PySDM: a novel Pythonic tool for modelling cloud microphysics

Sylwester Arabas[1,2]

Stony Brook University, New York, June 7 2022

[1]Atmospheric Sciences, University of Illinois at Urbana-Champaign (atmos.illinois.edu)
[2]Computer Science, Jagiellonian University (atmos.ii.uj.edu.pl)

PySDM: a novel Pythonic tool for modelling cloud microphysics

Sylwester Arabas[1,2]

co-authors, contributors & collaborators:
@Jagiellonian: P. Bartman, M. Olesik, G. Łazaski, O. Bulenok...
@Caltech: E. de Jong, C. Singer, A. Jaruga, B. Mackay, I. Dula, S. Azimi ...
@UIUC: N. Riemer, M. West & J. Curtis

[1]Atmospheric Sciences, University of Illinois at Urbana-Champaign (atmos.illinois.edu)
[2]Computer Science, Jagiellonian University (atmos.ii.uj.edu.pl)

Univ. Illinois Urbana-Champaign

Jagiellonian University, Kraków, Poland

Univ. Illinois Urbana-Champaign

- est. 1867 (Morrill Land-Grant Act)

Jagiellonian University, Kraków, Poland

Univ. Illinois Urbana-Champaign

- est. 1867 (Morrill Land-Grant Act)
- ca. 55k students, 10k staff (2.5k acad.)

Jagiellonian University, Kraków, Poland

Univ. Illinois Urbana-Champaign

- est. 1867 (Morrill Land-Grant Act)
- ca. 55k students, 10k staff (2.5k acad.)
- 40 years of Atmospheric Sciences Dept

Jagiellonian University, Kraków, Poland

Univ. Illinois Urbana-Champaign

- est. 1867 (Morrill Land-Grant Act)
- ca. 55k students, 10k staff (2.5k acad.)
- 40 years of Atmospheric Sciences Dept
- birth place of LLVM
  (as well as MRI, LED & Mosaic!)

Jagiellonian University, Kraków, Poland

Univ. Illinois Urbana-Champaign

- est. 1867 (Morrill Land-Grant Act)
- ca. 55k students, 10k staff (2.5k acad.)
- 40 years of Atmospheric Sciences Dept
- birth place of LLVM
  (as well as MRI, LED & Mosaic!)

Jagiellonian University, Kraków, Poland

- est.1364, (among 20 oldest in operation)

Univ. Illinois Urbana-Champaign

- est. 1867 (Morrill Land-Grant Act)
- ca. 55k students, 10k staff (2.5k acad.)
- 40 years of Atmospheric Sciences Dept
- birth place of LLVM
  (as well as MRI, LED & Mosaic!)

Jagiellonian University, Kraków, Poland

- est.1364, (among 20 oldest in operation)
- ca. 40k students, 7k staff (4k acad.)

Univ. Illinois Urbana-Champaign

- est. 1867 (Morrill Land-Grant Act)
- ca. 55k students, 10k staff (2.5k acad.)
- 40 years of Atmospheric Sciences Dept
- birth place of LLVM
  (as well as MRI, LED & Mosaic!)

Jagiellonian University, Kraków, Poland

- est.1364, (among 20 oldest in operation)
- ca. 40k students, 7k staff (4k acad.)
- American Studies since 1991

Univ. Illinois Urbana-Champaign

- est. 1867 (Morrill Land-Grant Act)
- ca. 55k students, 10k staff (2.5k acad.)
- 40 years of Atmospheric Sciences Dept
- birth place of LLVM
  (as well as MRI, LED & Mosaic!)

Jagiellonian University, Kraków, Poland

- est.1364, (among 20 oldest in operation)
- ca. 40k students, 7k staff (4k acad.)
- American Studies since 1991
- 1917 Smoluchowski elected as Rector
  (professor since 1913)

Plan of the talk

PySDM: context

PySDM: statement of need & goals

PySDM: demo (role play: reviewer)

PySDM: summary of key features

context: aerosol-cloud-precipitation interactions (scales!)

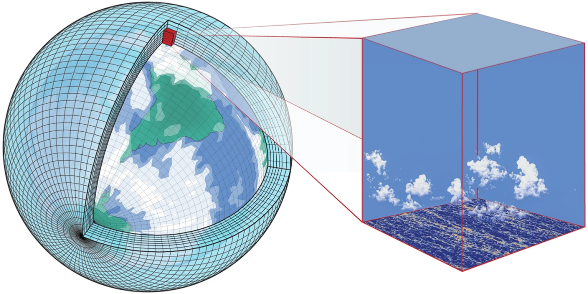
“Cloud and ship. Ukraine, Crimea, Black sea, view from Ai-Petri mountain”

(photo: Yevgen Timashov / National Geographic)

context: aerosol-cloud-precipitation interactions (scales!)

"Cloud and ship. Ukraine, Crimea, Black sea, view from Ai-Petri mountain"

(photo: Yevgen Timashov / National Geographic)



"Grid cells in a global climate model and a large-eddy simulation of shallow cumulus clouds at 5 m resolution"

(fig. from Schneider et al. 2017)

context: aerosol-cloud-precipitation interactions (uncertainty!)

context: aerosol-cloud-precipitation interactions (uncertainty!)

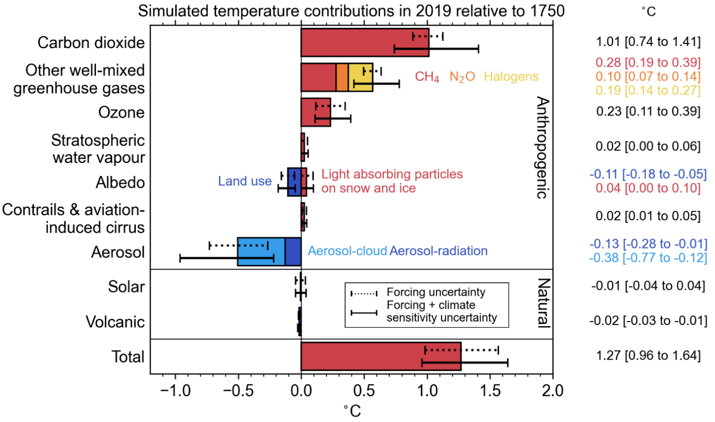
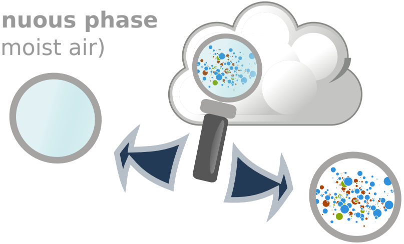


Figure 7.7: The contribution of forcing agents to 2019 temperature change relative to 1750 produced using the two-layer emulator (Supplementary Material 7.SM.2), constrained to assessed ranges for key climate metrics described in Cross-Chapter Box 7.1.

modelling cloud μ-physics: Eulerian vs. Lagrangian approaches

modelling cloud μ-physics: Eulerian vs. Lagrangian approaches

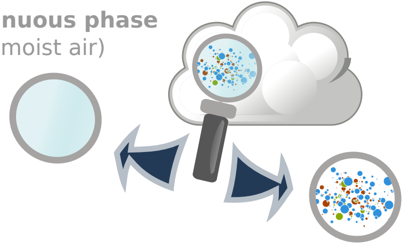
continuous phase
(moist air)



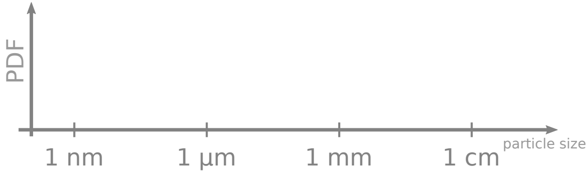
dispersed phase
(aerosol particles, cloud droplets, drizzle, rain, snow, ...)

modelling cloud μ-physics: Eulerian vs. Lagrangian approaches

continuous phase
(moist air)

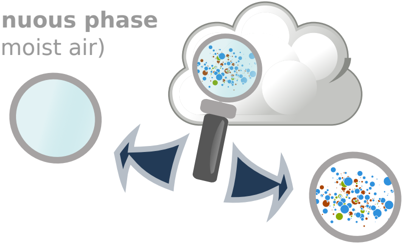


dispersed phase
(aerosol particles, cloud droplets, drizzle, rain, snow, ...)

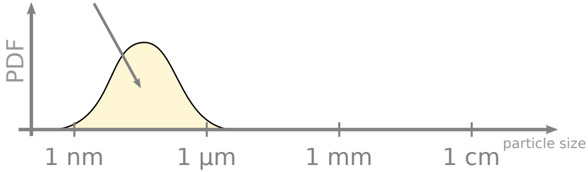


modelling cloud μ-physics: Eulerian vs. Lagrangian approaches

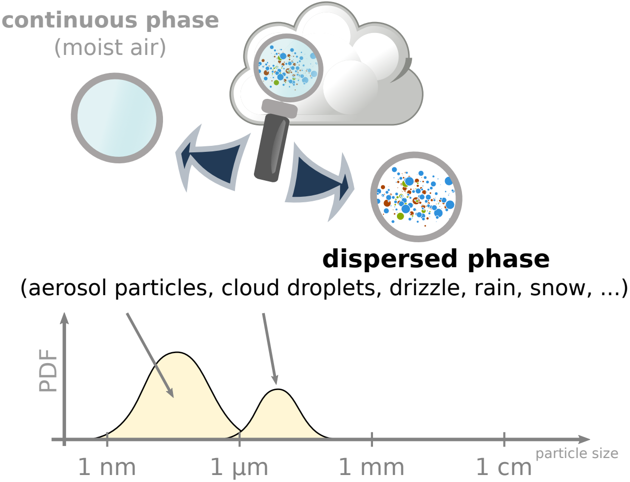
continuous phase
(moist air)



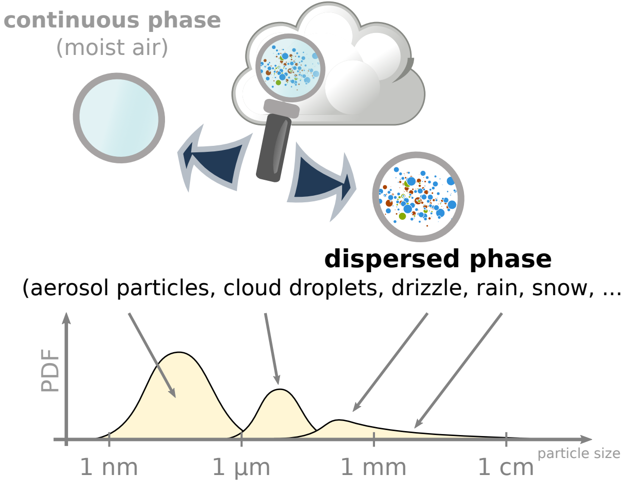
dispersed phase
(aerosol particles, cloud droplets, drizzle, rain, snow, ...)



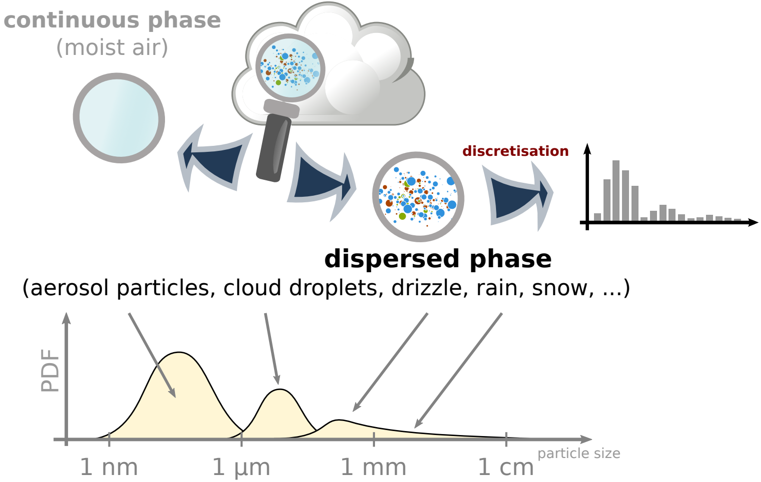
modelling cloud μ-physics: Eulerian vs. Lagrangian approaches

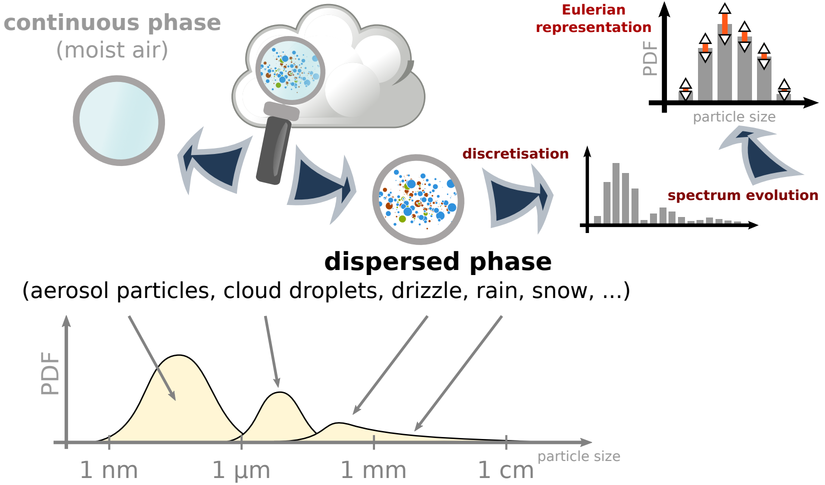


modelling cloud µ-physics: Eulerian vs. Lagrangian approaches



modelling cloud μ-physics: Eulerian vs. Lagrangian approaches




continuous phase
(moist air)
Eulerian
representation
PDF
particle size
discretisation
spectrum evolution
dispersed phase
(aerosol particles, cloud droplets, drizzle, rain, snow, ...)
PDF
1 nm
1 µm
1 mm
1 cm
particle size

Smoluchowski's coagulation equation (SCE)

concentration of particles of size x at time t: $c(x, t)$: $\mathbb{R}^+ \times \mathbb{R}^+ \to \mathbb{R}^+$
collision kernel: $a(x_1, x_2)$: $\mathbb{R}^+ \times \mathbb{R}^+ \to \mathbb{R}^+$

Smoluchowski's coagulation equation (SCE)

concentration of particles of size x at time t: $c(x,t)\colon \mathbb{R}^+ \times \mathbb{R}^+ \to \mathbb{R}^+$
collision kernel: $a(x_1, x_2)\colon \mathbb{R}^+ \times \mathbb{R}^+ \to \mathbb{R}^+$

$$\dot{c}(x) = \frac{1}{2} \int_0^x a(y, x-y) c(y) c(x-y) dy - \int_0^\infty a(y, x) c(y) c(x) dy \tag{1}$$

Smoluchowski's coagulation equation (SCE)

concentration of particles of size x at time t: $c(x,t)\colon \mathbb{R}^+ \times \mathbb{R}^+ \to \mathbb{R}^+$
collision kernel: $a(x_1, x_2)\colon \mathbb{R}^+ \times \mathbb{R}^+ \to \mathbb{R}^+$

$$\dot{c}(x) = \frac{1}{2}\int_0^x a(y, x-y)c(y)c(x-y)dy - \int_0^\infty a(y,x)c(y)c(x)dy \tag{1}$$

discretised particle concentration: $c_i = c(x_i)$ where $x_i = i \cdot x_0$

$$\dot{c}_i = \frac{1}{2}\sum_{k=1}^{i-1} a(x_k, x_{i-k})c_k c_{i-k} - \sum_{k=1}^{\infty} a(x_k, x_i)c_k c_i \tag{2}$$

cloud droplet collisional growth

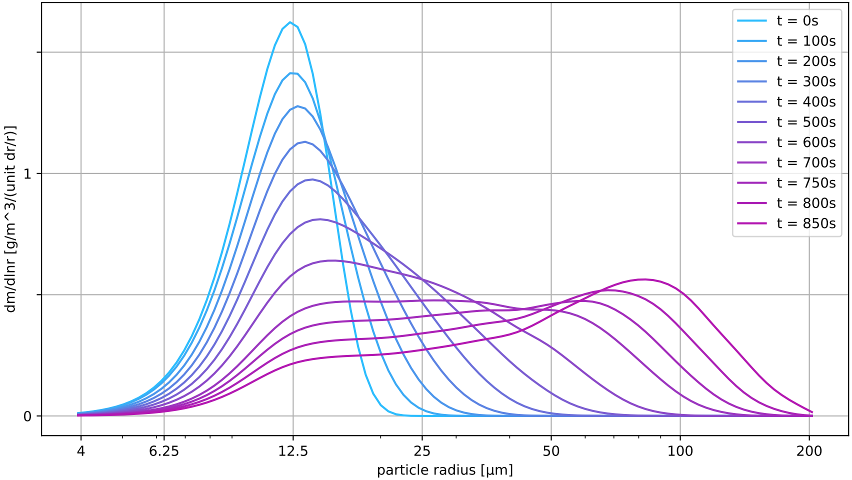


figure (PySDM simulation): Bartman, Arabas et al. 2021, LNCS (doi:10.1007/978-3-030-77964-1_2)

SCE: challenges/problems

- analytic solutions known only for simple kernels

SCE: challenges/problems

- analytic solutions known only for simple kernels
- numerical methods suffer from the curse of dimensionality
  when distinguishing particles of same size but different properties

SCE: challenges/problems

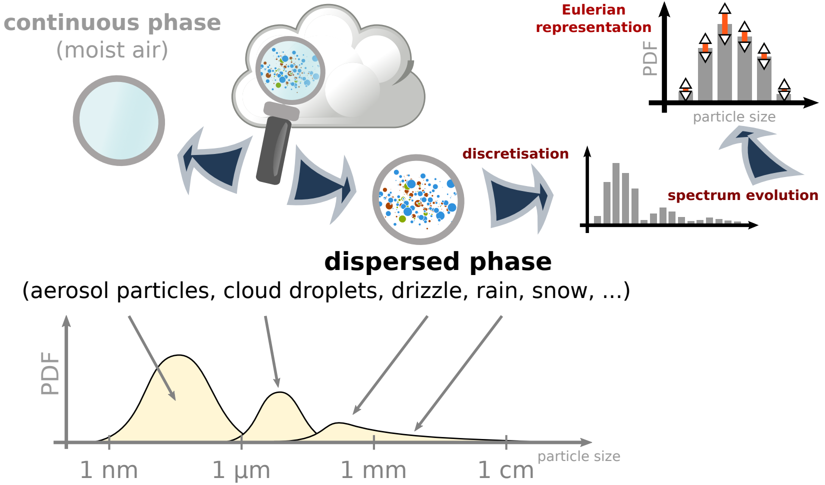
- analytic solutions known only for simple kernels
- numerical methods suffer from the curse of dimensionality
  when distinguishing particles of same size but different properties
- assumptions behind SCE difficult to meet in practice, e.g.:

SCE: challenges/problems

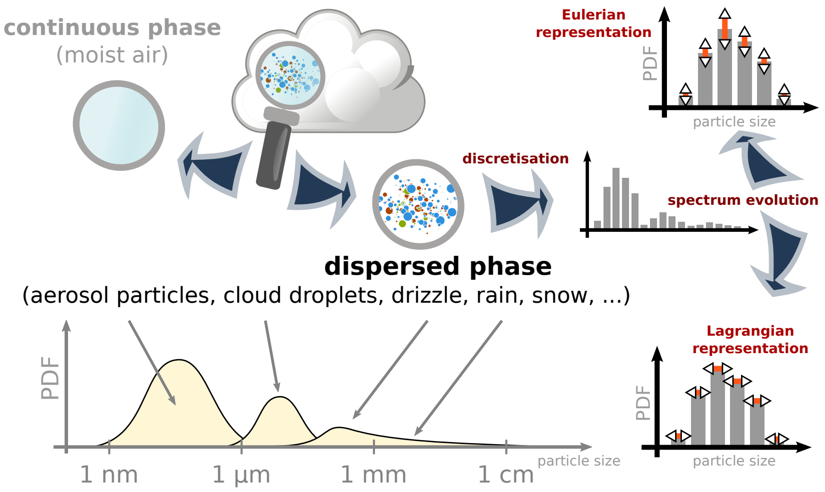
- analytic solutions known only for simple kernels
- numerical methods suffer from the curse of dimensionality
  when distinguishing particles of same size but different properties
- assumptions behind SCE difficult to meet in practice, e.g.:
  it is assumed that the system is large enough and the droplets inside are uniformly distributed, which in turn is only true for a small volume in the atmosphere

SCE: challenges/problems

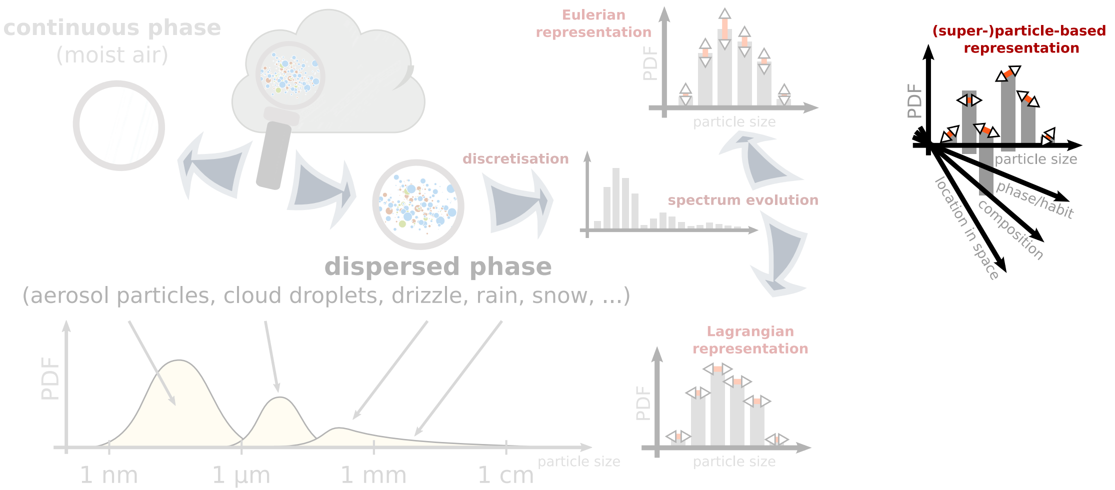
- analytic solutions known only for simple kernels
- numerical methods suffer from the curse of dimensionality
  when distinguishing particles of same size but different properties
- assumptions behind SCE difficult to meet in practice, e.g.:
  it is assumed that the system is large enough and the droplets inside are uniformly distributed, which in turn is only true for a small volume in the atmosphere
- ...


continuous phase
(moist air)
Eulerian
representation
PDF
particle size
discretisation
spectrum evolution
dispersed phase
(aerosol particles, cloud droplets, drizzle, rain, snow, ...)
PDF
1 nm
1 μm
1 mm
1 cm
particle size

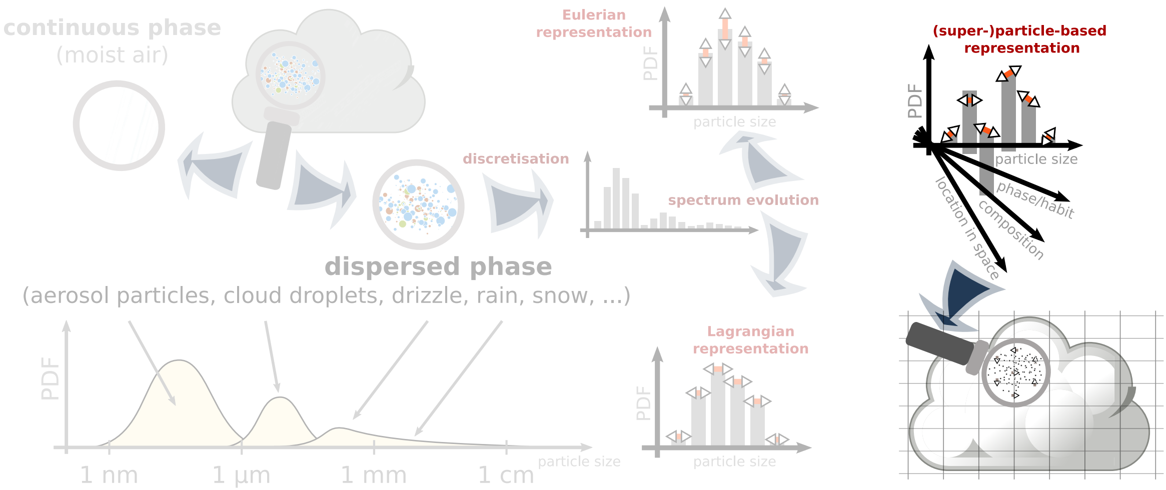
modelling cloud μ-physics: Eulerian vs. Lagrangian approaches



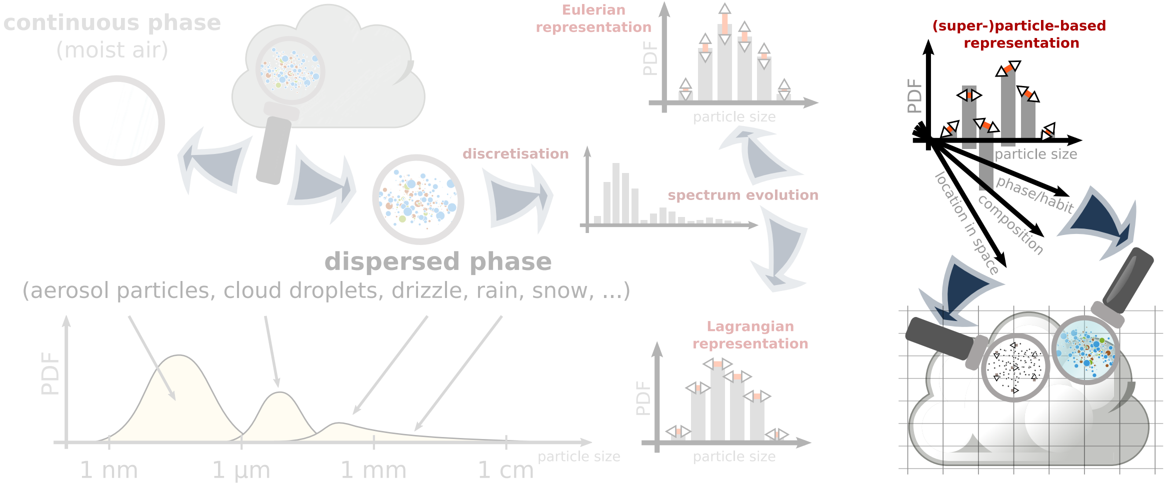
modelling cloud μ-physics: Eulerian vs. Lagrangian approaches



modelling cloud μ-physics: Eulerian vs. Lagrangian approaches



modelling cloud μ-physics: Eulerian vs. Lagrangian approaches



Monte-Carlo SCE alternatives: e.g., SDM by Shima et al.

Shima et al. 2009 (doi:10.1002/qj.441): warm-rain

Monte-Carlo SCE alternatives: e.g., SDM by Shima et al.

Shima et al. 2009 (doi:10.1002/qj.441): warm-rain

Shima et al. 2020 (doi:10.5194/gmd-13-4107-2020): mixed-phase

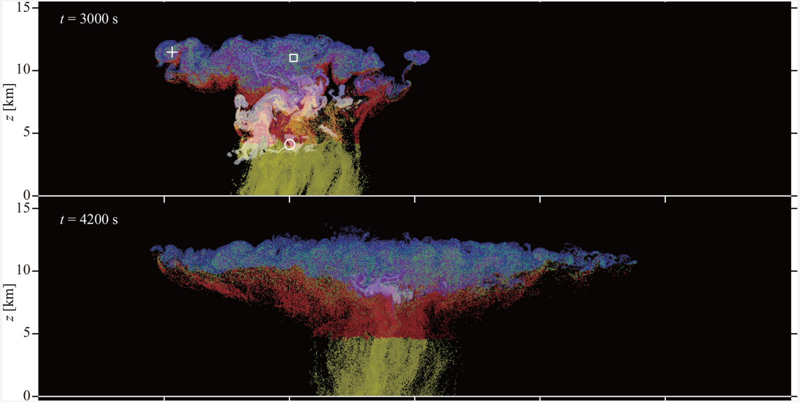


Figure 1. Typical realization of CTRL cloud spatial structures at $t = 2040$, 2460, 3000, 4200, and 5400 s. The mixing ratio of cloud water, rainwater, cloud ice, graupel, and snow aggregates are plotted in fading white, yellow, blue, red, and green, respectively. The symbols indicate examples of unrealistic predicted ice particles (Sects. 7.3 and 9.1). See also Movie 1 in the video supplement.

Super Droplet Method vs. SCE: differences

SCE (naïve impl)	SDM
method type	
mean-field, deterministic	Monte-Carlo, stochastic

Super Droplet Method vs. SCE: differences

SCE (naïve impl)	**SDM**
method type	
mean-field, deterministic	Monte-Carlo, stochastic
considered pairs	
all (i,j) pairs	random set of $n_{sd}/2$ non-overlapping pairs, probability up-scaled by $(n_{sd}^2 - n_{sd})/2$ to $n_{sd}/2$ ratio

Super Droplet Method vs. SCE: differences

SCE (naïve impl)	SDM
method type	
mean-field, deterministic	Monte-Carlo, stochastic
considered pairs	
all (i,j) pairs	random set of $n_{sd}/2$ non-overlapping pairs, probability up-scaled by $(n_{sd}^2 - n_{sd})/2$ to $n_{sd}/2$ ratio
computation complexity	
$\mathcal{O}(n_{sd}^2)$	$\mathcal{O}(n_{sd})$

Super Droplet Method vs. SCE: differences

SCE (naïve impl)	**SDM**
method type	
mean-field, deterministic	Monte-Carlo, stochastic
considered pairs	
all (i,j) pairs	random set of $n_{sd}/2$ non-overlapping pairs, probability up-scaled by $(n_{sd}^2 - n_{sd})/2$ to $n_{sd}/2$ ratio
computation complexity	
$\mathcal{O}(n_{sd}^2)$	$\mathcal{O}(n_{sd})$
collisions triggered	
every time step	by comparing probability with a random number

Super Droplet Method vs. SCE: differences

SCE (naïve impl)	SDM
method type	
mean-field, deterministic	Monte-Carlo, stochastic
considered pairs	
all (i,j) pairs	random set of $n_{sd}/2$ non-overlapping pairs, probability up-scaled by $(n_{sd}^2 - n_{sd})/2$ to $n_{sd}/2$ ratio
computation complexity	
$\mathcal{O}(n_{sd}^2)$	$\mathcal{O}(n_{sd})$
collisions triggered	
every time step	by comparing probability with a random number
collisions	
colliding a fraction of $\xi_{[i]}$, $\xi_{[j]}$	collide all of $\min\{\xi_{[i]}, \xi_{[j]}\}$ ("all or nothing")

Super Droplet Method vs. SCE: differences

SCE (naïve impl)	SDM
method type	
mean-field, deterministic	Monte-Carlo, stochastic
considered pairs	
all (i,j) pairs	random set of $n_{sd}/2$ non-overlapping pairs, probability up-scaled by $(n_{sd}^2 - n_{sd})/2$ to $n_{sd}/2$ ratio
computation complexity	
$\mathcal{O}(n_{sd}^2)$	$\mathcal{O}(n_{sd})$
collisions triggered	
every time step	by comparing probability with a random number
collisions	
colliding a fraction of $\xi_{[i]}$, $\xi_{[j]}$	collide all of $\min\{\xi_{[i]}, \xi_{[j]}\}$ ("all or nothing")
interpretation	
concentration "c_i" in size bin "i"	besides c_i, each "particle" i carries other physicochemical attributes, e.g. position (x_i, y_i, z_i)

SDM

PySDM

Plan of the talk

PySDM: context

PySDM: statement of need & goals

PySDM: demo (role play: reviewer)

PySDM: summary of key features

PySDM: goals

Develop an implementation of the SDM algorithm:

PySDM: goals

Develop an implementation of the SDM algorithm:

- applicable in research on aerosol-cloud-interactions (and beyond)
  KPI: reproduction of results from classic and recent literature

PySDM: goals

Develop an implementation of the SDM algorithm:

- applicable in research on aerosol-cloud-interactions (and beyond)
  KPI: reproduction of results from classic and recent literature

- **easy to reuse**: code (Python), examples (Jupyter), extensibility (modular, high test coverage),
  interoperability (other languages, i/o),
  leveraging modern hardware (GPUs, multi-core CPUs)
  KPI: user feedback & contributions

PySDM: goals

Develop an implementation of the SDM algorithm:

- applicable in research on aerosol-cloud-interactions (and beyond)
  KPI: reproduction of results from classic and recent literature
- **easy to reuse**: code (Python), examples (Jupyter), extensibility (modular, high test coverage), interoperability (other languages, i/o),
  leveraging modern hardware (GPUs, multi-core CPUs)
  KPI: user feedback & contributions
- **accessibility**: seamless Linux/macOS/Windows installation (pip)
  KPI: continuous integration on all targeted platforms

PySDM: goals

Develop an implementation of the SDM algorithm:

- applicable in research on aerosol-cloud-interactions (and beyond)
  KPI: reproduction of results from classic and recent literature
- **easy to reuse**: code (Python), examples (Jupyter), extensibility (modular, high test coverage), interoperability (other languages, i/o), leveraging modern hardware (GPUs, multi-core CPUs)
  KPI: user feedback & contributions
- **accessibility**: seamless Linux/macOS/Windows installation (pip)
  KPI: continuous integration on all targeted platforms
- **curation**: open licensing (GPL), public versioned development (Github)
  KPI: instant and anonymous execution on commodity environment

PySDM: 2D kinematic Sc test (Morrison & Grabowski '07)

2D flow field

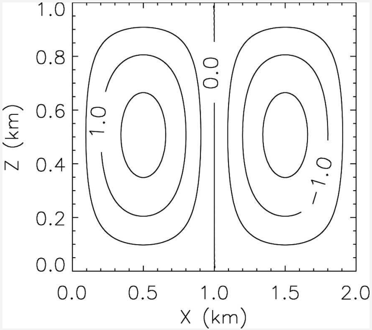
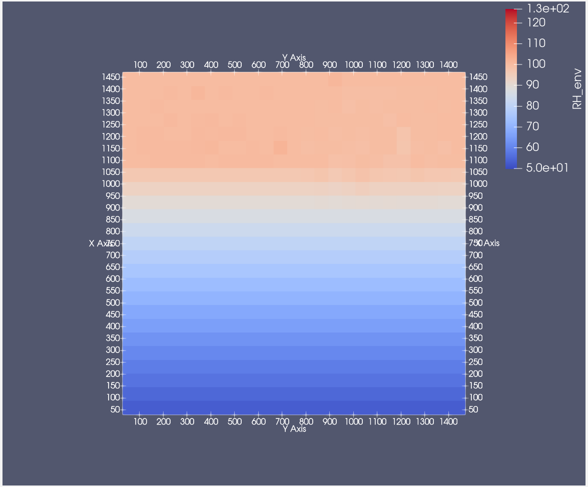
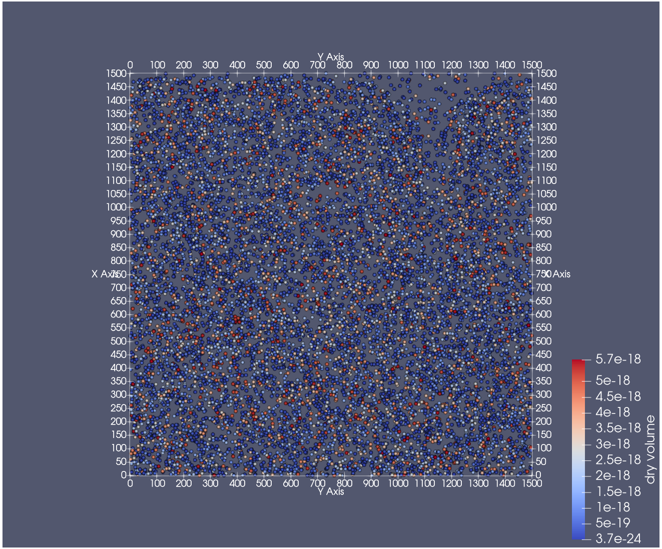


FIG. 1. Time-invariant vertical velocity for the stratocumulus case (contour interval is 0.5 m s^{-1}).

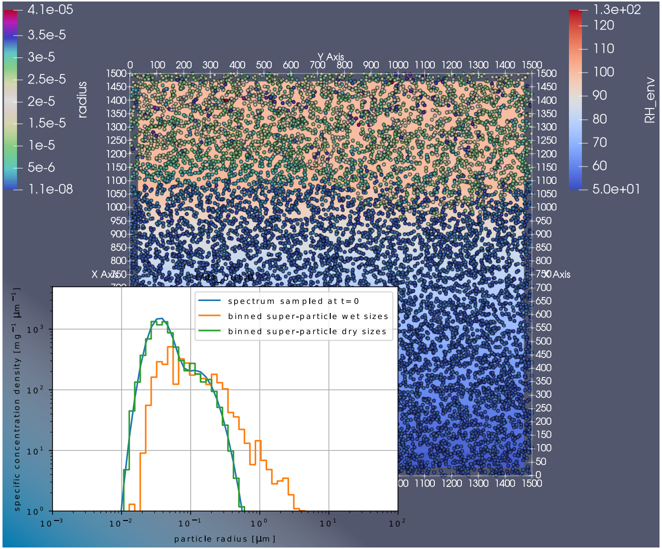
RH profile at t=0



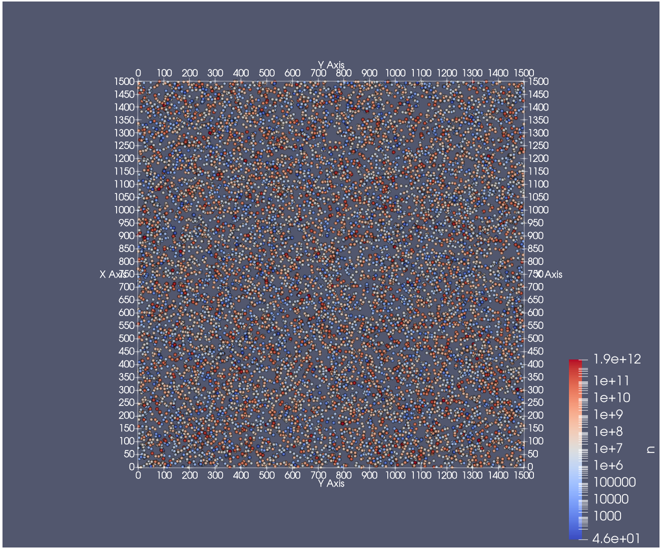
particle attribute initialisation: dry/wet volume



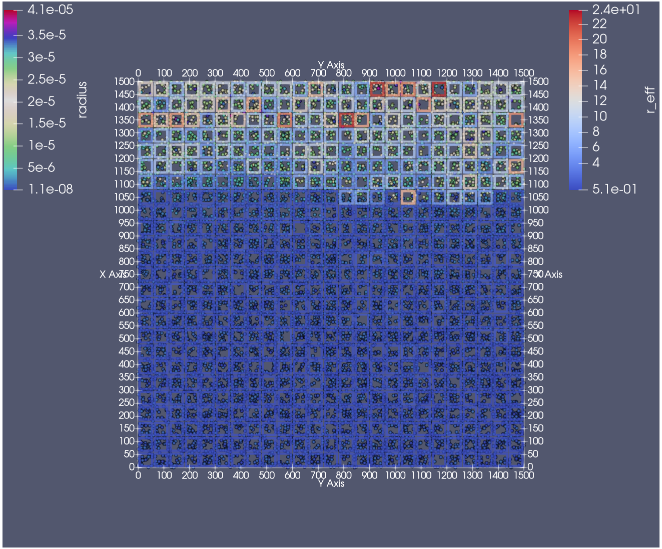
particle attribute initialisation: dry/wet volume



particle attribute initialisation: multiplicity

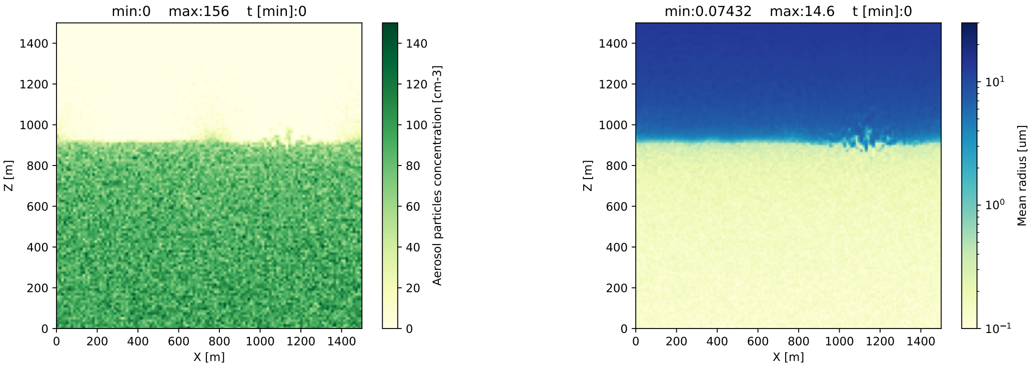


particle attribute evolution: droplet radius



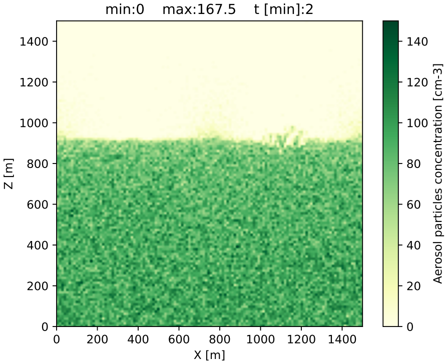
sample aerosol-cloud-precipitation interactions simulation

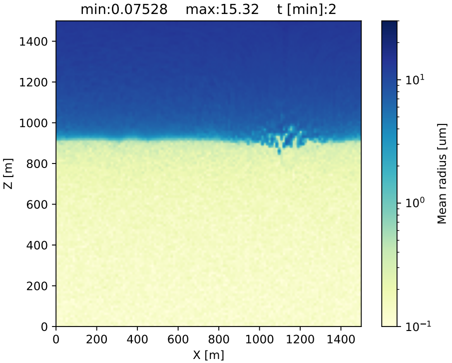
Computational grid: 128x128
Computational particles: 2^{21}



Simulation & visualisation: Piotr Bartman

Computational grid: 128x128
Computational particles: 2^{21}

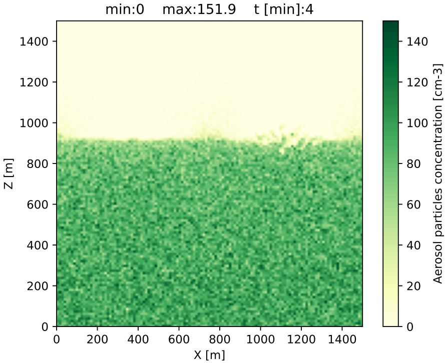


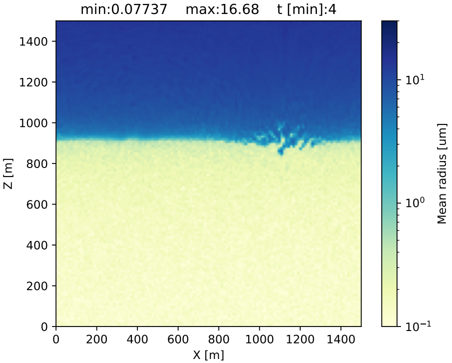


Simulation & visualisation: Piotr Bartman

sample aerosol-cloud-precipitation interactions simulation

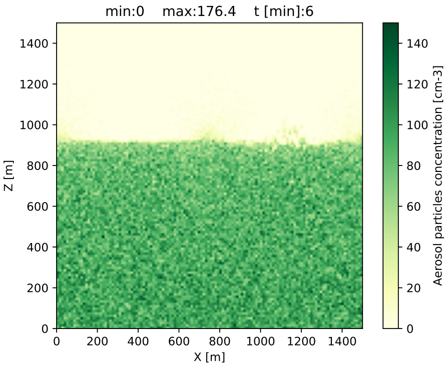
Computational grid: 128x128
Computational particles: 2^{21}

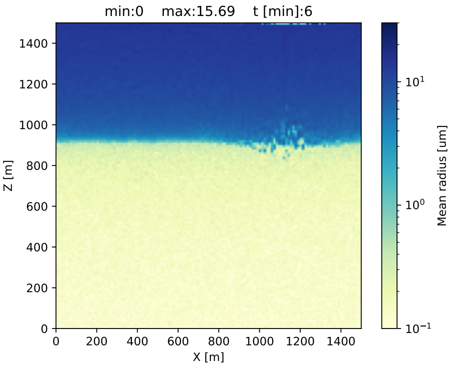




Simulation & visualisation: Piotr Bartman

Computational grid: 128x128
Computational particles: 2^{21}

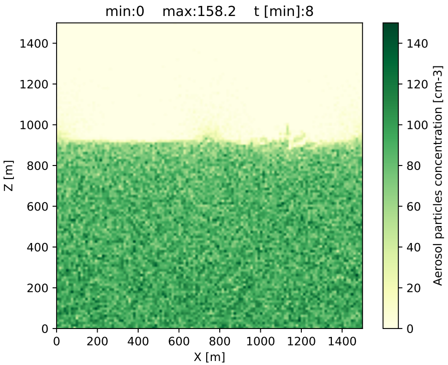


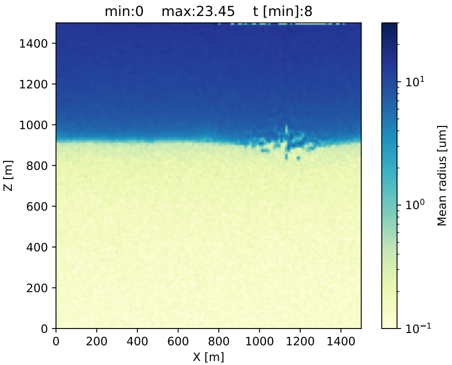


Simulation & visualisation: Piotr Bartman

sample aerosol-cloud-precipitation interactions simulation

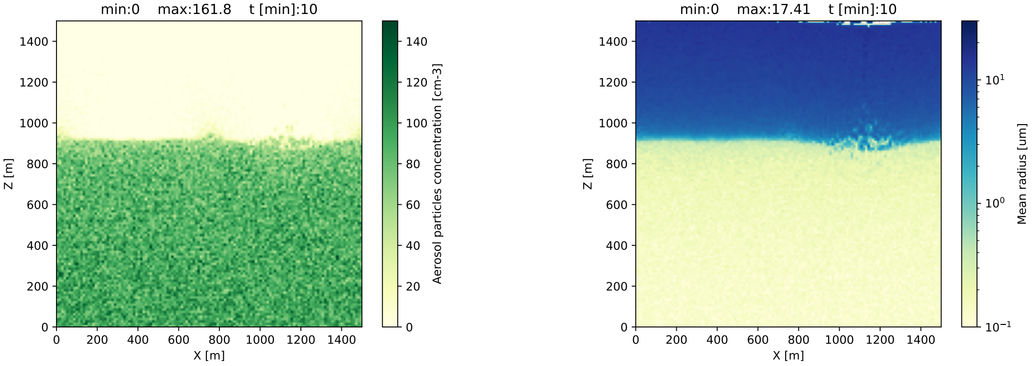
Computational grid: 128x128
Computational particles: 2^{21}





Simulation & visualisation: Piotr Bartman

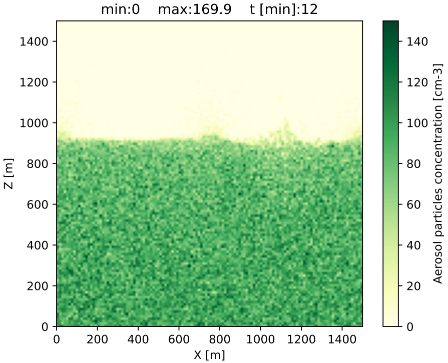
Computational grid: 128x128
Computational particles: 2^{21}

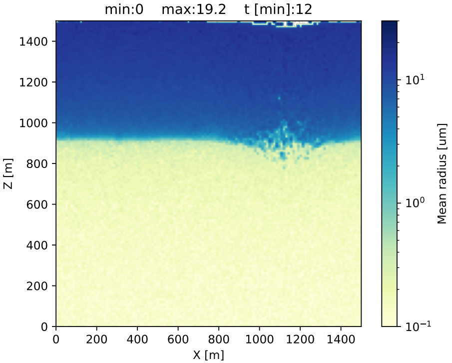


Simulation & visualisation: Piotr Bartman

sample aerosol-cloud-precipitation interactions simulation

Computational grid: 128x128
Computational particles: 2^{21}

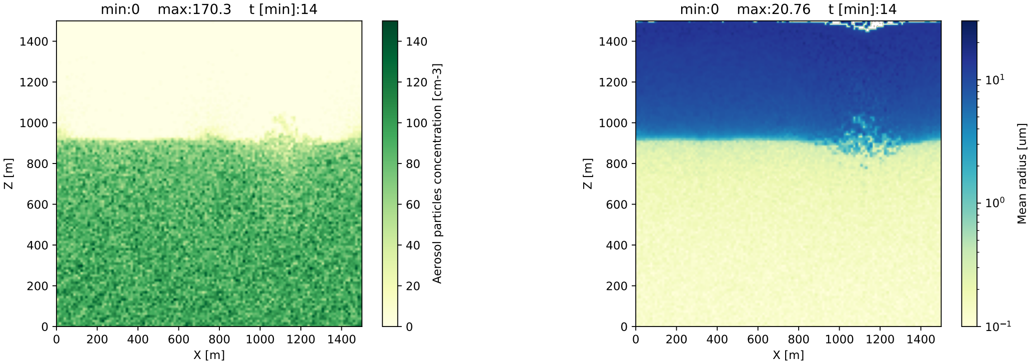




Simulation & visualisation: Piotr Bartman

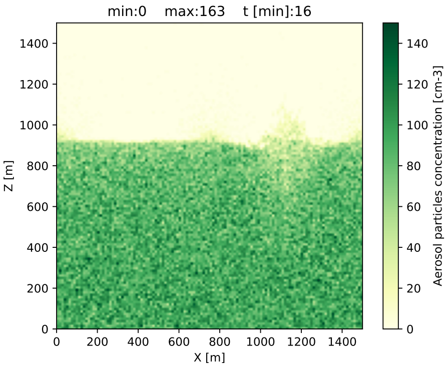
sample aerosol-cloud-precipitation interactions simulation

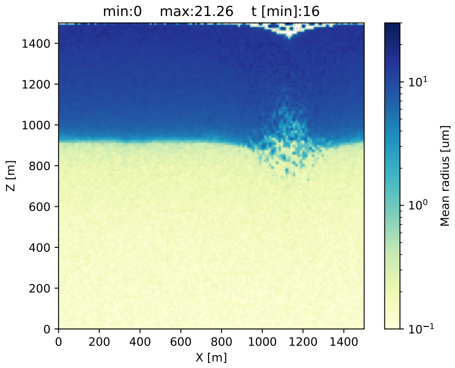
Computational grid: 128x128
Computational particles: 2^{21}



Simulation & visualisation: Piotr Bartman

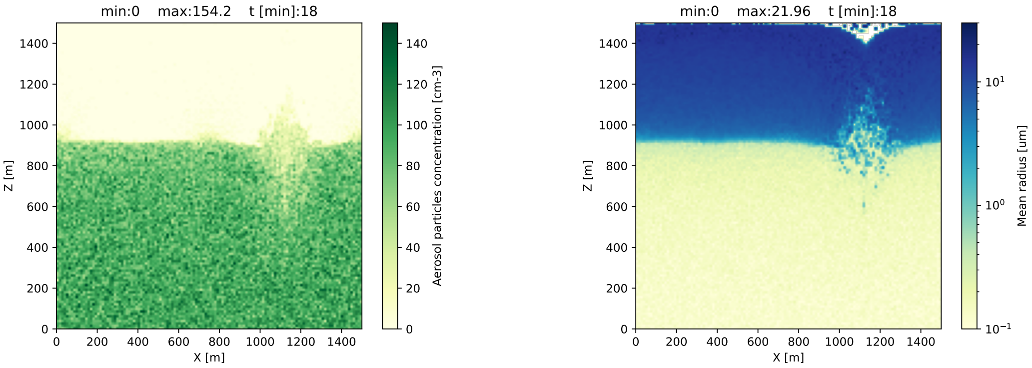
Computational grid: 128x128
Computational particles: 2^{21}





Simulation & visualisation: Piotr Bartman

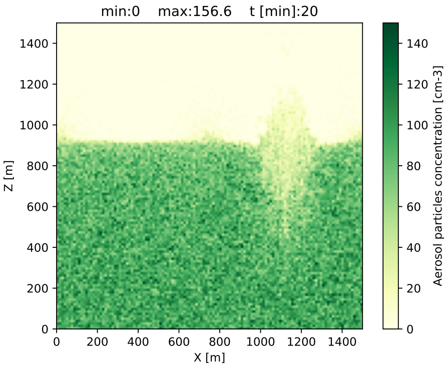
Computational grid: 128x128
Computational particles: 2^{21}

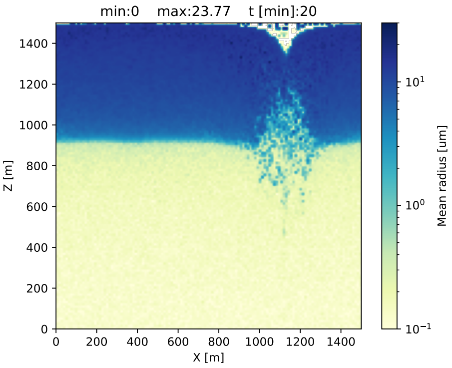


Simulation & visualisation: Piotr Bartman

sample aerosol-cloud-precipitation interactions simulation

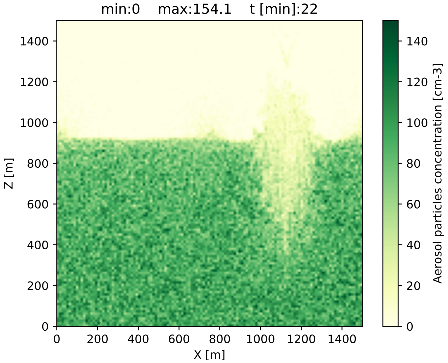
Computational grid: 128x128
Computational particles: 2^{21}

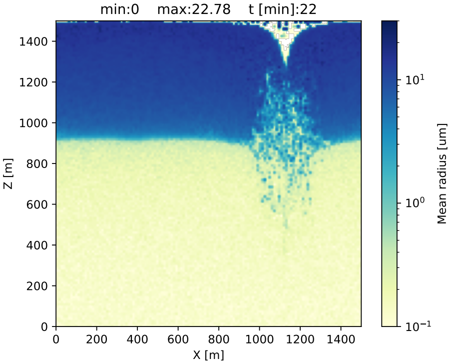




Simulation & visualisation: Piotr Bartman

Computational grid: 128x128
Computational particles: 2^{21}

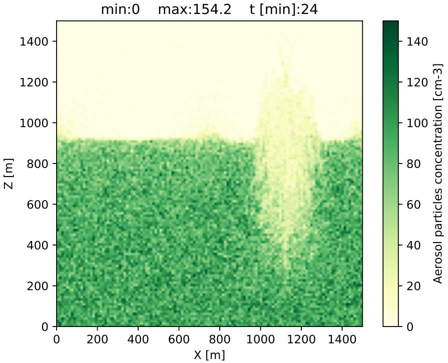


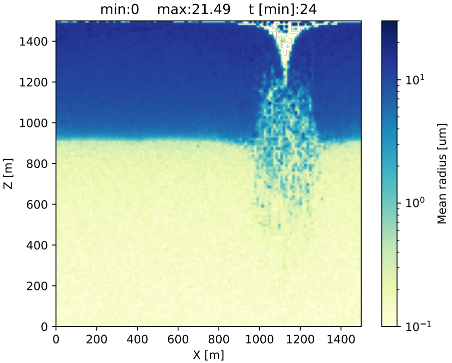


Simulation & visualisation: Piotr Bartman

sample aerosol-cloud-precipitation interactions simulation

Computational grid: 128x128
Computational particles: 2^{21}

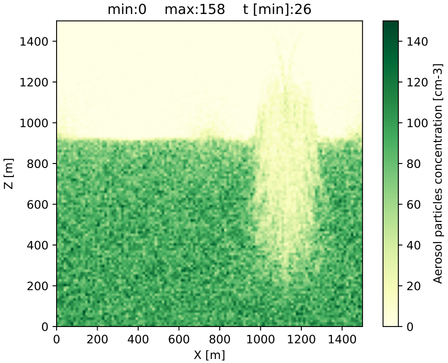


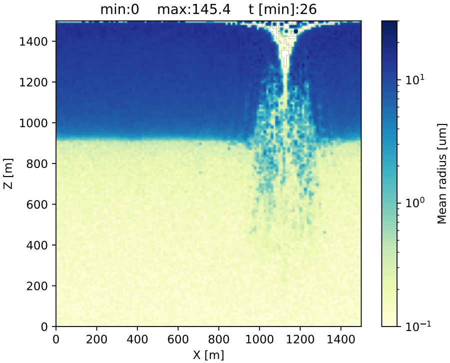


Simulation & visualisation: Piotr Bartman

sample aerosol-cloud-precipitation interactions simulation

Computational grid: 128x128
Computational particles: 2^{21}

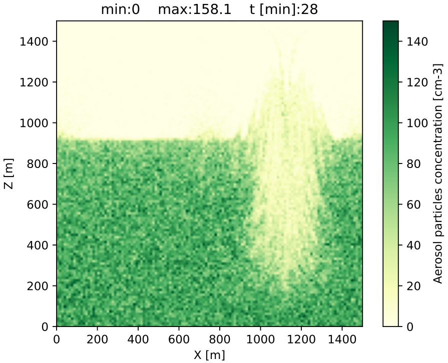


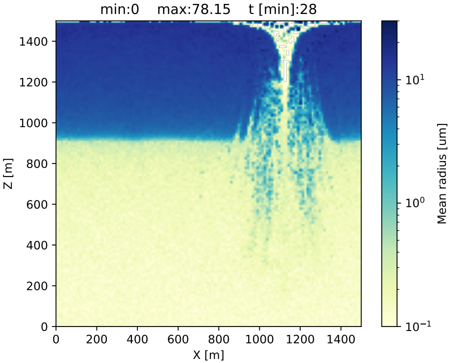


Simulation & visualisation: Piotr Bartman

sample aerosol-cloud-precipitation interactions simulation

Computational grid: 128x128
Computational particles: 2^{21}

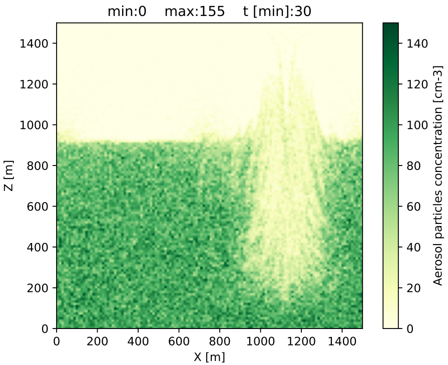


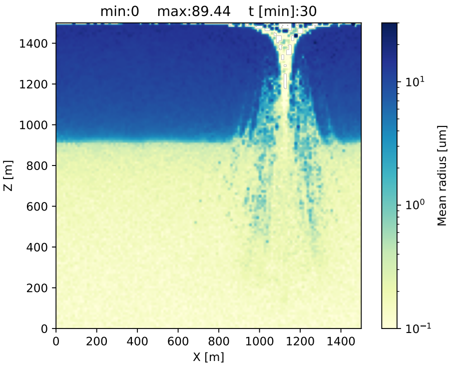


Simulation & visualisation: Piotr Bartman

sample aerosol-cloud-precipitation interactions simulation

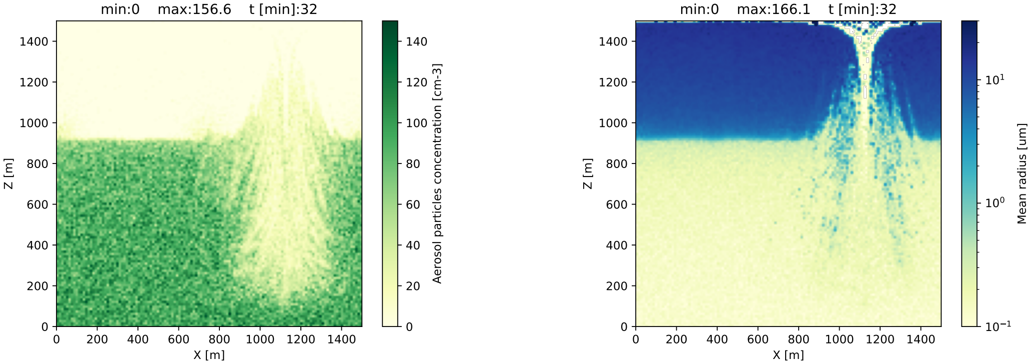
Computational grid: 128x128
Computational particles: 2^{21}





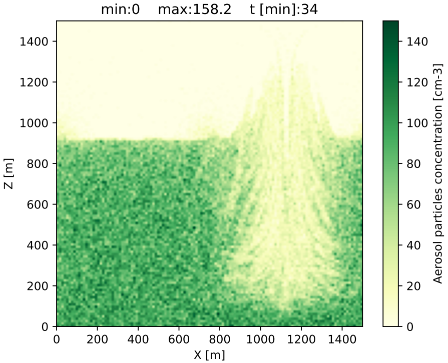
Simulation & visualisation: Piotr Bartman

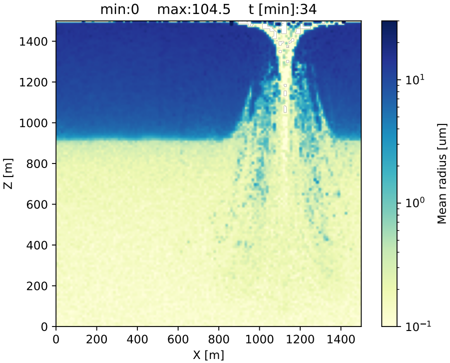
Computational grid: 128x128
Computational particles: 2^{21}



Simulation & visualisation: Piotr Bartman

Computational grid: 128x128
Computational particles: 2^{21}

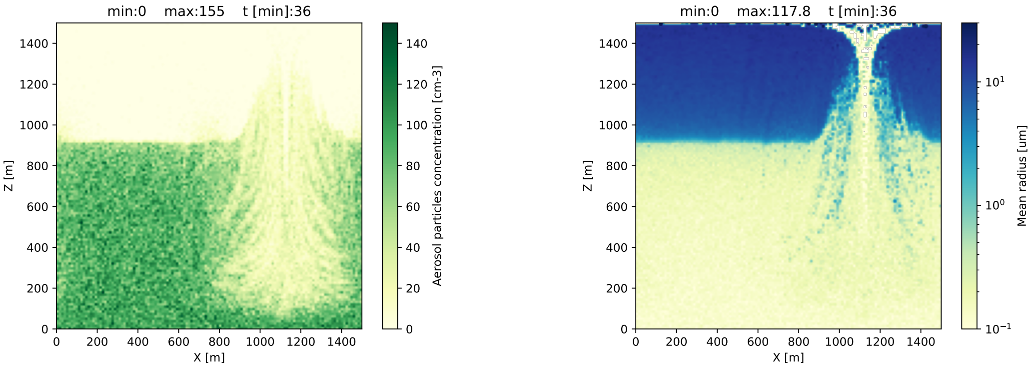




Simulation & visualisation: Piotr Bartman

sample aerosol-cloud-precipitation interactions simulation

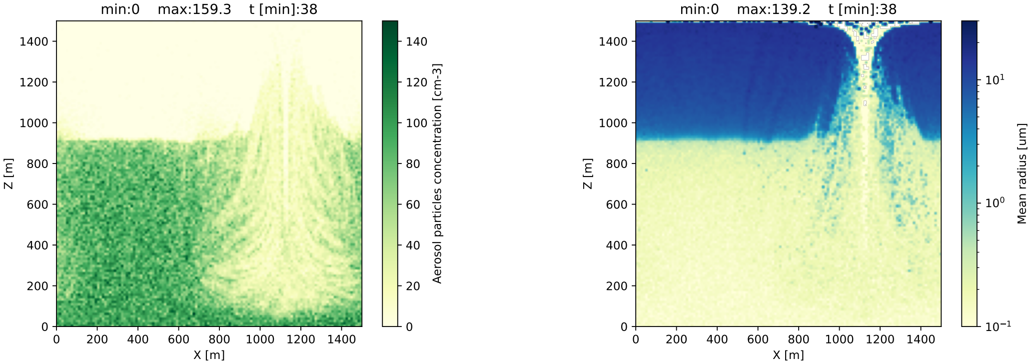
Computational grid: 128x128
Computational particles: 2^{21}



Simulation & visualisation: Piotr Bartman

sample aerosol-cloud-precipitation interactions simulation

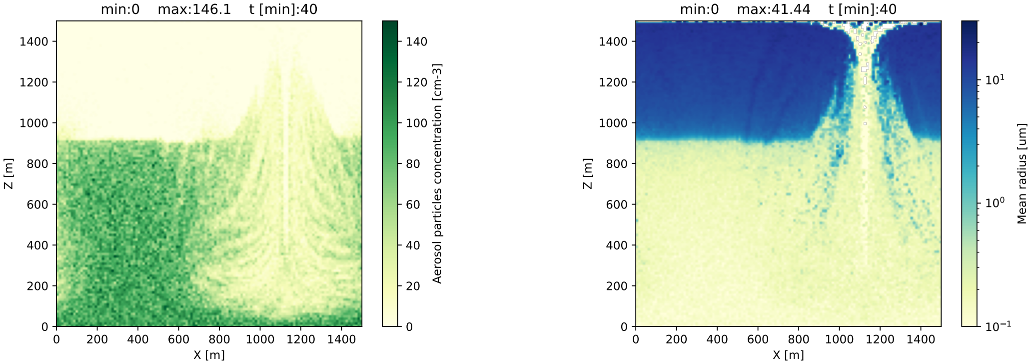
Computational grid: 128x128
Computational particles: 2^{21}



Simulation & visualisation: Piotr Bartman

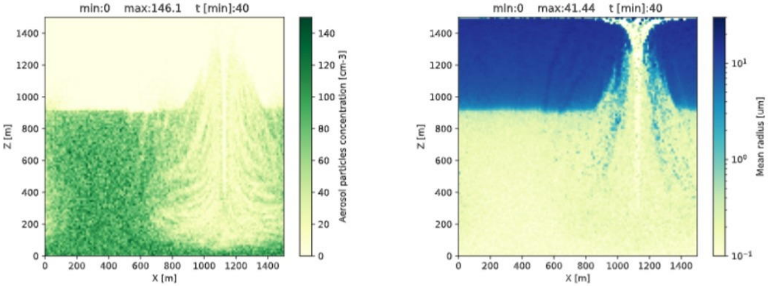
sample aerosol-cloud-precipitation interactions simulation

Computational grid: 128x128
Computational particles: 2^{21}



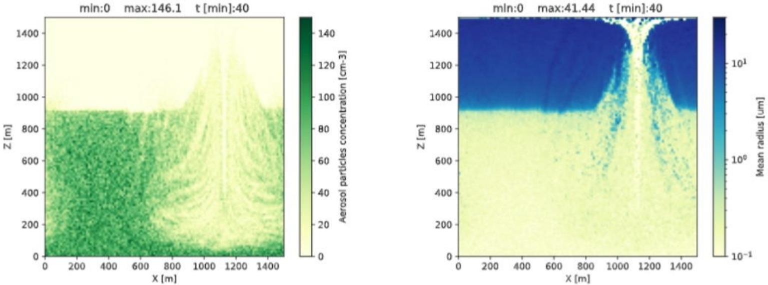
Simulation & visualisation: Piotr Bartman

PySDM:

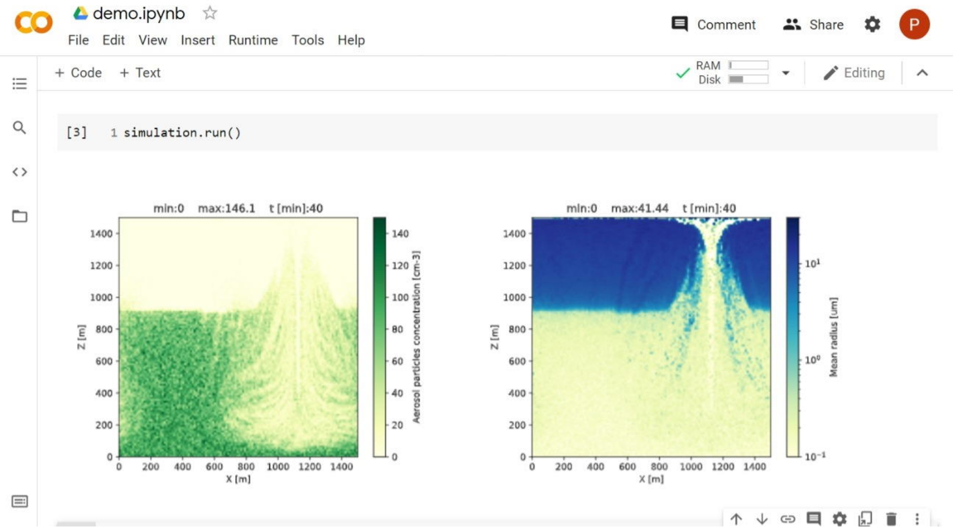

min:0 max:146.1 t [min]:40
Z [m]
X [m]
Aerosol particles concentration [cm-3]
min:0 max:41.44 t [min]:40
Z [m]
X [m]
Mean radius [um]

PySDM: Pythonic

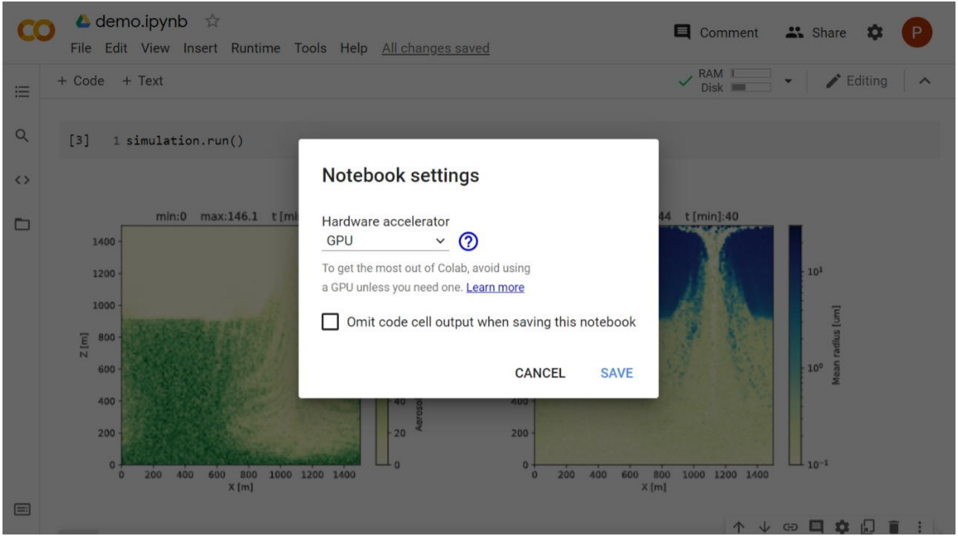
```
[3]  1 simulation.run()
```



PySDM: Pythonic, Jupyter-friendly



PySDM: Pythonic, Jupyter-friendly, GPU-enabled



first coupling with an external CFD code (Oleksii Bulenok) (https://github.com/CliMA/ClimateMachine.jl/pull/2244)

PySDM and ClimateMachine coupling examples in Kinematic setup #2244

<> Code

Open **abulenok** wants to merge 16 commits into `CliMA:master` from `abulenok:ob-pysdmachine`

Conversation 32 | Commits 16 | Checks 10 | Files changed 17 | +2,528 −1

abulenok commented on 27 Oct 2021 — Contributor

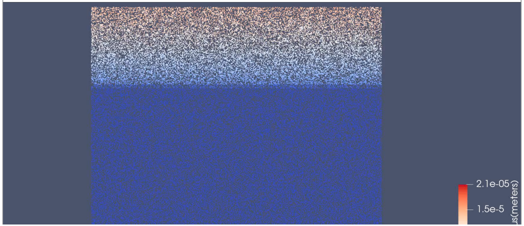
This PR includes a coupling logic for ClimateMachine.jl and PySDM.

PySDM is a particle-based aerosol/cloud microphysics package written entirely in Python.
This PR depicts how Python modules can be leveraged within ClimateMachine.jl including the continuous integration setup.

The initial set of tests included here is based on the kinematic 2D example previously used as a test case in both PySDM and ClimateMachine.jl. In the tests added in this PR, ClimateMachine.jl handles air motion and total water transport, while PySDM handles representation of aerosol and liquid water transport as well as phase changes leading to formation of cloud water.

Output from PySDM is handled using VTK files. Example animation with an evolution of radius computed from particle properties is shown below:

output.mp4



Reviewers
- slayoo
- charleskawczynski
- claresinger
- jakebolewski
- edejong-caltech
- tapios

Assignees
- trontrytel

Labels
Microphysics

Projects
None yet

Milestone
No milestone

Development
Successfully merging this pull request may close these issues.

None yet

first independent development!

manuscript submitted to *Journal of Advances in Modeling Earth Systems (JAMES)*

An efficient Bayesian approach to learning droplet collision kernels: Proof of concept using "Cloudy", a new n-moment bulk microphysics scheme

Melanie Bieli[1], Oliver R. A. Dunbar[1], Emily K. de Jong[2], Anna Jaruga[1], Tapio Schneider[1], Tobias Bischoff[1]

https://authors.library.caltech.edu/113312/1/essoar.10510248.1.pdf

22 of 28 180%

distributions capture the true parameter values within 5% of the posterior mass.

- Moving beyond perfect-model experiments, we have learned collision kernel parameters from output generated by PySDM (Bartman et al., 2021), a Lagrangian particle-based microphysics model. In this experiment, we represent model error resulting from the closure assumption in Cloudy (an assumption that PySDM does not need to make) as a simple bias term. This modification in the setup of the inverse problem allows CES to retrieve the posterior distribution of the "true" parameter, not of that which minimizes the mismatch with the PySDM data.

Plan of the talk

PySDM: context

PySDM: statement of need & goals

PySDM: demo (role play: reviewer)

PySDM: summary of key features

https://pypi.org/p/PySDM-examples

Search projects

Help Sponsors Log in Register

PySDM-examples 2.9

```
pip install PySDM-examples
```

Latest version

Released: 4 minutes ago

PySDM usage examples reproducing results from literature and depicting how to use PySDM from Python Jupyter notebooks

Navigation

- Project description
- Release history
- Download files

Project links

- Homepage

Statistics

GitHub statistics:

- Stars: 2
- Forks: 10

Project description

License GPL v3 | Copyright Jagiellonian University | DOI 10.5281/zenodo.6604645

PySDM-examples passing

pull requests 2 open | pull requests 159 closed

pypi package 2.8 | API docs pdoc3

This repository stores example files for `PySDM` depicting usage of `PySDM` from Python via Jupyter. For information on the `PySDM` package itself and examples of usage from Julia and Matlab, see PySDM README.md file.

Please use the PySDM issue-tracking and discussion infrastructure for `PySDM-examples` as well.

0D box-model coalescence-only examples:

- Shima et al. 2009 (Box model, coalescence only, test case employing Golovin analytical solution):
  - Fig. 2: render nbviewer | launch binder | Open in Colab
- Berry 1967 (Box model, coalescence only, test cases for realistic kernels):
  - Figs. 5, 8 & 10: render nbviewer | launch binder | Open in Colab
- Bieli et al. 2022 (Box model, coalescence and breakup with fixed coalescence efficiency):
  - Fig. 2: render nbviewer | launch binder | Open in Colab

PySDM-examples: Lowe et al. 2019

nature COMMUNICATIONS

https://doi.org/10.1038/s41467-019-12982-0 OPEN

Key drivers of cloud response to surface-active organics

S.J. Lowe[1,2], D.G. Partridge[3], J.F. Davies[4], K.R. Wilson[5], D. Topping[6] & I. Riipinen[1,2,7*]

Aerosol-cloud interactions constitute the largest source of uncertainty in global radiative forcing estimates, hampering our understanding of climate evolution. Recent empirical evidence suggests surface tension depression by organic aerosol to significantly influence the formation of cloud droplets, and hence cloud optical properties. In climate models, however, surface tension of water is generally assumed when predicting cloud droplet concentrations. Here we show that the sensitivity of cloud microphysics, optical properties and shortwave radiative effects to the surface phase are dictated by an interplay between the aerosol particle size distribution, composition, water availability and atmospheric dynamics. We demonstrate that accounting for the surface phase becomes essential in clean environments in which ultrafine particle sources are present. Through detailed sensitivity analysis, quantitative constraints on the key drivers – aerosol particle number concentrations, organic fraction and fixed updraft velocity – are derived for instances of significant cloud microphysical susceptibilities to the surface phase.

PySDM-examples: Lowe et al. 2019

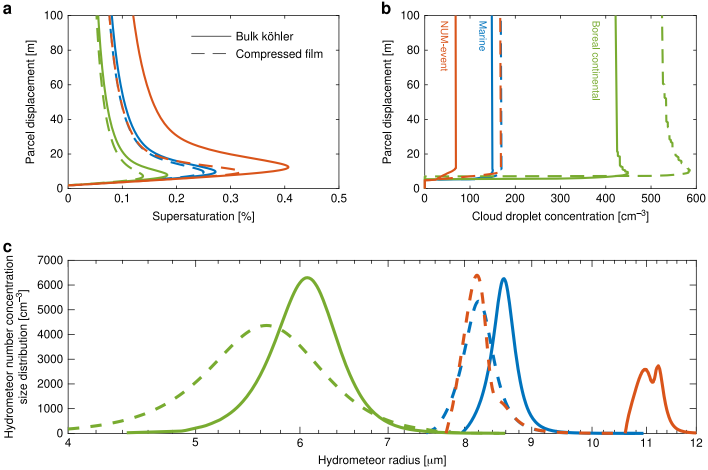
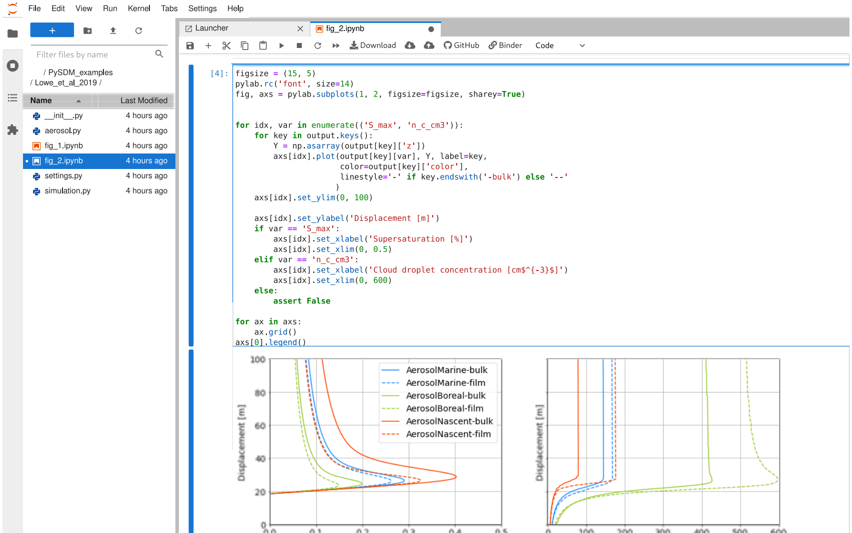


Fig. 2 Simulated microphysics of cloud events on marine (MA, blue), boreal (HYY, green) and NUM-event (NE, orange) aerosol populations. Cloud-formation event simulations using bulk Köhler BK (solid lines) and approximate compressed film CF (dotted lines) models of cloud droplet activation with initial temperature T = 280 K, pressure P = 98,000 Pa, supersaturation s = −0.1% and fixed updraft velocity w = 0.32 ms^{-1}. Simulated (a) ambient parcel supersaturation and (b) cloud droplet number concentration during parcel ascent. c Simulated droplet size distribution at a parcel displacement 200 m above initialisation

PySDM-examples: Lowe et al. 2019

example contributed by Clare Singer et al. (https://claresinger.github.io/)



Plan of the talk

PySDM: context

PySDM: statement of need & goals

PySDM: demo (role play: reviewer)

PySDM: summary of key features

PySDM: backends, dynamics & environments

PySDM: backends, dynamics & environments

"backends"

- CPU (Numba/LLVM)
- GPU (ThrustRTC/CUDA)

PySDM: backends, dynamics & environments

"dynamics"

- coalescence (SDM + dt-adaptivity)
- condensation (dt-adaptive, bespoke semi-implicit ODE solver)
- displacement (incl. sedimentation)
- aqueous chemistry (Hoppel gap)
- immersion freezing (INAS & ABIFM)
- collisional breakup
- ...

"backends"

- CPU (Numba/LLVM)
- GPU (ThrustRTC/CUDA)

PySDM: backends, dynamics & environments

"dynamics"

- coalescence (SDM + dt-adaptivity)
- condensation (dt-adaptive, bespoke semi-implicit ODE solver)
- displacement (incl. sedimentation)
- aqueous chemistry (Hoppel gap)
- immersion freezing (INAS & ABIFM)
- collisional breakup
- ...

"backends"

- CPU (Numba/LLVM)
- GPU (ThrustRTC/CUDA)

"environments"

- Box
- Parcel
- PyMPDATA-based:
  - Kinematic1D
  - Kinematic2D

https://github.com/atmos-cloud-sim-uj/PySDM/

Python 3 | LLVM Numba | CUDA ThrustRTC | Linux ✓ | macOS ✓ | Windows ✓ | Jupyter ✓ | Maintained? yes | Open Hub PySDM

JOSS 10.21105/joss.03219 | DOI 10.5281/zenodo.6604644

EU Funding by FNP | PL Funding by NCN | US DOE Funding by ASR

License GPL v3 | Copyright Jagiellonian University

PySDM passing | build passing | codecov 76%

pull requests 5 open | pull requests 423 closed

issues 79 open | issues 340 closed

pypi package 2.9 | API docs pdoc3

PySDM is a package for simulating the dynamics of population of particles. It is intended to serve as a building block for simulation systems modelling fluid flows involving a dispersed phase, with PySDM being responsible for representation of the dispersed phase. Currently, the development is focused on atmospheric cloud physics applications, in particular on modelling the dynamics of particles immersed in moist air using the particle-based (a.k.a. super-droplet) approach to represent aerosol/cloud/rain microphysics. The package features a Pythonic high-performance implementation of the Super-Droplet Method (SDM) Monte-Carlo algorithm for representing collisional growth (Shima et al. 2009), hence the name.

PySDM has two alternative parallel number-crunching backends available: multi-threaded CPU backend based on Numba and GPU-resident backend built on top of ThrustRTC. The `Numba` backend (aliased `CPU`) features multi-threaded parallelism for multi-core CPUs, it uses the just-in-time compilation technique based on the LLVM infrastructure. The `ThrustRTC` backend (aliased `GPU`) offers GPU-resident operation of PySDM leveraging the SIMT parallelisation model. Using the `GPU` backend requires nVidia hardware and CUDA driver.

For an overview paper on PySDM v1 (and the preferred item to cite if using PySDM), see Bartman et al. 2021 arXiv e-print (submitted to JOSS). For a list of talks and other materials on PySDM, see the project wiki.

A pdoc-generated documentation of PySDM public API is maintained at: https://atmos-cloud-sim-uj.github.io/PySDM

PySDM: technological stack

- 100% Python code `python.org`
- Numba (JIT, multi-threading) `numba.pydata.org`
- ThrustRTC (GPU-resident backend) `pypi.org/project/ThrustRTC`

PySDM: technological stack

- 100% Python code `python.org`
- Numba (JIT, multi-threading) `numba.pydata.org`
- ThrustRTC (GPU-resident backend) `pypi.org/project/ThrustRTC`

- GitHub & GitHub Actions `github.com`
- Codecov `codecov.io`
- AppVeyor `appveyor.com`

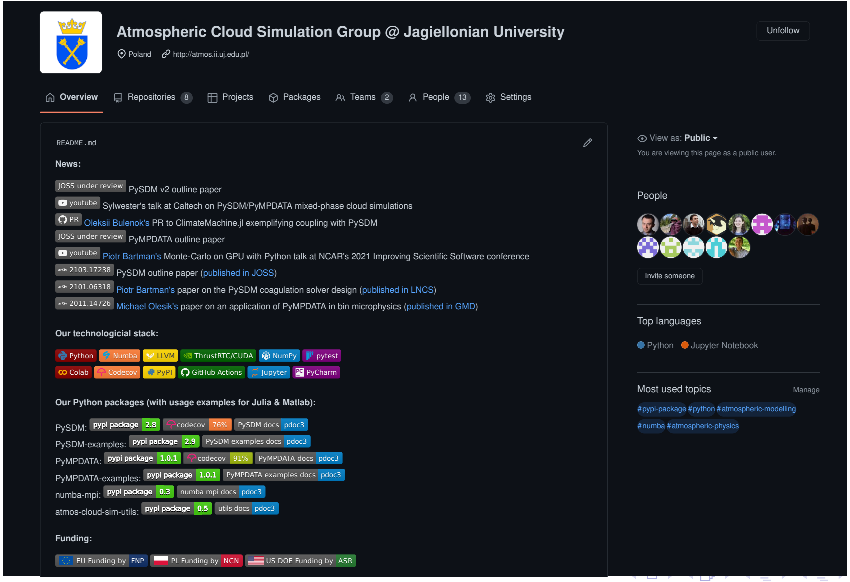
PySDM: technological stack

- 100% Python code `python.org`
- Numba (JIT, multi-threading) `numba.pydata.org`
- ThrustRTC (GPU-resident backend) `pypi.org/project/ThrustRTC`

- GitHub & GitHub Actions `github.com`
- Codecov `codecov.io`
- AppVeyor `appveyor.com`

- Jupyter `jupyter.org`
- Binder `mybinder.org`
- Colab `colab.research.google.com`

https://atmos.ii.uj.edu.pl/



JOSS The Journal of Open Source Software

Download paper

Software archive

Review

Editor: @dhhagan (all papers)
Reviewers: @darothen (all reviews), @josephhardinee (all reviews)

Authors

Piotr Bartman (0000-0003-0265-6428), Oleksii Bulenok (0000-0003-2272-8548), Kamil Górski, Anna Jaruga (0000-0003-3194-6440), Grzegorz Łazarski (0000-0002-5595-371X), Michael A. Olesik (0000-0002-6319-9358), Bartosz Piasecki, Clare E. Singer (0000-0002-1708-0997), Aleksandra Talar, Sylwester Arabas (0000-0003-2361-0082)

Citation

Bartman et al., (2022). PySDM v1: particle-based cloud modeling package for warm-rain microphysics and aqueous chemistry. Journal of Open Source Software, 7(72), 3219, https://doi.org/10.21105/joss.03219

Copy citation string · Copy BibTeX

Tags

physics-simulation monte-carlo-simulation gpu-computing atmospheric-modeling particle-system numba thrust nvrtc pint atmospheric-physics

Altmetrics

2

Markdown badge

JOSS 10.21105/joss.03219

PySDM v1: particle-based cloud modeling package for warm-rain microphysics and aqueous chemistry

Piotr Bartman[1], Oleksii Bulenok[1], Kamil Górski[1], Anna Jaruga[2], Grzegorz Łazarski[1,3], Michael A. Olesik[4], Bartosz Piasecki[1], Clare E. Singer[2], Aleksandra Talar[1], and Sylwester Arabas[5,1]

1 Faculty of Mathematics and Computer Science, Jagiellonian University, Kraków, Poland 2 Department of Environmental Science and Engineering, California Institute of Technology, Pasadena, CA, USA 3 Faculty of Chemistry, Jagiellonian University, Kraków, Poland 4 Faculty of Physics, Astronomy and Applied Computer Science, Jagiellonian University, Kraków, Poland 5 University of Illinois at Urbana-Champaign, Urbana, IL, USA

DOI: 10.21105/joss.03219

Software

- Review
- Repository
- Archive

Editor: David Hagan

Reviewers:

- @darothen
- @josephhardinee

Introduction

PySDM is an open-source Python package for simulating the dynamics of particles undergoing condensational and collisional growth, interacting with a fluid flow and subject to chemical composition changes. It is intended to serve as a building block for process-level as well as computational-fluid dynamics simulation systems involving representation of a continuous phase (air) and a dispersed phase (aerosol), with PySDM being responsible for representation of the dispersed phase. For major version 1 (v1), the development has been focused on atmospheric cloud physics applications, in particular on modeling the dynamics of particles immersed in moist air using the particle-based approach to represent the evolution of the size spectrum of aerosol/cloud/rain particles. The particle-based approach contrasts the more commonly used

Acknowledgements

co-authors, contributors, collaborators

- @uj.edu.pl: P. Bartman, M. Olesik, G. Łazarski, O. Bulenok, ...
- @caltech.edu: E. de Jong, C. Singer, A. Jaruga, B. Mackay, S. Azimi, ...
- @illinois.edu: N. Riemer, M. West & J. Curtis

Acknowledgements

co-authors, contributors, collaborators

- @uj.edu.pl: P. Bartman, M. Olesik, G. Łazarski, O. Bulenok, ...
- @caltech.edu: E. de Jong, C. Singer, A. Jaruga, B. Mackay, S. Azimi, ...
- @illinois.edu: N. Riemer, M. West & J. Curtis

funding

- PL / National Science Centre
- EU / Foundation for Polish Science
- US / DOE Atmospheric System Research & Schmidt Futures

Acknowledgements

co-authors, contributors, collaborators

- @uj.edu.pl: P. Bartman, M. Olesik, G. Łazarski, O. Bulenok, ...
- @caltech.edu: E. de Jong, C. Singer, A. Jaruga, B. Mackay, S. Azimi, ...
- @illinois.edu: N. Riemer, M. West & J. Curtis

funding

- PL / National Science Centre
- EU / Foundation for Polish Science
- US / DOE Atmospheric System Research & Schmidt Futures

Thank you for your attention!

more: https://atmos.ii.uj.edu.pl/

contact: sylwester.arabas@uj.edu.pl