Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie

WYDZIAŁ FIZYKI I INFORMATYKI STOSOWANEJ

KATEDRA ZASTOSOWAŃ FIZYKI JĄDROWEJ

Projekt dyplomowy

Automated 3D Visualisations of Particle-Based Atmospheric Cloud Simulations in the Cloud

Zautomatyzowane trójwymiarowe wizualizacje lagranżowskich symulacji chmur atmosferycznych w chmurze obliczeniowej

Autor:	Aleksandra Strząbała
Kierunek studiów:	Mikro– i Nanotechnologie w Biofizyce
Opiekun pracy:	Dr Sylwester Arabas

Kraków, 2026

Abstract

Computer simulations of atmospheric clouds provide a tool for researching phenomena ranging from microscale processes occurring on cloud droplets, through the dynamics of particle-laden flows, to large-scale atmospheric processes shaping weather and climate. Particle-based models of clouds leverage a probabilistic, Monte Carlo-type approach to address the multi-scale simulation challenges in coupling atmospheric thermo- and hydrodynamics with the multitude of processes shaping the microstructure of clouds. These processes include: formation of cloud droplets on condensation nuclei through diffusional growth, collisional growth and breakage of droplets, and transport of droplets due to advection and sedimentation.

The aim of the project summarised herein is the development and automation of visualisations of particle-based atmospheric cloud simulations. The key design goals are: visually appealing graphics; lightweight animated output suitable for inclusion in software documentation; provision of quantitative labelling; and automation enabling reproducibility and continuous deployment in the cloud.

The technological stack selected for the project consists of free and open-source software, including: ParaView (run in "headless" non-interactive mode using the pvpython scripting interface), Jupyter and Python; the open data format VTK; and the GitHub Actions automation platform. Visualisations are based on output from numerical simulations performed with two open-source research tools: the University of Warsaw Lagrangian Cloud Model (UWLCM) and PySDM — a Pythonic, particle-based cloud simulation software developed at the AGH University of Krakow. Simulation frameworks used in the project include a zero-dimensional parcel model, a two-dimensional prescribed-flow drizzling stratocumulus case, and a three-dimensional Large-Eddy Simulation of cumulus convection.

Two of the developed animated visualisations are already integrated into PySDM documentation generation workflow, triggered on each contribution to the public GitHub repository, thus ensuring that the animations featured in the documentation always represent the up-to-date state of the codebase. Selected results of this work were presented as scientific poster at the EuroSciPy conference in Kraków in August 2025.

Streszczenie

Symulacje komputerowe chmur atmosferycznych są narzędziem do badań szerokiego zakresu zjawisk fizycznych – od procesów mikroskalowych zachodzących na kroplach chmurowych, przez dynamikę przepływów, po procesy atmosferyczne kształtujące pogodę I klimat. Modele chmur bazujące na lagranżowskim opisie mikrofizyki wykorzystują probabilistyczne podejście typu Monte Carlo pozwalające sprostać wyzwaniom, jakie wynikają z szerokiego zakresu skal przestrzennych I czasowych zjawisk chmurowych oraz sprzężenia termodynamiki i hydrodynamiki atmosfery z licznymi procesami kształtującymi mikrostrukturę chmur. Do procesów tych należą: skraplanie wody na drobinach zanieczyszczeń, wzrost dyfuzyjny kropel, wzrost I rozpad kropel w wyniku zderzeń oraz ich transport spowodowany adwekcją I osiadaniem.

Celem projektu jest opracowanie oraz automatyzacja wizualizacji lagranżowskich symulacji chmur atmosferycznych. Dobór narzędzi I architektury projektu wynikał z następujących ramowych założeń: potrzeby atrakcyjnych dla oka wizualizacji; możliwości tworzenia animacji w rozmiarach dostosowanych do ograniczeń stron internetowych (w takiej formie publikowana jest dokumentacja projektu); umożliwienie ilościowej interpretacji wyników na podstawie wizualizacji; wymogu pełnej automatyzacji skutkującej odtwarzalnością I możliwością ciągłego wdrażania „w chmurze".

Technologie wykorzystane w projekcie obejmują wyłącznie otwarte oprogramowanie, m.in.: ParaView (uruchamiane w trybie „headless" poprzez interfejs skryptowy `pvpython`), Jupyter I Python; otwarty format danych VTK; oraz platformę automatyzacji obliczeń „w chmurze" GitHub Actions. Wizualizacje oparte są na wynikach symulacji numerycznych wykonanych przy użyciu dwóch otwartych narzędzi badawczych: University of Warsaw Lagrangian Cloud Model (UWLCM) oraz PySDM – pakietu dla języka Python do symulacji chmur rozwijanego na AGH w Krakowie. W ramach projektu wykorzystano różne rodzaje symulacji: zerowymiarowy model „cząstki powietrza", dwuwymiarowy model o zadanym przepływie naśladującym chmurę stratocumulus z mżawką oraz trójwymiarową symulację typu LES (Large Eddy Simulation) konwekcji w chmurze kłębiastej.

Dwie z opracowanych animowanych wizualizacji zostały zintegrowane z procesem generowania dokumentacji dla pakietu PySDM, uruchamianym przy każdej zmianie w publicznym repozytorium GitHub, dzięki czemu animacje prezentowane w dokumentacji zawsze odzwierciedlają aktualny stan kodu. Wybrane wyniki niniejszej pracy zostały zaprezentowane w formie plakatu naukowego na konferencji EuroSciPy w sierpniu 2025 r.

Contents

1. Introduction

This thesis is structured into seven chapters and two appendices:

Introductory sections 1.1 and 1.2 (based on [RY96; Mor+20; Paw23]) outline selected phenomena occurring in atmospheric clouds highlighting historical perspective on their discoveries, selected mathematical models used to describe these phenomena, and the basics and challenges of cloud microphysics modelling;

Tools section 2 offers a general description of the PySDM and UWLCM cloud modelling systems (based on [Bar+22] and [DWP19], respectively), the ParaView visualisation tool and file formats used in this project, as well as the Github Actions platform used for execution of visualisations "in the cloud";

Sections 3, 4 and 5 document the key tangible results of this diploma project – visualisations of numerical simulations of clouds performed using PySDM and UWLCM, the latter performed on a summer internship at the Institute of Geophysics, University of Warsaw;

Section 6 recalls fragments of developed scripts and workflows to outline the way user-controllable options are handled and engineered;

the Summary 7 reiterates key results and impact of this diploma project.

Appendix A details the steps needed to reproduce presented results with publicly available open-source code.

Appendix B includes a poster about the project presented at the EuroSciPy conference (August 2025, Kraków).

1.1 On microphysics of aerosol-cloud interactions

Year 1880 and the Nature paper of John Aitken [Ait80] can be regarded as the beginning of modern cloud physics[1]. Back then, the first conclusions were drawn that clouds and fogs form on atmospheric aerosol "germs". Based on a simple experiment with two glasses full of steam mixed with air, which

[1] Aitken admitted two months later [Ait81] that it was first reported 5 year earlier by Coulier and Mascart.

was filtered in one case by a cotton-wool philtre, researchers noticed that when it comes to the filtered air, the vapour stays uncondensed or super-saturated. However, if it is not pure, dust particles form nuclei and vapour can condense and create fog or cloud particles.

The germs of cloud formation are referred to as the cloud condensation nuclei (CCN). The chemical and physical properties of CCN are intrinsically linked to properties of clouds, and the probability of rain formation. Radiative processes (e.g., cooling due to emission of thermal radiation) and air flow dynamics (including turbulence) are processes that contribute to cloud and rain formation. Raindrops are those droplets that are large enough to fall and reach the ground. The two essential processes leading to cloud and rain formation are the growth of droplets by condensation and collisions.

1.1.1 Droplet growth by condensation

The growth of droplet in a vapour field can be described [see e.g., RY96] by the vapour density $\varrho_{\mathrm{v}}(\mathcal{R})$ (ϱ_{v} is equal to nm_0, where m_0 is the mass of one water molecule and n is the concentration of water molecules). The isotropy is assumed so that the concentration at distance $\mathcal{R}$ from the droplet centre ($n(\mathcal{R})$) depends only on $\mathcal{R}$. The vapour concentration satisfies the conditions of the diffusion equation:

$$\frac{\partial n}{\partial t} = D\nabla^2 n, \tag{1.1}$$

with t as time, molecular diffusion coefficient as D and ∇ denoting spatial differentiation. If the necessary boundary conditions are met and the stationary molecular flux to the droplet surface is $D\left(\frac{\partial n}{\partial \mathcal{R}}\right)_{\mathcal{R}=r}$, in terms of the density of the vapour, the increase in mass rate can be given by

$$\frac{dm}{dt} = 4\pi r D(\varrho_{\mathrm{v}} - \varrho_{\mathrm{vr}}), \tag{1.2}$$

where m is mass, ϱ_{vr} is the vapour density at the droplet's surface, r is the radius of the droplet. Analogously to this formula, a formula for diffusion of heat from the droplet can be derived as follows

$$\frac{dQ}{dt} = 4\pi r \mathcal{K}(T_r - T), \tag{1.3}$$

where heat is denoted as Q, T as ambient temperature, T_r is the surface temperature of the droplet and $\mathcal{K}$ is the thermal conductivity coefficient of air.

In a closed thermally insulated container, water and its vapour exchange molecules until the rate of evaporation is equal to the rate of condensation. At this point, the vapour is saturated, and its partial pressure is known as the saturation vapour pressure $e_{\mathrm{s}}(T)$. The transition of a substance from liquid to vapour requires an energy input, known as latent heat of vaporisation L. To describe this phase transition, there is a function known as the Gibbs function, that can be denoted as G for a given phase ϕ

$$G_\phi = u_\phi + e_{\mathrm{s}}\alpha_{\phi p} - T\phi_p, \tag{1.4}$$

where, u represents the internal energy, α a specific volume, and s the entropy. When phases are in equilibrium, the Gibbs functions are equal to $G_{\mathrm{liquid}} = G_{\mathrm{vapour}}$. This condition defines the saturation vapour pressure, the core concept to define relative humidity. Differentiating, and taking into account the fact that $dG_1 = dG_2$, the general Clapeyron relation can be obtained

$$\frac{de_{\mathrm{s}}}{dT} = \frac{s_2 - s_1}{\alpha_2 - \alpha_1}, \tag{1.5}$$

where α specific volume of phase, 1 and 2 represent liquid and vapour.

The ideal gas law describes a hypothetical gas (with gas particles that move in a random manner consistent with Newton's laws of motion) and relates the pressure p, volume $\mathcal{V}$, number of moles ν and absolute temperature T as

$$p\mathcal{V} = \nu \mathrm{R}T, \tag{1.6}$$

where R is the universal gas constant. an alternative form is $p\mathcal{V} = N\mathrm{k}T$, where N is the number of molecules and k is the Boltzmann constant, related to R by $\mathrm{R} = N_{\mathrm{A}}\mathrm{k}$, where N_{A} is Avogadro's number.

Water vapour can be treated as an ideal gas under ordinary atmospheric conditions where $\alpha_1 \gg \alpha_2$. In this case, the specific equation should satisfy $\alpha_2 = \mathrm{R_v}T/e_{\mathrm{s}}$, where R_{v} is the gas constant for water vapour. Using this relation and the definition of latent-heat, leads to the Clausius-Clapeyron equation

$$\frac{de_{\mathrm{s}}}{dT} = \frac{Le_{\mathrm{s}}}{\mathrm{R_v}T^2}. \tag{1.7}$$

If L is treated as a constant (equal to $2.5 \times 10^6 \mathrm{J/kg}$ at $T_0 = 0°\mathrm{C}$), an approxi-

mate expression for the saturation vapour pressure over water can be obtained

$$e_{\mathrm{s}}(T) = \mathrm{A}e^{-\mathrm{B}/T}, \tag{1.8}$$

where $\mathrm{A} \approx 2.5 \times 10^8$ kPa and $\mathrm{B} \approx 5.4 \times 10^3$K.

In a steady-state growth of a spherical droplet in a vapour-gas field, the temperature at the droplet surface adjusts, and the latent-heat balance leads to a vapour-temperature coupling characterised by:

$$\varrho_{\nu} - \varrho_{\nu r} = \frac{\mathcal{K}}{LD}(T_{\mathrm{r}} - T), \tag{1.9}$$

which connects the vapour density gradient with the gradient in temperature at the droplet surface and surrounding air. In this equation T_r and $\varrho_{\nu r}$ are unknown; to determine them, an additional relation is applied to give

$$\varrho_{\nu r} = \frac{e_{\mathrm{s}}(T_r)}{\mathrm{R_v}T_r}, \tag{1.10}$$

where $e_{\mathrm{s}}(T_r)$ is the saturation vapour pressure on a flat water surface at temperature T_r. The above equations are derived and can be solved, once one uses the ideal gas law and the Clausius-Clapeyron relation for $e_{\mathrm{s}}(T_r)$. Equations (1.9) and (1.10) can be solved numerically to determine the surface temperature of the droplet and the density of the vapour. An approximate analytical approach leads to an expression for the droplet growth rate:

$$r\frac{dr}{dt} = \frac{\mathrm{RH} - 1}{F_{\mathrm{k}} + F_{\mathrm{d}}}, \tag{1.11}$$

where RH is the relative humidity, and where the term F_{k} is associated with heat conduction and F_{d} with the diffusion of vapours.

Relative humidity RH represents the ratio of the actual water vapour content of the air to the saturation water vapour content at the same temperature and pressure. The formula for calculating it is as follows

$$\mathrm{RH} = \frac{w}{w_{\mathrm{s}}(p,T)} \approx \frac{e}{e_{\mathrm{s}}}, \tag{1.12}$$

where p symbolises pressure, w mixing ratio, w_{s} saturation mixing ratio, and e vapour pressure. In the interior of the cloud it is really unusual for the relative humidity to exceed the value of 102% and is generally in the range between 98% and 102%, while at their edges it drops to about 90% or less due

to mixing with dry air [RY96]. Although 100% humidity represents an equilibrium between condensation and evaporation, the process of droplet formation in the atmosphere does not begin precisely at this point. it requires the previously mentioned condensation nuclei. In perfectly clean air, without these particles, condensation would require a supersaturation of hundreds of percent. However, thanks to the abundance of aerosols, the typical supersaturation in clouds is only about 1%, which is enough for efficient formation of clouds.

In natural clouds, droplets interact with each other and their environment in ways that determine their size distribution and concentration. Droplets grow by condensation when air cools and shrink by evaporation when dry air mixes with clouds. Sedimentation primarily influences larger droplets, causing them to fall out of the cloud under gravity, whereas coalescence becomes significant only for droplets exceeding 10 µm in radius. During the initial stage of cloud formation, the droplets are too small to be affected by sedimentation or coalescence, and condensation dominates the growth. This process depends on the relative humidity, which in this context is usually referred to as saturation ratio. In an adiabatic framework (without mixing or radiative cooling effects), the saturation dynamics can be represented by

$$\frac{d\mathrm{RH}}{dt} = P - \mathcal{C} = Q_1 \frac{dz}{dt} - Q_2 \frac{d\chi}{dt}, \tag{1.13}$$

where P is a source term, $\mathcal{C}$ a condensation term, $\frac{d\chi}{dt}$ is the condensation rate measured in units of mass of condensate per mass of air per unit of time and $\frac{dz}{dt}$ is the vertical air velocity. The variables Q_1 and Q_2 represent thermodynamic variables and are given by:

$$Q_1 = \frac{1}{T}\left(\frac{\varepsilon L_\mathrm{g}}{\mathrm{R'} c_\mathrm{p} T} - \frac{g}{\mathrm{R'}}\right) \tag{1.14}$$

$$Q_2 = \varrho\left(\frac{\mathrm{R'} T}{\varepsilon e_\mathrm{s}} + \frac{\varepsilon L^2}{p T c_\mathrm{p}}\right), \tag{1.15}$$

where R' is a constant gas of air, g denotes the gravitational acceleration, c_p is the specific heat capacity of the constant pressure air and $\varepsilon = \mathrm{R_v}/\mathrm{R'} \approx 0.622$ is the ratio of gas constants for water vapour and dry air. In physical terms, $Q_1 \frac{dz}{dt}$ describes the increase in supersaturation that occurs as a result of cooling during the adiabatic ascent, and $Q_2 \frac{d\chi}{dt}$ expresses the opposing effect - the reduction of supersaturation as water vapour condenses on existing droplets.

1.1.2 Collisional growth and breakage

Due to the impact of gravitation on droplets, they can grow into rain droplets [see, e.g. the overview in Paw23]. Larger droplets fall faster and collide with smaller ones; which can lead to coalescence. This interaction is known as collisional growth. To describe the impact of it, the following simple model is outlined. The droplet with radius r_1 moves through a cylinder-shaped area, with a volume equal to $r_1 + r_2$, where r_2 represents a smaller droplet radius. The height of such a volume is proportional to $V(r_1) - V(r_2)$, which is a difference in the rate of droplet descent. Collision can only appear if both droplets are inside this volume, it can be depicted by the equation

$$\frac{dm_{r_1}}{dt} = \pi \cdot (r_1 + r_2)^2 \left[V(r_1) - V(r_2)\right] \mathcal{N} \left(\frac{4}{3}\pi r_2^3 \rho_l\right), \tag{1.16}$$

where m_{r_1} is a droplet mass with radius r_1, water density as ρ_l and $\mathcal{N}$ as concentration [Paw23]. Hydrodynamic airflow can affect this dependency, and droplets are diverted around the collector and do not actually collide. That is why collision efficiency $E(r_1, r_2)$ gets introduced to correct equation 1.16. To describe realistic growth in natural clouds, droplet size must be considered as a spectrum. In addition, the extension of a cylinder-shaped area has to be taken into account through the whole spectrum, which leads to the following results

$$\frac{dm_{r_1}}{dt} = \int (\frac{4}{3}\pi r_2^3 \rho_l)\pi (r_1 + r_2)^2 [V(r_1) - V(r_2)] E(r_1, r_2) n_r(r_2) dr_2, \tag{1.17}$$

where $n_r(r_2)$ is the distribution of the number of droplets with radius equal to r_2 [Paw23].

Collision does not always lead to coalescence; everything depends on physical characteristics such as droplet size or trajectory. Smaller droplets usually bounce off each other or merge and remain permanently joined. Less frequently, they fuse together, but not for good or split into multiple smaller droplets [Mor+20]. To mathematically describe it and measure how effective collisions are, the ratio of the number of mergers to the number of collisions is introduced; the ratio is referred to as the coalescence efficiency. It is approximated to be around 1, because most collisions lead to coalescence and can be enhanced by the electrical field. The collection efficiency, which is equal to the collision efficiency multiplied by the coalescence efficiency, determines the real proportion between droplets that collide and then are absorbed.

The collection efficiency is estimated to be equal to the coalescence efficiency [Mor+20].

Collision-coalescence growth affects droplet size and number concentration. Describe the evolution of the droplet number and size due to collisions is governed by the Smoluchowski coagulation equation, which provides a description of changes in the droplet population over time

$$\frac{\partial n(x,t)}{\partial t} = \frac{1}{2}\int_0^x n(x-y,t)\, n(y,t)\, K(x-y,y)\, dy - n(x,t)\int_0^\infty n(y,t)\, K(x,y)\, dy, \tag{1.18}$$

where $\frac{\partial n(x,t)}{\partial t}$ represents the rate of change in droplet concentration, $n(x,t)$ the number of droplet concentrations of volume x at time t, $K(x,y)$ the collision kernel - collision rate between droplets of volume x and y. The first term is the gain term; it represents the creation of droplets of volume x by merging two smaller droplets of volumes y and $x-y$. The second is the loss term, it depicts droplets of volume x collide with any droplet y and merge into a larger droplet [Paw23]. It is a deterministic approach. Collision-coalescence growth is treated rather statistically because it is impossible to track individual droplets.

1.2 Cloud microphysics modelling approaches

1.2.1 Eulerian moment- and bin-resolved methods

The first developments of microphysics schemes started around the 1950's and were not focused on improving the weather forecast, because of the lack of accuracy, but on understanding cloud processes [Ran+19]. Over time, two modelling approaches became commonly distinguished: the "bulk" approach and the "bin" approach. These models predict microphysical processes at discrete points in the atmosphere, small grid cells. By combining results from many of such cells, weather can be predicted on a large scale.

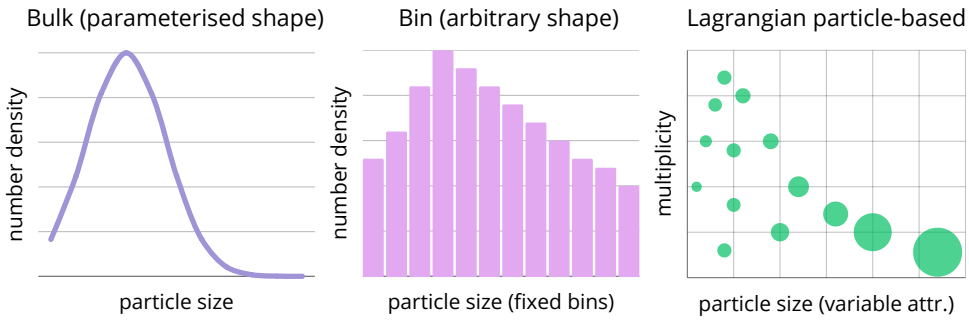


Figure 1.1: Comparison of particle size distribution representations in the three main types of cloud microphysics schemes: bulk (left), bin (centre), and Lagrangian particle-based (right). the x axis represents number of particles and y axis the size of particles.

The first approach shown on the left in Fig. 1.1, known as the "bulk" method, incorporates the separation of water into cloud water and rainwater, and as a consequence the continuity equations are formulated for the mixing ratio of bulk mass of vapour, cloud and rain. Bulk models parameterise the conversion processes between vapour, cloud and rain by defining sink/source terms for these continuity equations. The most important trait of bulk models is that they do not explicitly resolve the size spectrum of cloud droplets but represents them as a bulk mass. During later developments bulk models started to include ice particles and applied multi-moment schemes. Depending on the number of statistical moments of the size spectrum, the state of the model may include more information about droplets, starting with the mass, droplet quantity, and standard deviation among others. Since supersaturation within clouds depends on the size spectrum of droplets, multiple statistical moments are needed to resolve the supersaturation budget (as in eq. 1.13) – models based on a single moment are employing so-called saturation-adjustment approach where it is assumed that all vapour above saturation condenses instantaneously.

Statistical moments of the droplet size spectrum can be defined as:

$$M = \int_{\mathrm{a}}^{\mathrm{b}} n_r(r) m^{\mathrm{k}}(r) dr, \tag{1.19}$$

where M represents the total droplet mass if k = 1, a = 0 and $textb = \infty$, while other values of k, a and b yield different moments for different size ranges.

Bulk models were widely used in mesoscale models, the two-moment ap-

proach was applied to kilometre-scale models, and to model aerosols in climate models. The main downsides of bulk methods are the threshold-based categorisation across cloud and rain (as well as ice categories). As a result, bulk models have high sensitivity to the assumptions and parameters defining these categories [Mor+20].

The second approach shown in the middle of 1.1 is called the "bin" model. It numerically solves equations describing cloud and raindrop spectrum evolution without assuming a functional form of the spectrum. In such "bin" models, the size distribution function is discretised; this way, it is divided into non-overlapping intervals. Mathematically, it can be represented with a partial differential equation expressing continuity in both spatial and spectral dimensions:

$$\frac{dm}{dt} = \partial_t m + \nabla \cdot (\mathbf{u} m) + \partial_r \left(\dot{r}\, m \right), \tag{1.20}$$

where $\partial_t m$ represents the change in time, $\nabla \cdot (\mathbf{u} m)$ spatial advection (with $\mathbf{u}$ being the air speed vector) and $\partial_r \left(\dot{r}\, m \right)$ the change in particle size – spectral advection (with $\dot{r}$ being rate of change of particle radius). While "bin" models offer detailed information about the droplet size spectrum, and hence allow for coupling with supersaturation dynamics, there are numerical limitations hampering their applications: both in terms of computational cost, as well as accuracy (e.g, numerical diffusion).

To represent collisional growth and breakage of drops, "bin" models need to be equipped with a numerical method for solution of the Smoluchowski equation (1.18).

1.2.2 Lagrangian particle-resolved methods

An entirely different approach from the Eulerian (grid-based) bulk and bin methods was popularised during the mid 2000's and is referred to as the Lagrangian particle-based approach. It is shown on the right side of 1.1. This approach was applied in all the simulations presented herein. Particle-based method represents a population of real particles through super-particles (super-droplets), each associated with a multiplicity that denotes the number of actual particles it represents. Every super-particle has attributes such as position, particle size and composition. This method also allows us to obtain accurate results for S (eq.1.13). Similarly to the bin, it predicts the evolution of the particle size distribution but does so at the level of individual particles rather than size bins. Unlike bin models, this approach tracks real trajectories of particles,

eliminating numerical diffusion of the size or mass spectrum. Among the key advantages of the Lagrangian particle-based method is that it condensational growth/evaporation and particle transport are represented by ordinary differential equations, while collisional processes can be represented with efficient Monte Carlo methods.

With the growing number of super-particles around the amount of actual particles, particle-based models converge to a fully resolved particle-by-particle direct numeral simulation [Mor+20], which is not the case for bulk or bin methods. Super-particles have a set of prognostic attributes, and the computational cost of adding new attributes scales much more favourably than addition of new "dimensions" in "bin" models. This makes it possible to simulate processes responsible for aerosol-cloud interactions, for which tracking of the nucleus properties for each droplet is essential.

The accuracy of particle-based simulations depends on the number of employed super-particles (usually given per grid cell). Not only the model accuracy grows with the number of super-particles, but also computational cost, however the numerical schemes implemented in the software employed herein assure linear scaling of the cost with the number of super-particles.

Lagrangian particle-based schemes are mostly used in large-eddy simulations (LES) – i.e., relatively small domains and research-orientated computations. These simulations are used to develop and improve bulk microphysics parameterisations by deriving better estimates for process rates [Mor+20].

In this work, two simulation packages featuring Lagrangian particle-based models were employed: PySDM [Bar+22] and UWLCM [DWP19] (see the following Tools section for more information).

2. Tools employed in the project

2.1 PySDM

PySDM [Bar+22] is an open-source Python package maintained at the AGH University of Krakow. The package applies the Super-Droplet Method (SDM) for representing collisional growth of droplets (as described by Shima et al. 2009 [Shi+09]). Besides collisional processes, PySDM features representation of growth by vapour diffusion, nucleation of ice, and aqueous chemical reactions. From performance perspective, the key design trait of PySDM is the employment of just-in-time (JIT) compilation for both CPU and GPU.
PySDM has a modular architecture aimed at facilitating development or customisation without interaction with the core Package code. Among the developments carried out in this project, a new "exporter" module was development for storing simulation output in VTK file format.

2.2 UWLCM

UWLCM is a C++ open-source tool developed at University of Warsaw coupling Lagrangian particle-based cloud microphysics with large-eddy simulation (LES) for modelling turbulent atmospheric flows [DWP19]. As in PySDM, particle collisional processes are represented using the SDM Monte-Carlo approach. Output of UWLCM simulations used in this project was provided in XDMF and HDF5 file formats.

2.3 ParaView

ParaView is an open-source engine for scientific visualisation of two- and three-dimensional data [Aya15]. The name "ParaView" has its origin from the word parallel, because developers from the start planned the software for parallel work on clusters and supercomputers. It is one of the most popular visualisation tools used for high performance computing (HPC), it allows massive datasets to be visualised in-situ with distributed rendering. The genesis of this software lies in the collaboration, established in 2000, between Kitware and the Los Alamos National Laboratory [Kit24].

ParaView can run on all major operating system platforms such as Windows, macOS, and Linux, on machines ranging from laptops up to powerful

supercomputers. ParaView applications span material science, computational fluid dynamics, medical sciences and beyond.

ParaView offers a graphical user interface as well as `pvpython`/`pvbatch` Python API. The desktop application with GUI [Kit24] provides an interactive environment and allows to develop visualisations without programming. On the other hand, pvpython gives us full access to ParaView functions with Python, enabling reproducibility, automation, and integration with external Python libraries. The Python scripts can be created by a "Trace" function in the GUI, where from choosing start until clicking end, at any given moment, every step done in the GUI is rewritten into a `pvpython` script. It is a viable prototyping approach, but it produces machine-generated code that is generally bloated and has many unneeded lines. Pvbatch is a specialised tool dedicated to large scale computations on HPC architecture and follows similar principles to pvpython.

Among the ParaView features used in this project, there are [Mor18]:

- time-stepping of 3D animations;
- "Calculator" performing mathematical operations on data, for each point or cell individually, to rescale it by a component;
- "Slice" and "Clip" filters for extracting two-dimensional cross-section without removing geometry, and removal of a part of the geometry, respectively;
- "Threshold" and "Contour" to produce surfaces representing constant field values;
- "Glyph" for placing small arrows, for example, on every point and points in the direction based on data sets;
- output storage as video or still frames;
- VTK, HDF5 and XDMF file format readers.

2.4 File formats: VTK, HDF5, XDMF

The most commonly used data format loaded into ParaView is VTK (native format of Visualisation Toolkit; ParaView is a visualisation application founded on top of VTK). This format supports the preservation of many time steps in a single file. It is an open-source format to represent scientific

data for use with computer graphics and visualisations [SML06]. The data model used in VTK allows for both structured or unstructured data, which can posses different topological dimensions, ranging from zero-dimensional forms like points to three-dimensional volumes. These features determine how it organises datasets into points and cells, and attributes data such as scalars or vectors. Regular grids calculate point positions from spacing, whilst an unstructured grid stores all point coordinates and cell connections directly.

HDF5 - Hierarchical Data Format - is intended for large amounts of data, and besides being a file format, also includes a data model and a software library [Fol+11]. It supports virtually any datatype and is designed for high-performance computing. This format is both portable and extensible, enabling long-term use and covers many applications. HDF5 organises data hierarchically, it is based on three main foundations: datasets, attributes and groups. Datasets store multidimensional arrays with clearly defined shapes and datatypes, groups work as a connection between different HDF5 information items to achieve navigable structure and attributes provide metadata that annotate objects in the file [Fol+11]. All of these combined components deliver flexible, graph-like hierarchy intended for complex data models. However, it is not only equipped with ability to handle high-performance data, but also supports parallel input/output through MPI to handle data in HPC environment efficiently. HDF5 has a flexible filter pipeline that allows transparent compression and other transformations [Fol+11]. Broad datatype support and core library written in C equipped with binding to many popular languages (e.g., Python, C++) are further advantages of this technology allowing it to be useful in commercial and open-source applications. HDF5 is used, e.g., for satellite data storage, in climate and weather modelling systems, and in bioinformatics.

XDMF builds on top of HDF5, providing a description of the mesh, topology, geometry, and associated fields. XDMF stands for eXtensible Data Model and Format and is designed for consistent scientific data exchange across HPC workflows and visualisation tools. It separates metadata and structure that are kept in eXtensible Markup Language (XML) from large numerical arrays stored in HDF5, allowing clear descriptions of meshes and fields while storing heavy data in binary container [CCH12]. An XDMF dataset is organised in XML as a hierarchy: a "Domain" contains "Grid" elements; and each of these elements defines "Topology", "Geometry", and optionally "Attributes" [CCH12]. Large arrays typically live externally in HDF5 datasets

referenced from XML, while XML carries a light and descriptive part. "Topology" encodes connections between mesh entities, "Geometry" defines the spatial coordinates of nodes, and "Attributes" hold field data. Data values come from "DataItem" elements that can be embedded directly or linked to external storage, for example, HDF5, to describe different structures and layouts XDMF provides several "DataItem" families [CCH12]. XDMF is used by modern visualisation software, like ParaView, and was developed with HPC in mind.

2.5 GitHub Actions

GitHub Actions is an automation framework integrated with the GitHub platform. It is an event-driven API for automated development workflows and trigger actions in response to a specific event. These events can be standard development operations like pushing a new commit or creating a pull request. The key to achieving automation in the system is a workflow, written in a YAML configuration file and stored inside the "./github" folder, allowing to specify when and how automated tasks should be executed [Kin+21]. Workflow can consist of jobs and steps, enabling the construction of comprehensive pipelines customised for specific software. GitHub Actions allows users to create custom actions; each one of them requires a metadata file that specifies input parameters, outputs, and entry points executed through the workflow [Kin+21]. This characteristics allow them to be reused, shared, and combined into large pipelines.

To enable execution of automated workflows, GitHub Actions is equipped with runners that are computational environments responsible for providing the tasks defined in jobs. GitHub provides hosted runners as part of its cloud infrastructure, enabling programmers to avoid maintaining dedicated build servers [Kin+21]. However, programmers can prefer to configure self-hosted runners, as it offers greater control and compatibility with certain projects.

3. Microscale visualisations with PySDM

3.1 Parcel model framework

The simplest cloud microphysical modelling framework can be constructed by supplementing the equations (1.11) and (1.13) with thermodynamic relations that govern the evolution of an adiabatically isolated air parcel. The parcel is assumed to undergo no exchange of heat or mass with the surrounding environment.

Under adiabatic conditions, the temporal evolution of the parcel temperature can be described by

$$\frac{dT}{dt} = -\frac{\mathrm{g}}{c_\mathrm{p}}\frac{dz}{dt} + \frac{L_\mathrm{v}}{c_\mathrm{p}}\frac{dw}{dt}, \tag{3.1}$$

where L_v represents the latent heat of vaporisation [RY96].

The parcel is also assumed to ascend vertically through a hydrostatically stratified atmosphere. As a consequence of hydrostatic balance, the pressure tendency of the parcel is given by

$$\frac{dp}{dt} = \rho' \mathrm{g}\frac{dz}{dt}, \tag{3.2}$$

where ρ' denotes air density.

Within this framework, cloud microphysical evolution is driven by coupled variations of temperature, pressure, and supersaturation. The total mass of liquid water is split across a population of droplets such that m_i denotes the time-varying mass of a droplet belonging to the size category i with multiplicity ξ_i. Water vapour condenses onto a fixed population of pre-existing solution droplets, some of which growing beyond their critical size for activation into cloud droplets. The critical size depends on the mass and hygroscopicity of soluble material present in a droplet, and separates regimes where (under slightly supersaturated conditions) a positive perturbation to the droplet mass results in dampening (unactivated aerosol) or amplification of the perturbation (activated droplets).

In the initial stage of cloud formation considered in this work, droplet growth is assumed to be governed exclusively by diffusional condensation, hence processes such as collision-coalescence and sedimentation are neglected.

The model is therefore fully specified by the prescribed vertical velocity, the initial thermodynamics of the air of the air parcel, and the initial droplet population.

3.2 Sample simulation

Figure 3.1 illustrates the evolution of the parcel model over time. Droplet colours represent the volume of liquid water within each size category, highlighting its temporal growth. The colour of the rectangular background indicates the relative humidity RH of the parcel. It can be observed that both droplet volume and RH increase during the ascent of the parcel, reflecting the combined effect of adiabatic cooling and condensational growth.

3.3 Visualisation methodology

For the purposes of this project, a custom VTK exporter was developed for PySDM parcel model to enable visualisation in ParaView. This exporter is implemented as the "VTKExporterParcel" class, extending the standard PySDM "VTKExporter" and providing functionality for exporting particle attributes along with parcel-model geometry in the VTK format. Particle data are written as point clouds, while thermodynamic profiles, such as relative humidity, are exported on an unstructured grid. Time series data are organised in .pvd collection files, allowing animation of the parcel state over time. The spatial extent of each cell of the grid is dynamically calculated on the basis of the volume of the parcel, ensuring a physically consistent representation of the expansion of the parcel during the ascent.

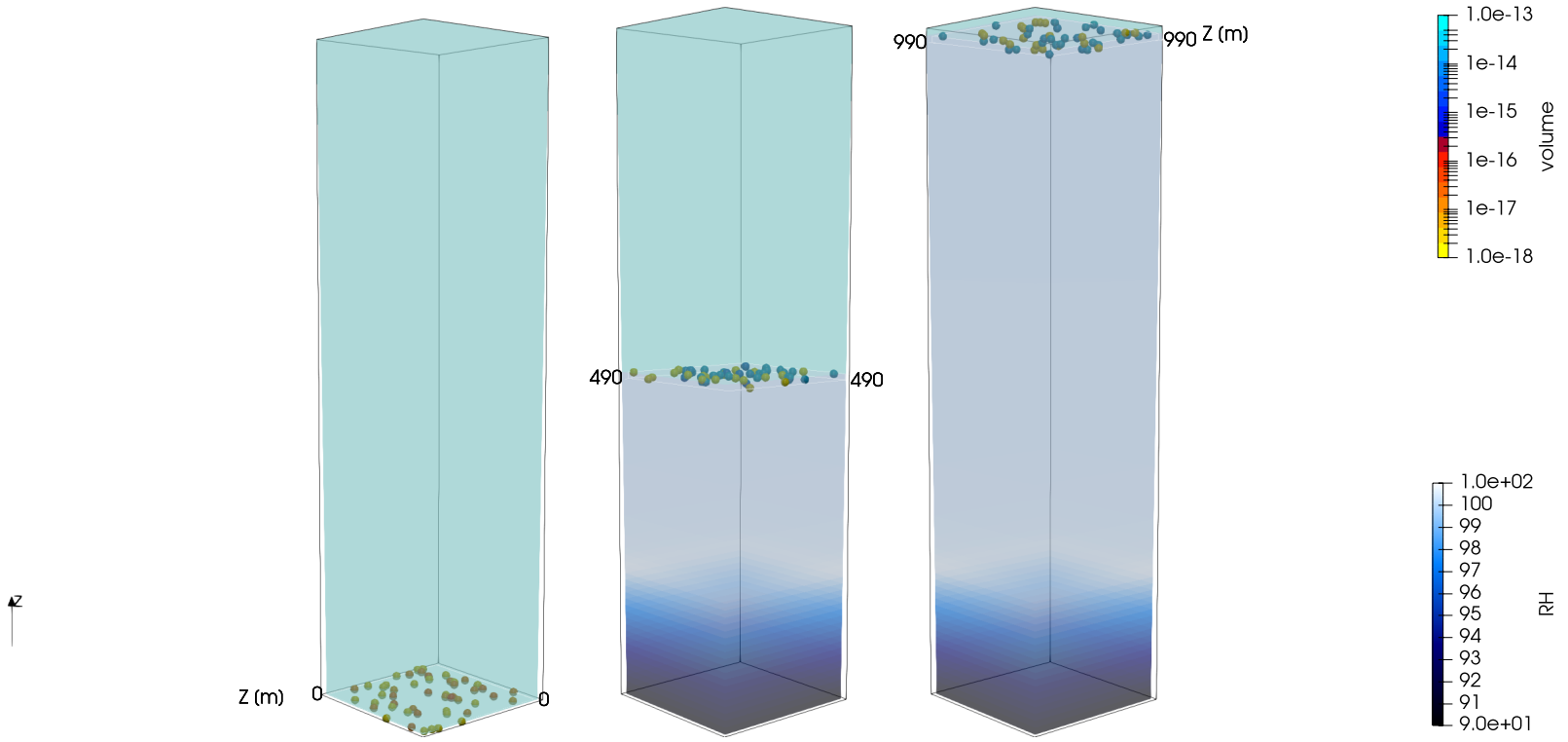


Figure 3.1: Parcel model implemented in PySDM showing the development of thermodynamic state and microphysical characteristics of particles over time.

4. Macroscale visualisations with PySDM

Physical processes included in PySDM extend beyond the parcel model to more complex multi-dimensional environments. It enables simulation of cloud microphysics coupled with atmospheric flows. The simplest flow-coupled simulation can be achieved with a 2D kinematic framework which provides a bridge between idealised 0D models and full 3D large eddy simulations.

4.1 2D kinematic framework (prescribed flow in a vertical air slab)

The 2D prescribed-flow framework used in PySDM can be traced back to the work of Kessler (1969) [Kes69]. The flow field is described analytically with a stream function. The setup used herein mimics a stratiform cloud deck and features periodic horizontal boundary conditions with vanishing flow at vertical boundaries. The framework applies a single-eddy flow field, resulting in an updraft-downdraft pair within the domain. This flow advects two Eulerian scalar fields: water vapour mixing ratio and dry air potential temperature. Eulerian advection is handled using the PyMPDATA library [Jon+23].

4.2 Scalar fields

Scalar fields in the 2D kinematic simulations represent thermodynamic quantities defined on the Eulerian grid: water vapour mixing ratio q_v and the dry-air potential temperature θ. These fields evolve through advection by the prescribed flow and interact with Lagrangian super-droplets through condensation, governed by the droplet growth equation (1.11). The supersaturation field S can be derived from q_v and θ, while microphysical fields such as the moments of cloud droplets size spectrum, and the liquid water content can be derived from particle attributes.

Figure 4.1 illustrates the spatial distribution of the effective radius (ratio of the third and second statistical moments of droplet radii) of the cloud drop at different simulation times, showing how condensational growth interacts with the flow structure.

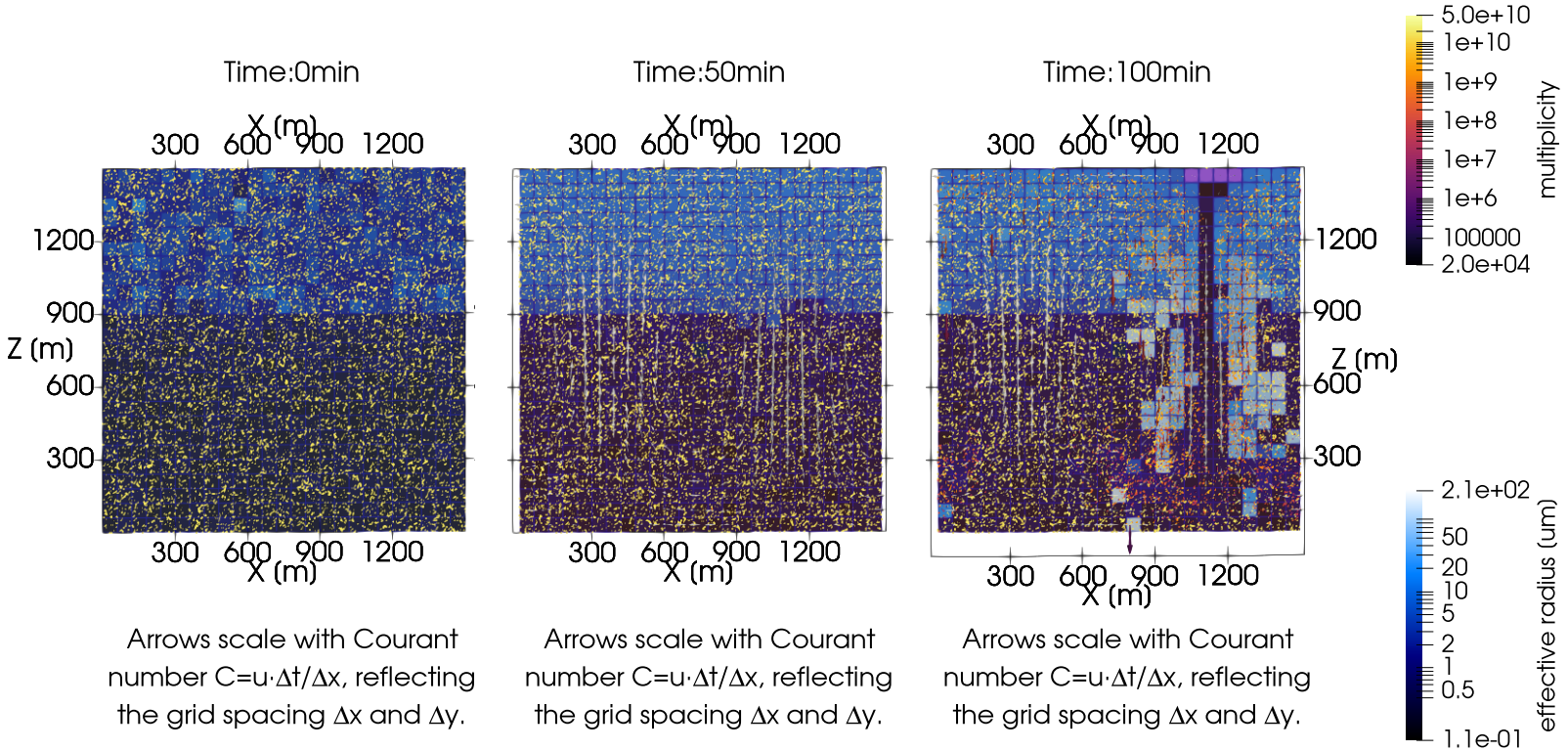


Figure 4.1: Sample output animation frames. Simulation times are $T_S = 0$ min, $T_S = 50$ min, $T_S = 100$ min. Data sources for the animation are outputs of PySDM simulation - products and attributes objects. Grid source data is representing products.effectiveradius, particle data is representing attributes.multiplicity and arrow data is representing the flow of particles.

4.3 Vector fields

The prescribed flow field is represented as a 2D vector field $\mathbf{u} = (u, w)$ defining the horizontal and vertical velocity components. In visualisations, vector fields are depicted as glyphs scaled according to velocity magnitude or Courant number $C = u \triangle t / \triangle x$, illustrating the transport of both Eulerian scalars and Lagrangian particles.

The right panel of figure 4.2 shows the flow field visualised with arrows scaled by the Courant number, highlighting regions of strong advection and numerical stability constraints.

4.4 Particle attribute fields

Lagrangian super-droplets are characterised by a set of attributes that evolve during the simulation. These include position (x, z), multiplicity ξ (number of real droplets represented), volume V or radius r, wet/dry aerosol composition, and activation status. The aerosol composition can be initialised using the framework introduced in [Jon+23]. During simulation, attributes evolve via microphysical processes: radius change by condensation (1.11), transport by advection/sedimentation, and changes to multiplicity and size due to stochastic collisions (simulated using a Monte-Carlo alternative to the Smoluchowski equation 1.18). In visualisation, values of these attributes are mapped to colour, size, and opacity, enabling analysis of droplet growth, collision-coalescence, and precipitation formation.

Figures 4.1 and 4.3 show particle data visualised according to multiplicity and combined with scalar fields, demonstrating the coupling between Lagrangian particles and Eulerian flow within the kinematic framework.

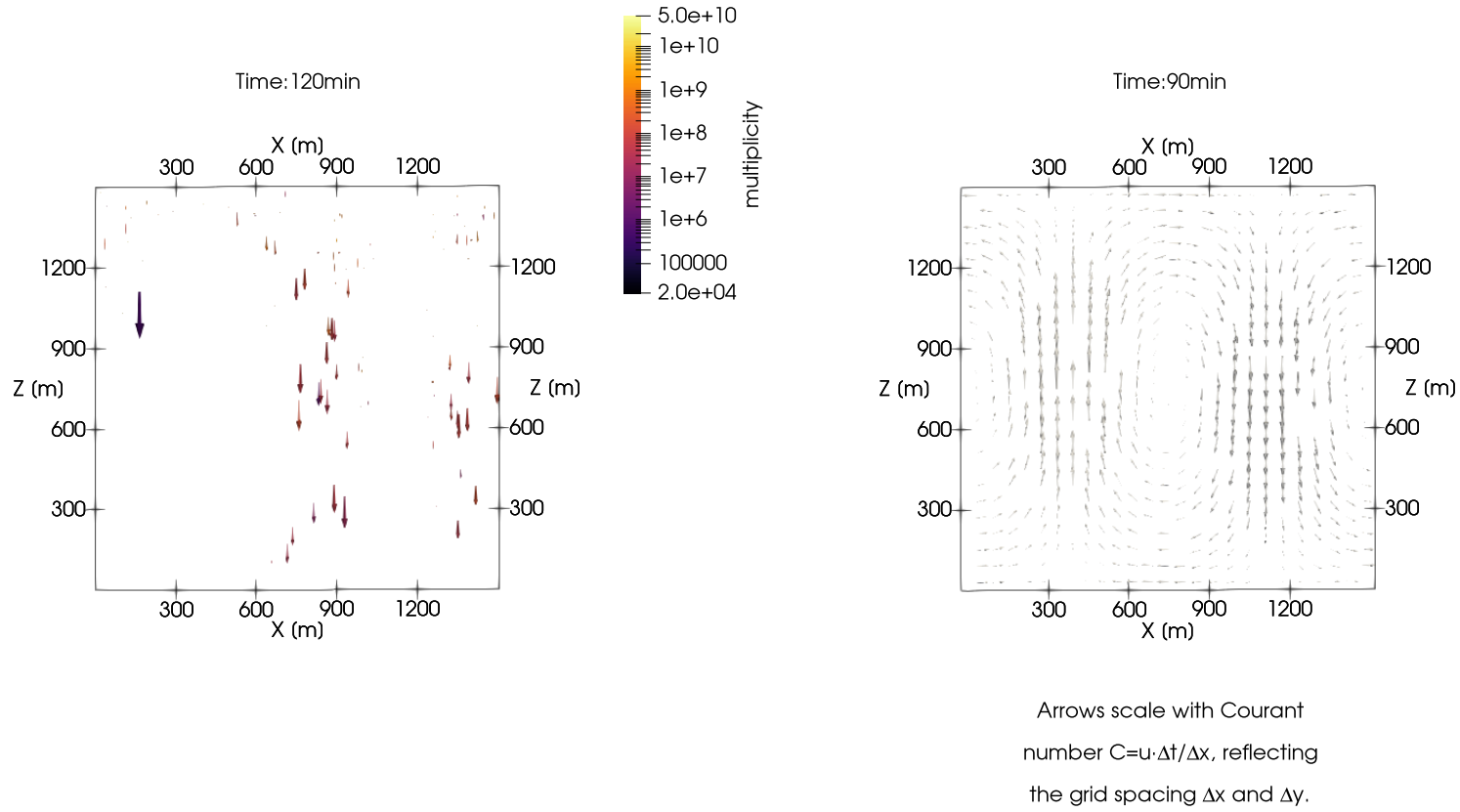


Figure 4.2: Sample output animation frames from a PySDM simulation. The left panel shows particle flow at $T_S = 120$ min, while the right panel shows arrow scaling based on the Courant number $C = u\,\Delta t/\Delta x$ at $T_S = 90$ min.

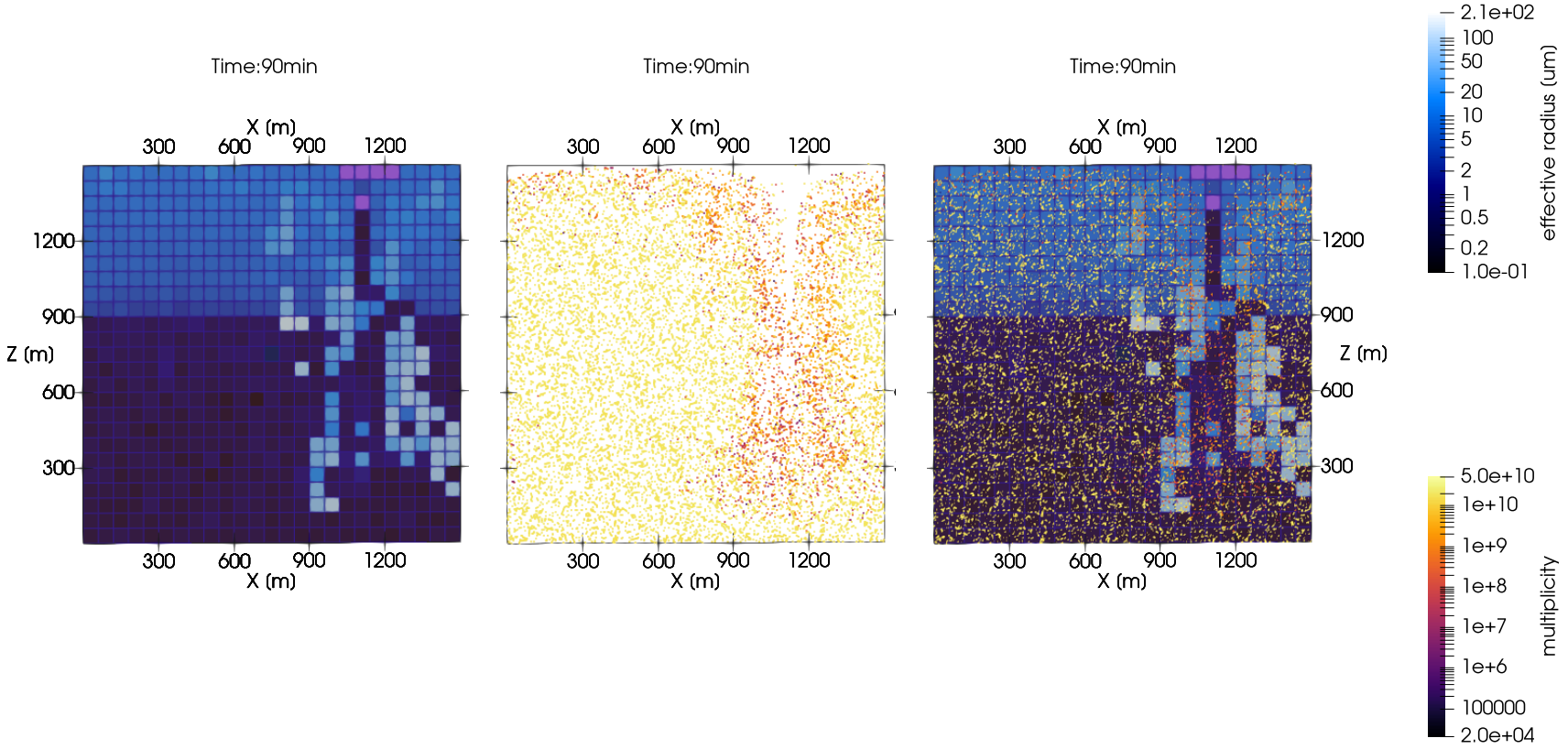


Figure 4.3: Sample animation frames. From left to right: grid source (products.effectiveradius), particle data (attributes.multiplicity) and their combined visualisation. Simulation time: $T_S = 90$ min. Data from PySDM simulation.

5. Macroscale visualisations with UWLCM (Summer Internship at University of Warsaw)

5.1 Shallow cumulus setup for a 3D LES framework

To test the UWLCM scheme in a shallow cumulus environment, a benchmark is applied, called BOMEX (Barbados Oceanographic and Meterological Experiment); it has been used largely as a benchmark problem in the LES frameworks of non-precipitating shallow cumulus convection over the ocean [Sie+03]. The setup included initial vertical profiles of potential temperature, specific humidity and horizontal wind; prescribed large-scale forcing (subsidence and advective tendencies for heat and moisture); surface fluxes of sensible and latent heat; radiative cooling profile [Sie+03]. The BOMEX problem has highly standard initial and boundary conditions, allowing direct comparison with a variety of LES models employing Eulerian bulk/bin microphysics and Lagrangian particle-based schemes [DWP19], among others. Furthermore, this makes it an ideal test case for validation of performance under a convective boundary layer dominated by cloud-scale updrafts and mixing such as in UWLCM. The initial and boundary conditions, as well as the aerosol size distribution for this project, follow the standard BOMEX framework, which defines a clean maritime environment.

Model configuration for BOMEX simulation:

1. Microphysics: the Lagrangian Super-Droplet Method solver from the libcloudph++ library, providing a high-resolution, numerical-diffusion-free droplet size spectrum evolution [Ara+15; DWP19].

2. Computational backend: While the CPU computes Eulerian dynamics, Lagrangian microphysics are computed in parallel at the same time on GPUs (high parallelisation and computational efficiency) [DZ22].

3. Subgrid-scale turbulence: Unsolved turbulent motions are parameterised by a Smagorinsky-type model [DWP19].

4. Temporal discretisation: a dynamical core time step of $0.5s$, with substepping giving an effective time step $0.1s$ for the condensation and coalescence processes [DWP19].

Employing the BOMEX setup allows for isolation and validation of the core model physics under controlled, homogeneous oceanic conditions. This part is vital before introducing complicated continental environments. Successful validation in idealised conditions instils confidence in the physical correctness of the model for its subsequent application in a more complex land-based situation. Alternatively, if the focus is on precipitation from shallow cumulus, the RICO (Rain in Cumulus over the Ocean) setup may be utilised. RICO generalises similar to the BOMEX setup to include precipitation processes, modifying the initial sounding, large-scale forcing, and microphysical parameters for the light rain formation. The UWLCM supports BOMEX and RICO test cases.

5.2 Sample visualisations

The visualisations presented in this section were developed during a research internship at the University of Warsaw, where the UWLCM model is developed. The simulations used for these visualisations were provided by the University of Warsaw. They are based on 3D Large-Eddy simulation (LES) of shallow cumulus convection. A custom pvpython script was written to visualise the output, offering user-configurable options such as: camera angle, colourmap selection, or photo file to be used as a background.

Figures 5.1, 5.2 and 5.3 present the scientific mode of visualisation presenting data more clearly with insights on what the data really represent. Figure 5.4 presents the realistic mode, which is more visually appealing but does not represent the data as well.

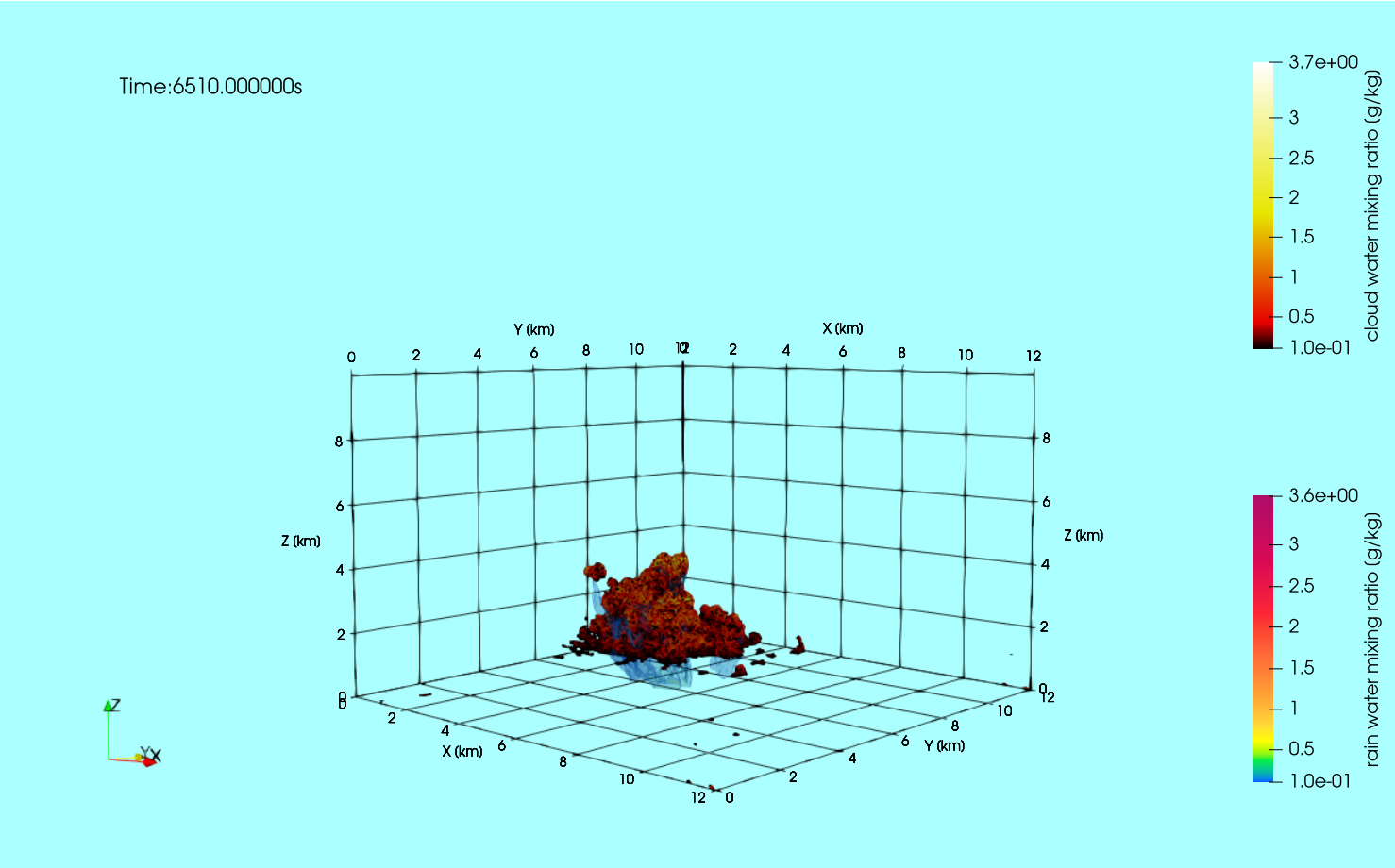


Figure 5.1: Visualisation at T=6510s, using a light-blue background with grid lines and labelled axes. Cloud water mixing ratio (rc) is shown in "Black-Body Radiation" colourmap, while rain water (rr) appears in "Rainbow Uniform", with transparency revealing internal cloud structures.

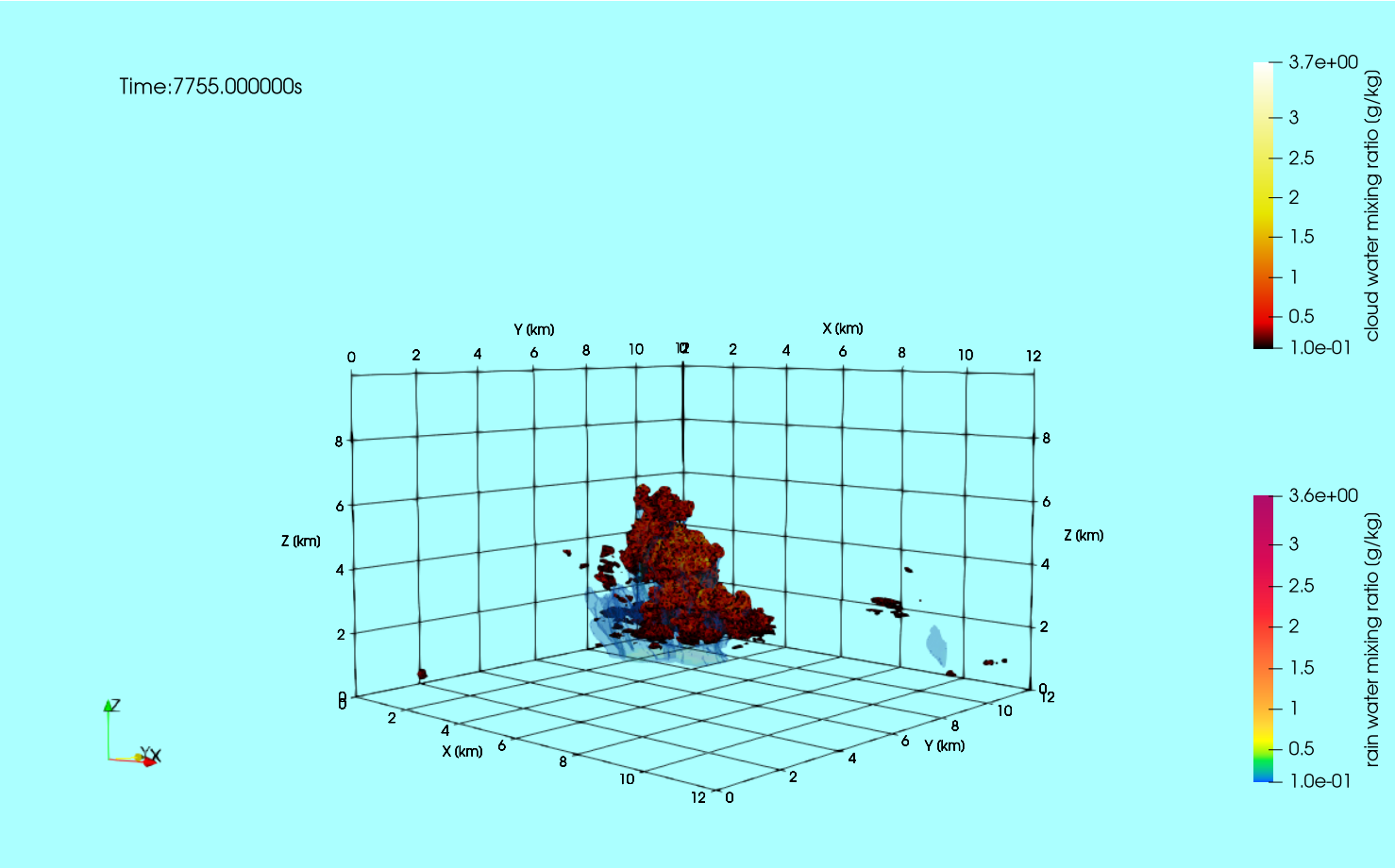


Figure 5.2: Visualisation at T=7755s, using a light-blue background with grid lines and labelled axes. Cloud water mixing ratio (rc) is shown in "Black-Body Radiation" colourmap, while rain water (rr) appears in "Rainbow Uniform", with transparency revealing internal cloud structures.

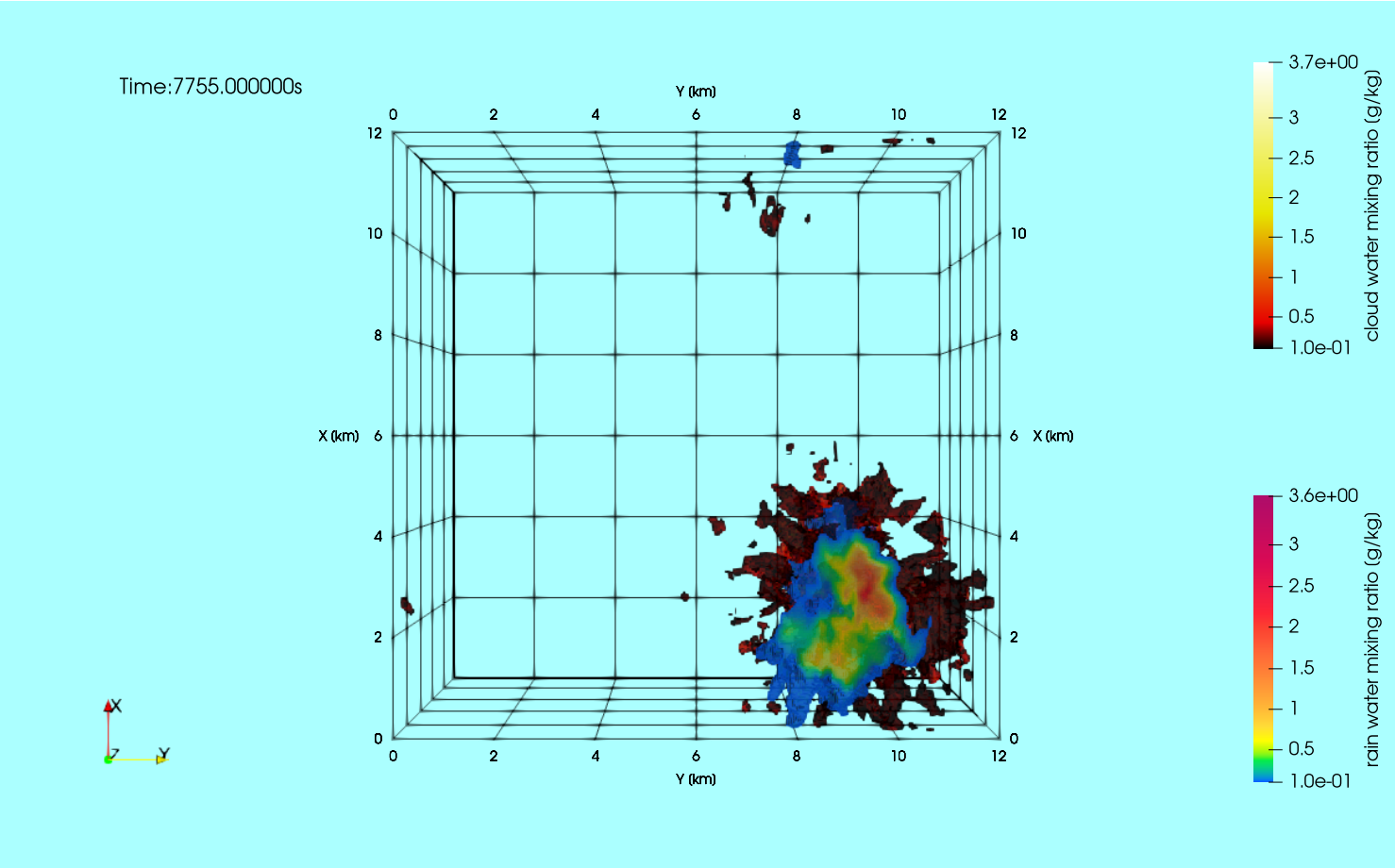


Figure 5.3: Visualisation at T=7755s from a different camera angle, using a light-blue background with grid lines and labelled axes. Cloud water mixing ratio (rc) is shown in "Black-Body Radiation" colourmap, while rain water (rr) appears in "Rainbow Uniform", with transparency revealing internal cloud structures.

Figure 5.4: Visualisation at T=6510s, visualised in realistic mode, with cloud volumes overlaid on a photographic sky background of Warsaw. Axes and grid lines are removed.

6. Engineering solutions

6.1 Handling user-controllable options

To achieve flexibility and customisability of the visualisation, all developed animations feature user-controllable options. Designed using the Python "argparse" package, all of them were equipped with a command-line interface. This characteristic allows users to manipulate visualisation parameters without the need to understand the source code, when running the script directly from a terminal or when executing it from a Jupyter Notebook. The use of a unified command-line interface ensures a consistent way of setting up visualisations while facilitating different needs and ideas depending on the underlying data and project requirements.

Each script registers a set of arguments using the "argparse" library, these arguments correspond to different parameters. They differ between different scripts, all of them share options for input and output file paths or animation resolution, but not all of them can determine colourmap presets or opacity values, it all depends on complexity of presented visualisations.

Listing 6.1: Example of selected argument.

```
argp.add_argument( "--mode",
    choices=["light", "dark"],
    default="dark",
    help="Choose 'light' or 'dark' mode.",
)
```

Listing 6.1 is an example of argument implementation, it defines mode of visualisation, the developed animations handle different types of user-controllable options, including parameters with predefined choices, Boolean flags or parameters type definition.

In addition to direct execution from the terminal, the visualisation scripts tailored for 2.1 can be launched from a Jupyter Notebook using the Python "subprocess" module. an example of such usage is shown in the listing 6.2.

Listing 6.2: Launch of visualisation script from a Jupyter Notebook

```
result = subprocess.run(
    [
        "pvpython",
```

```
        "--force-offscreen-rendering",
        str(pvanim),
        str(product),
        str(attributes),
        str(output_path),
        "--animationname", "docs_intro_animation.ogv",
        "--mode", "dark",
    ],
    capture_output=True,
    text=True,
    env=SUBPROCESS_ENV,
)
```

6.2 Continuous integration workflows

Continuous integration (CI) workflows were applied to automate the deployment of scientific visualisation in the online documentation of the PySDM project (2.1). The implemented pipeline integrates ParaView executed in headless mode with Python scripts triggered by GitHub Actions, enabling full automation and reproducibility for visualisation updates.

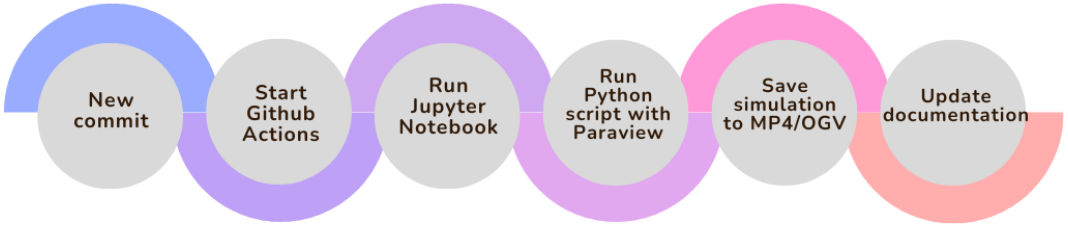


Figure 6.1: Diagram of the continuous integration workflow used to visualise the PySDM 2.1 project.

In the developed workflow presented in the diagram 6.1, a new commit pushed to the repository automatically starts a CI pipeline. This pipeline executes a simulation from Jupyter Notebook, exports simulation data in .VTK format, and then runs a ParaView Python script using "pvpython". As a result, visualisations are generated and deployed into the documentation without manual interaction, ensuring reflection of the current state.

Listing 6.3: CI artifact upload.

```
    - name: animation movie upload
      if: ( ! startsWith(matrix.platform, 'windows-') )
          && contains(matrix.test-suite, env.anim_test-
          suite) && matrix.python-version == env.
          anim_python-version
      uses: actions/upload-artifact@v4
      with:
        name: animation-movie-${{ matrix.platform }}
        if-no-files-found: error
        path: ~/work/_temp/_github_home/figures/*.gif
    - name: tip release upload
      if: github.ref == 'refs/heads/main' && (
          startsWith(matrix.platform, 'ubuntu-') ) &&
          contains(matrix.test-suite, env.anim_test-
          suite) && matrix.python-version == env.
          anim_python-version
      uses: eine/tip@master
      with:
        token: ${{ secrets.GITHUB_TOKEN }}
        files: /github/home/figures/*.gif
```

This workflow (6.1) includes dedicated steps for collecting and publishing visualisation outputs exactly like 6.3. Animation movies and single frames, as in listing 6.3 presented for animation movies, are uploaded as CI artefacts.

Approaching this project with a special workflow eliminates manual post-processing, ensures consistency, and supports reproducible research. Generated animations are automatically updated in the project documentation and made available as CI artefacts, which can also facilitate code review before merging changes.

6.3 ParaView Python - pvpython

The ParaView Python interpreter - pvpython was used to execute visualisation pipelines in headless mode as a part of the CI workflow. This enabled fully script driven rendering and animation generation without using GUI, making the solution tailored for cloud based execution. visualisation logic was designed programatically using pvpython among others, including data loading, filter application, data visual design, and camera configuration. As mentioned in 6.1, the parameters were controlled from command-line arguments, allowing scripts

to be reused. ParaView processed time dependent simulation data directly and created animation frames and videos.

Listing 6.4: Animation export using pvpython.

```
from paraview import simple as pvs

# ...

scene = pvs.GetAnimationScene()
scene.UpdateAnimationUsingDataTimeSteps()
pvs.Render(setup.renderView1)
pvs.SaveAnimation(
    str(pathlib.Path(args.output_path) / args.
        animationname),
    setup.renderView1,
    FrameRate=args.framerate,
)
```

In Listing 6.4 the animation scene is synchronised with simulation time steps to ensure consistency between the numerical data and the rendered frames. The final animation is exported using ParaView's Python API.

7. Summary

This diploma project concentrated on the development and automation of three visualisations for particle-based atmospheric cloud simulations, using cloud computing to ensure reproducibility and continuous development. The main goals were to create visually appealing animations that fit online documentation, to provide adequate data labelling for easier interpretation, and to fully automate the visualisation pipeline (see PySDM examples). These objectives were fulfilled thanks to the integration of ParaView in headless mode (pvpython), Python, Jupyter Notebook, VTK library, and Github Actions. Visualisations were generated based on simulation data produced by two Lagrangian cloud models: PySDM developed at the AGH University of Kraków and UWLCM developed at the University of Warsaw. The simulation data used in chapter 3 and 4 were generated as part of this thesis; while the data used in chapter 5 were provided by the University of Warsaw. The work covered simulations across different scales:

1. Microscale parcel model, visualising droplet activation and condensational growth.

2. 2D prescribed-flow simulation, showing collisional growth and precipitation formation.

3. 3D large-eddy simulation, demonstrating cloud dynamics and microphysical processes in cumulus convection.

The main engineering achievement was the implementation of CI workflow with usage of Github Actions, automatically triggering creation of visualisation every time a new update appears in the PySDM repository, ensuring that documentation always reflect the actual state of the repository. Visualisation scripts created for this project are highly configurable through command-line interfaces, supporting different model rendering depending on user needs, and are designed for seamless execution in cloud environment. The project demonstrates the application of ParaView and Github Actions to automated visualisation pipelines in open-source scientific software.

Appendix A: Code availability and result reproducibility

The purpose of this section is to present the availability of the source code developed within this diploma project and to provide detailed instructions to reproduce the results presented in this thesis. Ensuring reproducibility is an important aspect of scientific research, providing independent verification and allowing for reuse of the scripts.

A.1 PySDM

The source code developed in this project is available in the main PySDM repository: https://github.com/open-atmos/PySDM. This repository includes the CI workflow proposed in 6.2, including scripts created as part of this thesis. The code is structured in a modular and readable manner to facilitate understanding and reuse. The main components to reproduce the results are:

1. Parcel model
   (a) `PySDM/exporters/vtk_exporter_parcel.py` - Exports simulation data to VTK format.
   (b) `examples/PySDM_examples/Strzabala_2025_BEng/paraview.ipynb` - Runs simulation and exports to VTK.
   (c) `examples/PySDM_examples/Strzabala_2025_BEng/paraview_parcel_model.py` - ParaView visualisation script.
   (d) `examples/PySDM_examples/Pyrcel/settings.py` - Simulation configuration and parameters.
   (e) `examples/PySDM_examples/Pyrcel/simulation.py` - PySDM parcel simulation logic.

   These scripts form an integrated workflow, starting from `settings.py` that defines the physical parameters of the simulation that are passed to `simulation.py`, which executes the core parcel model using PySDM. The simulation outputs are then processed by `vtk_exporter_parcel.`

`py`, which converts the data into VTK format. The entire workflow is orchestrated by the Jupyter Notebook `paraview.ipynb` which runs the simulation, triggers the VTK exporter, and executes the ParaView script. The `paraview_parcel_model.py` is launched by the notebook, it loads the VTK files into ParaView, configures the visual representation, and creates the final output.

2. 2D kinematic model

   (a) `examples/PySDM_examples/utils/pvanim.py` - Creates ParaView visualisations.

   (b) `examples/PySDM_examples/_HOWTOs/paraview_hello_world.ipynb` - Runs simulation and exports to VTK.

   (c) `PySDM/exporters/vtk_exporter.py` - VTK exporter.

   The three files form a visualisation pipeline, where `vtk_exporter.py` exports simulation data to VTK format, `pvanim.py` processes these files in ParaView creating updated visualisations and `paraview_hello_world.ipynb` coordinates the entire workflow by running the simulation using analogical files to once mentioned in Parcel model, calls the exporter, and executes the visualisation script.

Both simulations can be run through the mentioned Jupyter Notebooks or command-line. The PySDM framework should be run in a Python-based computational environment. The implementation relies on standard scientific libraries including NumPy, Numba, SciPy, Pint, ChemPy, PyEVTK, ThrustRTC, CURandRTC and ParaView. PySDM can be installed using a pip package manager, with the source code obtained from the online repository. Example simulations were performed using the PySDM-examples package. this environment ensures reproducibility of the results obtained using the PySDM framework.

The code developed as part of this thesis is available under the GNU General Public License v3.0 (GPLv3).

To obtain an identical result, the intended Jupyter Notebook should be run. After the first run and obtaining the data it can be easier to run the visualisation through command-line in an analogical way to the way available in Jupyter Notebook with additional or different arguments, available with explanation in pvpython scripts.

A.2 UWLCM

The source code developed in this project is available in a dedicated repository: https://github.com/igfuw/UWLCM_plotting, while the UWLCM code repository is at: https://github.com/igfuw/UWLCM. This repository contains a script in pvpython for visualisation of realistic and imaginary versions.

The UWLCM model was executed in a high-performance computing (HPC) environment because of the high time consumption on the standard personal computer. Therefore, the datasets used in this work were generated using an HPC environment and provided by the project supervisor. The implementation relies on ParaView library.

The model is operated through the command-line and executed using a dedicated run script `paraview_scripts/paraview_updated_animation.py` that can be found in the plot repository. Given appropriate input data and configuration files, the simulations can be executed using a command in the listing A.1 or a similar suitable for the purpose.

Listing A.1: Command-line input example for UWLCM visualisation.

```
pvpython paraview_updated_animation.py "temp.xmf" ‘
--lowerthreshold 0.0001 ‘
--background 0.67 1 1 ‘
--boxcolor 0.0 0.0 0.0 ‘
--boxsize 1.0 1.0 0.6 ‘
--colormap1 "Black-Body Radiation" ‘
--colormap2 "Rainbow Uniform" ‘
--opacity1 1.0 ‘
--opacity2 0.17 ‘
--logscale1 ‘
--logscale2 ‘
--camera default
```

Appendix B: Poster presented at EuroSciPy

Paraview via Python on Github CI

Authors
Aleksandra Strząbała
Sylwester Arabas

Affiliations
AGH University of Krakow

We present an automated workflow that integrates ParaView with Python scripts executed via Github Actions. This approach enables consistent, reproducible and high-quality visualizations without manual processing supporting reproducible open-science.

Motivation

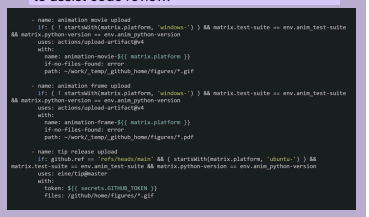
High-performance computing and engineering demand clear, accurate visualizations to interpret complex datasets. Manual processing is error-prone and time-consuming, making reproducibility difficult and risking inconsistencies. Automation solves these challenges. By integrating ParaView in headless mode with GitHub Actions, we ensure that visualizations update automatically with the latest code. This eliminates manual errors, enforces consistency, and guarantees results always matching the newest update.

Tools and technologies

- **ParaView** – Open-source scientific visualization tool.
- **ParaView Python API** – Programming interface for automating ParaView visualizations.
- **GitHub Actions** – CI/CD workflow automation platform.
- **Headless mode (ParaView)** – Enables execution without GUI.
- **pvpython** – Python interpreter for headless ParaView scripting.
- **pyeVTK** – Python library for generating VTK-format files.
- **Jupyter Notebook** – interactive coding environment, with CI ensuring package compatibility for reliable execution from the repository.

Workflow

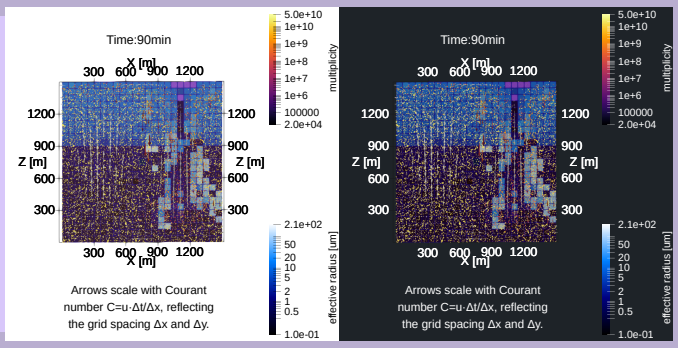
New code commit triggers GitHub Actions to execute ParaView scripts in headless mode. This generates new visualizations that are automatically updated in documentation, with animations available as CI artifacts before merge to assist code review.



New commit → Start Github Actions → Run Jupyter Notebook → Run Python script with ParaView → Save simulation to MP4/PDF → Update documentation

Application — PySDM

PySDM is a package that simulates particle growth dynamics using the high-performance Super-Droplet Method. The workflow processes simulation data via pyeVTK, using a Jupyter notebook to run simulations and a ParaView script to visualize attributes, products and flow of particles. It has two modes and every element is removable and customizable while running the simulation in Jupyter Notebook. This steps are triggered with every push to the repository. Automatically regenerating visualizations in the documentation, ensuring up-to-date results. Making automated visualisations of particle-resolved flow-coupled simulations in the cloud.



Benefits

- Eliminates manual processing
- Maintains visualization consistency
- Enables cloud execution
- Ensures research reproducibility
- Simplifies complex data interpretation
- Supports collaborative science
- Error-free offscreen rendering
- Time-efficient data processing

Future Work

This poster features work included in engineering thesis, that takes on topic of automated 3D visualisations of particle-resolved flow-coupled simulations in the cloud with usage of Python. The work will cover 3D simulation of UWLCM project (the effect is available online - https://github.com/olastrz) and more. We encourage to make a pull request to PySDM and check that simulation and generation of visualization work.

Links

https://open-atmos.github.io/PySDM/
https://github.com/open-atmos/PySDM

Table of symbols

Symbol	Description
A	2.53×10^8 kPa (constant in saturation pressure approximation)
a, b, k	integration limits and exponent for droplet size spectrum moments
B	5.42×10^3 K (constant in saturation pressure approximation)
C	Courant number ($C = u\,\Delta t/\Delta x$)
c_p	specific heat capacity of air at constant pressure
$\mathcal{C}$	condensation term in supersaturation equation
D	molecular diffusion coefficient
$E(r_1, r_2)$	collision efficiency
e	vapour pressure
$e_\mathrm{s}(T)$	saturation vapour pressure at temperature T
$e_\mathrm{s}(T_r)$	saturation vapour pressure over flat water at droplet surface temperature T_r
ε	ratio of gas constants for water vapour and dry air ($R_\mathrm{v}/R' \approx 0.622$)
F_d	term associated with vapour diffusion in droplet growth
F_k	term associated with heat conduction in droplet growth
G_ϕ	Gibbs function for phase ϕ
g	gravitational acceleration
i	droplet size category index
$\mathcal{K}$	thermal conductivity of air
$K(x, y)$	collision kernel (collision rate between droplets of volume x and y)
L	latent heat of vaporisation
M	total droplet mass (when k = 1)
m	mass (general)
m_0	mass of one water molecule
m_i	mass of a droplet in size category i
m_{r_1}	droplet mass with radius r_1
N	number of molecules
N_A	Avogadro's number
n	concentration of water molecules
$\mathcal{N}$	droplet concentration (general)
ν	number of moles (in $P\mathcal{V} = \nu RT$)
$n_r(r)$	number distribution of droplets with radius r
$n(x, t)$	number concentration of droplets of volume x at time t
p	pressure
P	production term in supersaturation equation
Q	heat
Q_1, Q_2	thermodynamic variables in supersaturation equation
q_v	water vapour mixing ratio

r	droplet radius
r_1, r_2	droplet radii (larger and smaller droplet)
$\dot{r}$	rate of change of droplet radius
$\mathcal{R}$	distance from droplet centre
R	universal gas constant
R'	gas constant for dry air
$\mathrm{R_v}$	gas constant for water vapour
RH	relative humidity
S	supersaturation ($S = \mathrm{RH} - 1$)
s	entropy
T	ambient temperature
T_r	droplet surface temperature
T_S	simulation time (used in figure captions)
t	time
u_ϕ	internal energy of phase ϕ
$\mathbf{u}$	air velocity vector (u, w components)
$\mathcal{V}$	volume (in ideal gas law)
$V(r)$	terminal fall speed of droplet with radius r
w	mixing ratio
w_s	saturation mixing ratio
α_1, α_2	specific volumes of liquid and vapour phases
α_ϕ	specific volume of phase ϕ
θ	dry-air potential temperature
$\varrho_\mathrm{v}(\mathcal{R})$	vapour density at distance $\mathcal{R}$ from droplet centre
$\varrho_{\mathrm{v}r}$	vapour density at droplet surface
ρ'	air density
ρ_l	liquid water density
ϕ	phase (e.g., liquid, vapour)
ξ_i	multiplicity (number of real droplets represented by super-droplet i)
$\partial_t m$	rate of change in time
$\nabla \cdot (\mathbf{u}m)$	spatial advection term
$\partial_r(\dot{r}m)$	spectral advection term
$\frac{d\chi}{dt}$	condensation rate (mass of condensate per mass of air per time)
$\frac{dz}{dt}$	vertical air velocity

Acknowledgements

I express my gratitude to my supervisor, dr Sylwester Arabas, for his guidance and support throughout this project. I also thank the open-atmos group, particularly for the development of PySDM. I extend my thanks to dr Piotr Dziekan and prof. dr hab. Hanna Pawłowska from the University of Warsaw for their support during my summer internship. This work was supported by the Polish National Science Centre (grant no. 2020/39/D/ST10/01220) and I acknowledge the funding for participation in EuroSciPy.

Bibliography

[Ait80] J. Aitken. "On Dust, Fogs and Clouds". *Nature* 23 (1880). Abstract of a paper read to the Royal Society of Edinburgh. DOI: 10.1038/023195d0 (cit. on p. 2).

[Ait81] J. Aitken. "Dust, Fogs, and Clouds". 23 (1881). DOI: 10.1038/023384a0 (cit. on p. 2).

[Ara+15] S. Arabas, A. Jaruga, H. Pawlowska, and W. W. Grabowski. "libcloudph++ 1.0: a single-moment bulk, double-moment bulk, and particle-based warm-rain microphysics library in C++". *Geoscientific Model Development* 8.6 (2015). DOI: 10.5194/gmd-8-1677-2015. URL: https://gmd.copernicus.org/articles/8/1677/2015/ (cit. on p. 24).

[Aya15] U. Ayachit. *The ParaView Guide: A Parallel Visualization Application*. Kitware Inc., 2015. ISBN: 1930934300. URL: https://dl.acm.org/doi/abs/10.5555/2789330 (cit. on p. 12).

[Bar+22] P. Bartman, O. Bulenok, K. Górski, A. Jaruga, G. Łazarski, M. A. Olesik, B. Piasecki, C. E. Singer, A. Talar, and S. Arabas. "PySDM v1: particle-based cloud modeling package for warm-rain microphysics and aqueous chemistry". *J. Open Source Software* 7.72 (2022). DOI: 10.21105/joss.03219 (cit. on pp. 2, 11, 12).

[CCH12] R. de la Cruz, H. Calmet, and G. Houzeaux. *Implementing a XDMF/HDF5 Parallel File System in Alya*. PRACE White Paper – I/O. 2012. DOI: 10.5281/zenodo.6241986 (cit. on pp. 14, 15).

[DWP19] P. Dziekan, M. Waruszewski, and H. Pawlowska. "University of Warsaw Lagrangian Cloud Model (UWLCM) 1.0: a modern large-eddy simulation tool for warm cloud modeling with Lagrangian microphysics". *Geosci. Model Dev.* 12.6 (2019). DOI: 10.5194/gmd-12-2587-2019 (cit. on pp. 2, 11, 12, 24).

[DZ22] P. Dziekan and P. Zmijewski. "University of Warsaw Lagrangian Cloud Model (UWLCM) 2.0: adaptation of a mixed Eulerian–Lagrangian numerical model for heterogeneous computing clusters". *Geosci. Model Dev.* 15.11 (2022). DOI: 10.5194/gmd-15-4489-2022 (cit. on p. 24).

[Fol+11] M. Folk, G. Heber, Q. Koziol, E. Pourmal, and D. Robinson. "An overview of the HDF5 technology suite and its applications". *Proc. EDBT/ICDT 2011 workshop on array databases*. 2011. DOI: 10.1145/1966895.1966900 (cit. on p. 14).

[Jon+23] E. K. de Jong, C. E. Singer, S. Azimi, P. Bartman, O. Bulenok, K. Derlatka, I. Dula, A. Jaruga, J. B. Mackay, R. X. Ward, and S. Arabas. "New developments in PySDM and PySDM-examples v2: collisional breakup, immersion freezing, dry aerosol initialization, and adaptive time-stepping". *J. Open Source Software* 8.84 (2023). DOI: 10.21105/joss.04968 (cit. on pp. 19, 21).

[Kes69] E. Kessler. “On the Distribution and Continuity of Water Substance in Atmospheric Circulations”. *Meteorological Monographs*. Vol. 10. Boston, MA: American Meteorological Society, 1969. DOI: 10.1007/978-1-935704-36-2_1 (cit. on p. 19).

[Kin+21] T. Kinsman, M. Wessel, M. A. Gerosa, and C. Treude. “How do software developers use github actions to automate their workflows?” *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*. IEEE. 2021. DOI: 10.1109/MSR52588.2021.00054 (cit. on p. 15).

[Kit24] Kitware Inc. *ParaView - Open-source, multi-platform data analysis and visualization application*. 2024. URL: https://www.paraview.org (visited on 12/15/2025) (cit. on pp. 12, 13).

[Mor+20] H. Morrison et al. “Confronting the Challenge of Modeling Cloud and Precipitation Microphysics”. *J. Adv. Model. Earth Sys.* 12.8 (2020). DOI: 10.1029/2019MS001689 (cit. on pp. 2, 7, 8, 10, 11).

[Mor18] K. Moreland. *The ParaView Tutorial*. Version 5.6. Sandia National Laboratories. 2018. URL: https://www.mn.uio.no/astro/english/services/it/help/visualization/paraview/paraviewtutorial-5.6.0.pdf (visited on 12/15/2025) (cit. on p. 13).

[Paw23] H. Pawlowska. “Mikrofizyka chmur”. *Postępy Fizyki* 74 (2023). Access: 2025-10-18. URL: https://www.ptf.net.pl/sites/default/files/PF/PF_4_2023.pdf (cit. on pp. 2, 7, 8).

[Ran+19] D. Randall, C. Bitz, G. Danabasoglu, S. Denning, P. Gent, A. Gettelman, S. Griffies, P. Lynch, H. Morrison, R. Pincus, and J. Thuburn. “100 Years of Earth System Model Development”. *Meteorol. Monogr.* 59 (2019). DOI: 10.1175/AMSMONOGRAPHS-D-18-0018.1 (cit. on p. 8).

[RY96] R. R. Rogers and M. K. Yau. *A Short Course in Cloud Physics*. Third edition. Butterworth-Heinemann, 1996. URL: https://archive.org/details/shortcourseinclo0000roge_m3k2 (cit. on pp. 2, 3, 6, 16).

[Shi+09] S. Shima, K. Kusano, A. Kawano, T. Sugiyama, and S. Kawahara. “The super-droplet method for the numerical simulation of clouds and precipitation: a particle-based and probabilistic microphysics model coupled with a non-hydrostatic model”. *Q. J. R. Meteorol. Soc.* 135.642 (2009). DOI: 10.1002/qj.441 (cit. on p. 12).

[Sie+03] A. Siebesma, C. Bretherton, A. Brown, A. Chlond, J. Cuxart, P. Duynkerke, H. Jiang, M. Khairoutdinov, D. Lewellen, C.-H. Moeng, E. Sanchez, B. Stevens, L. And, and D. Stevens. “A Large Eddy Simulation Intercomparison Study of Shallow Cumulus Convection”. *J. Atmos. Sci.* 60 (2003). DOI: 10.1175/1520-0469(2003)60<1201:ALESIS>2.0.CO;2 (cit. on p. 24).

[SML06] W. Schroeder, K. Martin, and B. Lorensen. *The Visualization Toolkit (4th ed.)* Kitware, 2006. ISBN: 978-1-930934-19-1 (cit. on p. 14).