_data]

    plt.scatter(escaped_final_times, escaped_final_temps,
                c='green', s=30, marker='o', label=f'Escaped
({len(escaped_data)})', zorder=5)
    plt.scatter(trapped_final_times, trapped_final_temps,
                c='red', s=30, marker='s', label=f'Trapped
({len(trapped_data)})', zorder=5)

    plt.xscale('log')
    plt.xlabel('Residence Time (s) - log scale', fontsize=12)
    plt.ylabel('Temperature (K)', fontsize=12)
    plt.title(f'Particle Temperature Evolution\n{len(escaped_data)}
Escaped, {len(trapped_data)} Trapped', fontsize=14)
    plt.legend()
    plt.grid(True, alpha=0.3)

    if save_path:
        plt.savefig(save_path.replace('.png', '_temp_vs_time.png'),
dpi=150, bbox_inches='tight')
        print(f"💾 Graph saved: {save_path.replace('.png',
'_temp_vs_time.png')}")

    plt.show()


# ============================================
# 📊 FUNCTION 4: FINAL TEMPERATURE HISTOGRAM
# ============================================


def plot_temperature_histogram(escaped_data, trapped_data, save_path=None):
    """
    Histogram of final temperatures
```

```
    """

    escaped_temps = [p['Temp_final'] for p in escaped_data]
    trapped_temps = [p['Temp_final'] for p in trapped_data]

    plt.figure(figsize=(12, 8))

    bins = np.linspace(min(escaped_temps + trapped_temps),
                       max(escaped_temps + trapped_temps), 30)

    plt.hist([escaped_temps, trapped_temps], bins=bins,
             label=[f'Escaped ({len(escaped_temps)})', f'Trapped
({len(trapped_temps)})'],
             color=['green', 'red'], alpha=0.6, edgecolor='black',
linewidth=0.5)

    plt.xlabel('Final Temperature (K)', fontsize=12)
    plt.ylabel('Number of Particles', fontsize=12)
    plt.title(f'Final Temperature Distribution\n{len(escaped_data)}
Escaped, {len(trapped_data)} Trapped', fontsize=14)
    plt.legend()
    plt.grid(True, alpha=0.3)

    if save_path:
        plt.savefig(save_path.replace('.png', '_temp_histogram.png'),
dpi=150, bbox_inches='tight')
        print(f"💾 Graph saved: {save_path.replace('.png',
'_temp_histogram.png')}")

    plt.show()


# ==========================================
# 📊 FUNCTION 5: AVERAGE TEMPERATURE PLOT
# ==========================================


def plot_temperature_average(escaped_data, trapped_data, save_path=None):
    """
    Plot of average temperatures with standard deviation
    """

    plt.figure(figsize=(12, 8))

    if len(escaped_data) > 0:
        # Interpolation for escaped particles
```

```
        max_time_escaped = max([p['Time_final'] for p in escaped_data])
        time_grid = np.logspace(np.log10(1e-6), np.log10(max_time_escaped),
100)

        escaped_temps_interp = []
        for p in escaped_data:
            interp_func = np.interp(time_grid, p['Time_series'],
p['Temp_series'])
            escaped_temps_interp.append(interp_func)

        escaped_mean = np.mean(escaped_temps_interp, axis=0)
        escaped_std = np.std(escaped_temps_interp, axis=0)

        plt.plot(time_grid, escaped_mean, 'g-', linewidth=2, label='Escaped
(mean)')
        plt.fill_between(time_grid, escaped_mean - escaped_std,
escaped_mean + escaped_std,
                         color='green', alpha=0.2, label='Std Dev')

    if len(trapped_data) > 0:
        # Interpolation for trapped particles
        max_time_trapped = max([p['Time_final'] for p in trapped_data])
        time_grid = np.logspace(np.log10(1e-6), np.log10(max_time_trapped),
100)

        trapped_temps_interp = []
        for p in trapped_data:
            interp_func = np.interp(time_grid, p['Time_series'],
p['Temp_series'])
            trapped_temps_interp.append(interp_func)

        trapped_mean = np.mean(trapped_temps_interp, axis=0)
        trapped_std = np.std(trapped_temps_interp, axis=0)

        plt.plot(time_grid, trapped_mean, 'r-', linewidth=2, label='Trapped
(mean)')
        plt.fill_between(time_grid, trapped_mean - trapped_std,
trapped_mean + trapped_std,
                         color='red', alpha=0.2, label='Std Dev')

    plt.xscale('log')
    plt.xlabel('Residence Time (s) - log scale', fontsize=12)
    plt.ylabel('Temperature (K)', fontsize=12)
    plt.title('Average Particle Temperature vs Time', fontsize=14)
    plt.legend()
```

```
    plt.grid(True, alpha=0.3)

    if save_path:
        plt.savefig(save_path.replace('.png', '_temp_average.png'),
dpi=150, bbox_inches='tight')
        print(f"💾 Graph saved: {save_path.replace('.png',
'_temp_average.png')}")

    plt.show()


# ============================================
# 📊 FUNCTION 6: TEMPERATURE STATISTICS
# ============================================


def print_temperature_statistics(escaped_data, trapped_data):
    """Displays temperature statistics"""

    escaped_temps = [p['Temp_final'] for p in escaped_data]
    trapped_temps = [p['Temp_final'] for p in trapped_data]

    print("\n" + "="*60)
    print("📊 TEMPERATURE STATISTICS")
    print("="*60)

    if escaped_data:
        print(f"\n🟢 ESCAPED PARTICLES ({len(escaped_temps)}) :")
        print(f"   Initial mean temperature : {np.mean([p['Temp_series'][0]
for p in escaped_data]):.1f} K")
        print(f"   Final mean temperature : {np.mean(escaped_temps):.1f}
K")
        print(f"   Mean increase : {np.mean(escaped_temps) -
np.mean([p['Temp_series'][0] for p in escaped_data]):.1f} K")
        print(f"   Min temperature : {min(escaped_temps):.1f} K")
        print(f"   Max temperature : {max(escaped_temps):.1f} K")
        print(f"   Std Dev : {np.std(escaped_temps):.1f} K")

    if trapped_data:
        print(f"\n🔴 TRAPPED PARTICLES ({len(trapped_temps)}) :")
        print(f"   Initial mean temperature : {np.mean([p['Temp_series'][0]
for p in trapped_data]):.1f} K")
        print(f"   Final mean temperature : {np.mean(trapped_temps):.1f}
K")
        print(f"   Mean increase : {np.mean(trapped_temps) -
np.mean([p['Temp_series'][0] for p in trapped_data]):.1f} K")
```

```
        print(f"   Min temperature : {min(trapped_temps):.1f} K")
        print(f"   Max temperature : {max(trapped_temps):.1f} K")
        print(f"   Std Dev : {np.std(trapped_temps):.1f} K")


# ==========================================
# 📊 FUNCTION 7: FINAL TEMPERATURE VS TIME SCATTER PLOT
# ==========================================


def plot_temperature_vs_time_scatter(escaped_data, trapped_data,
save_path=None):
    """
    Scatter plot of final temperature vs residence time
    """

    plt.figure(figsize=(12, 8))

    escaped_final_times = [p['Time_final'] for p in escaped_data]
    escaped_final_temps = [p['Temp_final'] for p in escaped_data]
    trapped_final_times = [p['Time_final'] for p in trapped_data]
    trapped_final_temps = [p['Temp_final'] for p in trapped_data]

    plt.scatter(escaped_final_times, escaped_final_temps,
                c='green', s=50, label=f'Escaped ({len(escaped_data)})',
alpha=0.7)
    plt.scatter(trapped_final_times, trapped_final_temps,
                c='red', s=50, label=f'Trapped ({len(trapped_data)})',
alpha=0.7)

    plt.xscale('log')
    plt.xlabel('Residence Time (s) - log scale', fontsize=12)
    plt.ylabel('Final Temperature (K)', fontsize=12)
    plt.title(f'Final Particle Temperature vs Residence
Time\n{len(escaped_data)} Escaped, {len(trapped_data)} Trapped',
fontsize=14)
    plt.legend()
    plt.grid(True, alpha=0.3)

    if save_path:
        plt.savefig(save_path.replace('.png', '_temp_vs_time_scatter.png'),
dpi=150, bbox_inches='tight')
        print(f"💾 Graph saved: {save_path.replace('.png',
'_temp_vs_time_scatter.png')}")

    plt.show()
```

```python
# ============================================
# 🚀 MAIN
# ============================================


if __name__ == "__main__":

    print("\n🔍 PARTICLE TEMPERATURE ANALYSIS")
    print(f"   File: {FILE_PATH}")

    # Load data
    df = load_particle_data(FILE_PATH)

    if df is not None:
        # Detect particles with temperature
        escaped_data, trapped_data = detect_particles_with_temperature(df)

        # Display statistics
        print_temperature_statistics(escaped_data, trapped_data)

        # Create graphs
        plot_temperature_vs_time(escaped_data, trapped_data,
save_path=PLOT_PATH if SAVE_PLOT else None)
        plot_temperature_histogram(escaped_data, trapped_data,
save_path=PLOT_PATH if SAVE_PLOT else None)
        plot_temperature_average(escaped_data, trapped_data,
save_path=PLOT_PATH if SAVE_PLOT else None)
        plot_temperature_vs_time_scatter(escaped_data, trapped_data,
save_path=PLOT_PATH if SAVE_PLOT else None)

        # Conclusion
        print("\n" + "="*50)
        print("🔍 CONCLUSION :")
        print("="*50)
        if escaped_data:
            escaped_avg_temp = np.mean([p['Temp_final'] for p in
escaped_data])
            print(f"✅ Mean temperature of escaped particles :
{escaped_avg_temp:.1f} K")
        if trapped_data:
            trapped_avg_temp = np.mean([p['Temp_final'] for p in
trapped_data])
            print(f"✅ Mean temperature of trapped particles :
{trapped_avg_temp:.1f} K")
```

```
            print(f"   Difference : {trapped_avg_temp -
escaped_avg_temp:.1f} K higher for trapped particles")
```

Appendix C: Complete Experimental Data

Table C1: Complete experimental data for all particle counts and droplet sizes

Particles	Zone A	Zone B	Zone C	Zone D	Other
50	9	4	5	8	0
100	19	11	9	13	1
150	30	14	18	16	1
200	45	15	23	18	2
250	49	15	33	22	3

Table C2: Droplet size study (250 particles)

Diameter (μm)	Zone A	Zone B	Zone C	Zone D	Other
1	49	15	33	22	1
2	61	24	27	24	31
3.2	59	23	38	32	58
4	47	19	50	28	88
5	54	14	27	28	124

6	49	8	18	30	145

Table c3: Particle size study (250 particles)

Diameter (μm)	Zone A	Zone B	Zone C	Zone D	Other
1	49	15	33	22	3
2	61	24	27	24	31
3.2	59	23	38	32	58
4	47	19	50	28	88
5	49	18	27	28	14
6	49	18	18	30	5