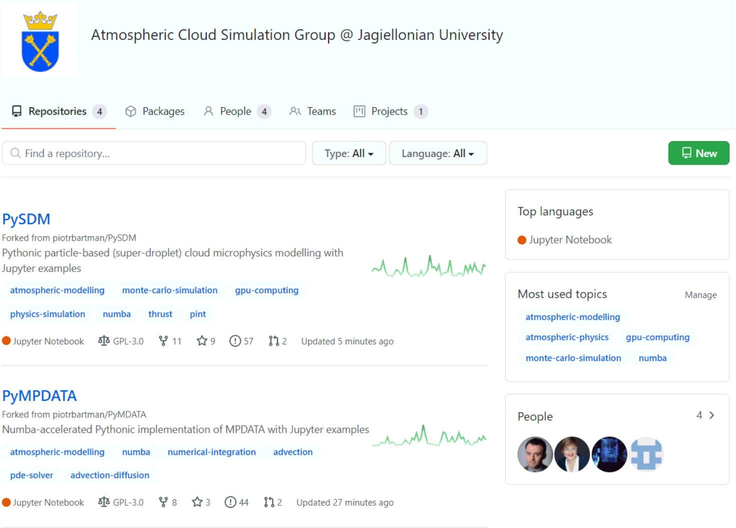
PySDM: Pythonic particle-based cloud microphysics package

Piotr Bartman

Jagiellonian University in Kraków

Introduction

PySDM

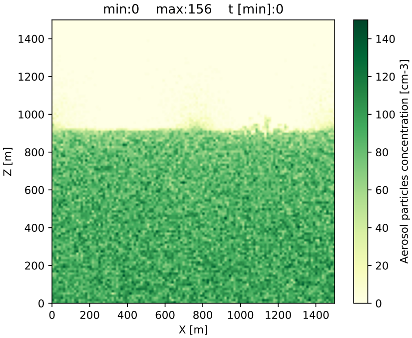


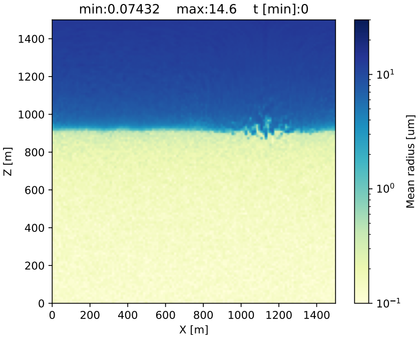
ACSG: Sylwester Arabas, Piotr Bartman, Michael Olesik

Aerosol-cloud-precipitation interactions

“Cloud and ship. Ukraine, Crimea, Black sea, view from Ai-Petri mountain”
(photo: Yevgen Timashov / National Geographic)

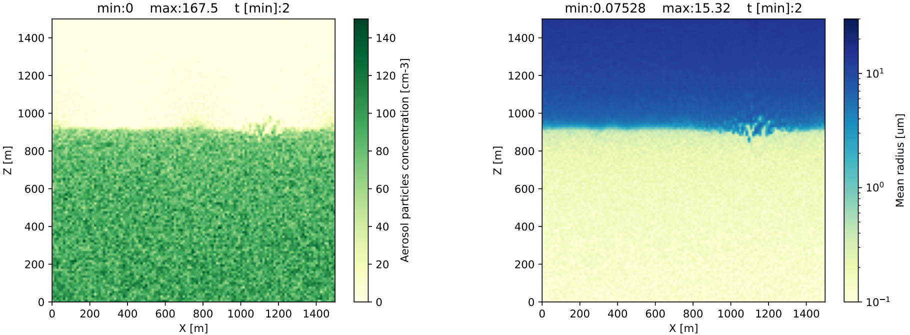
Simulation of stratocumulus and virga phenomenon





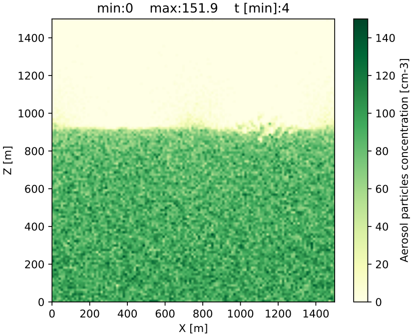
Computational grid: 128x128
Computational particles: 2^{21}

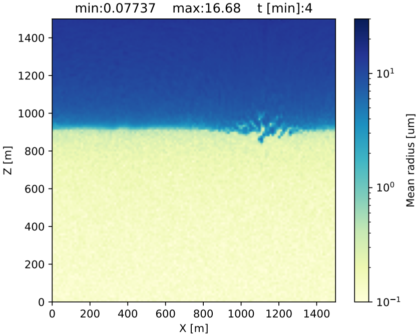
Simulation of stratocumulus and virga phenomenon



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

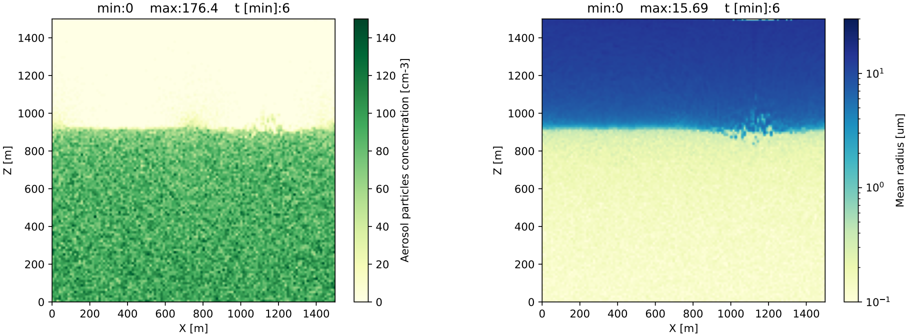
Simulation of stratocumulus and virga phenomenon





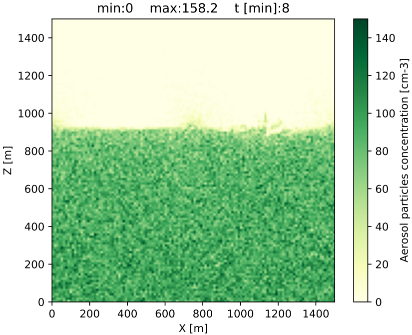
Computational grid: 128x128
Computational particles: 2^{21}

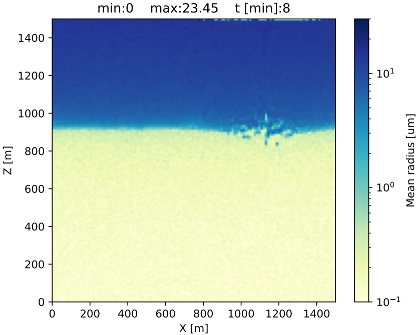
Simulation of stratocumulus and virga phenomenon



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

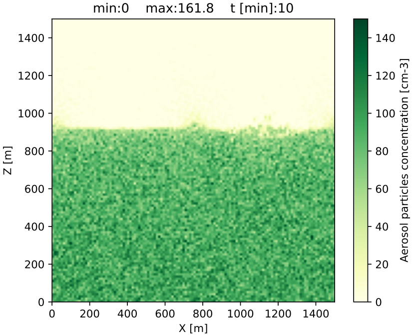
Simulation of stratocumulus and virga phenomenon

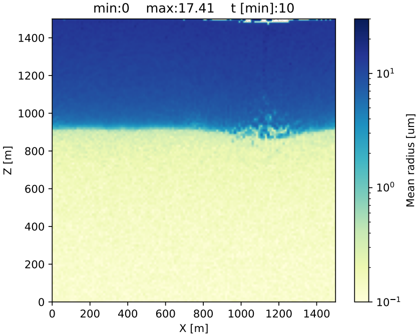




Computational grid: 128x128
Computational particles: 2^{21}

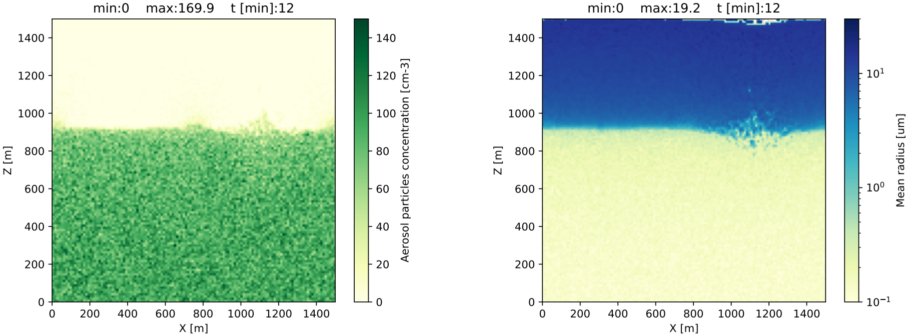
Simulation of stratocumulus and virga phenomenon





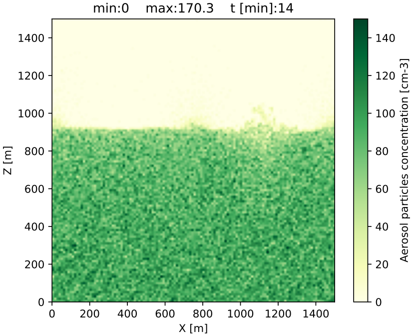
Computational grid: 128x128
Computational particles: 2^{21}

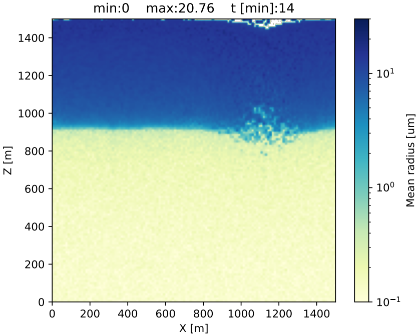
Simulation of stratocumulus and virga phenomenon



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

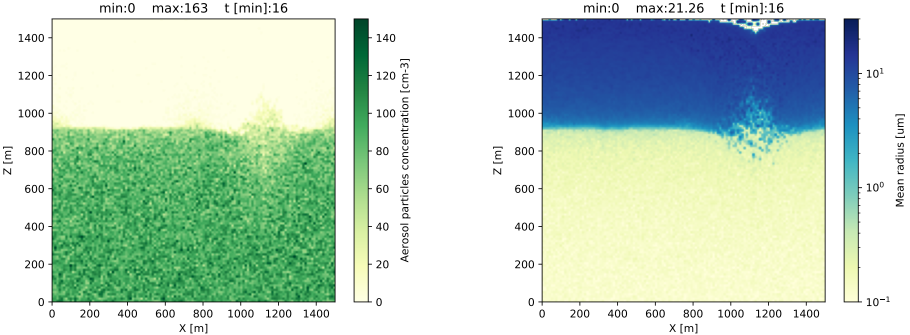
Simulation of stratocumulus and virga phenomenon





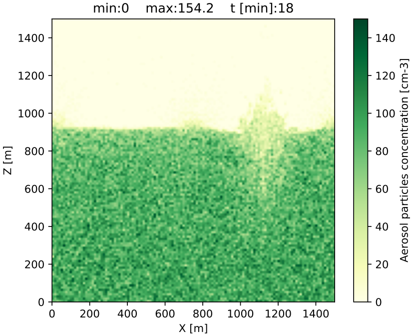
Computational grid: 128x128
Computational particles: 2^{21}

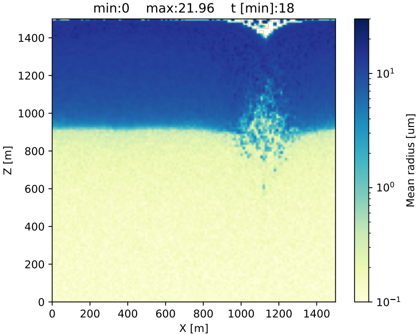
Simulation of stratocumulus and virga phenomenon



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

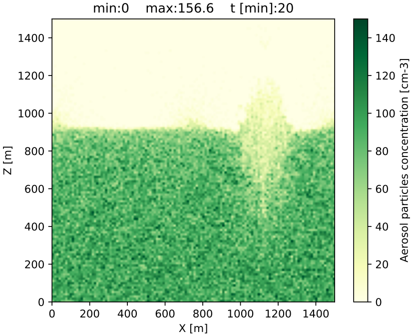
Simulation of stratocumulus and virga phenomenon

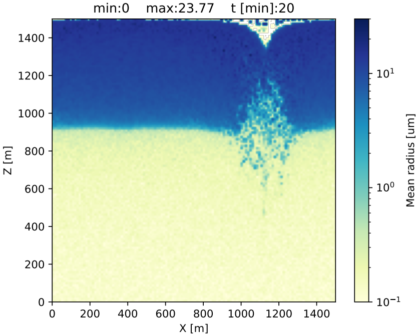




Computational grid: 128x128
Computational particles: 2^{21}

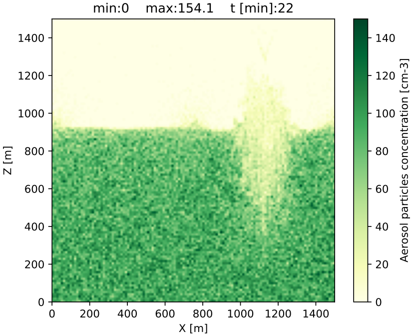
Simulation of stratocumulus and virga phenomenon

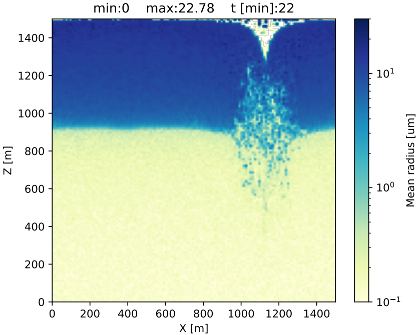




Computational grid: 128x128
Computational particles: 2^{21}

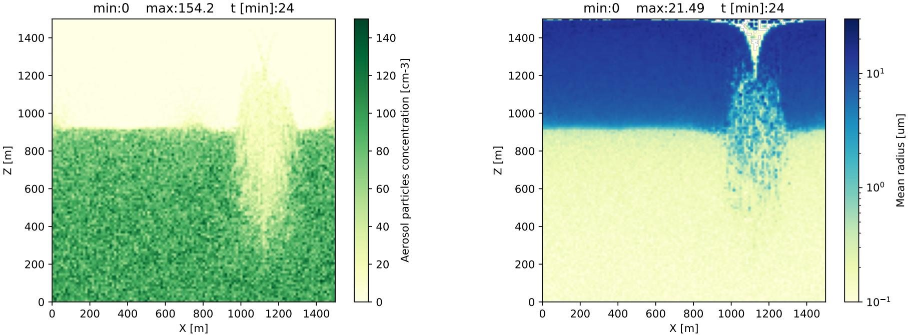
Simulation of stratocumulus and virga phenomenon





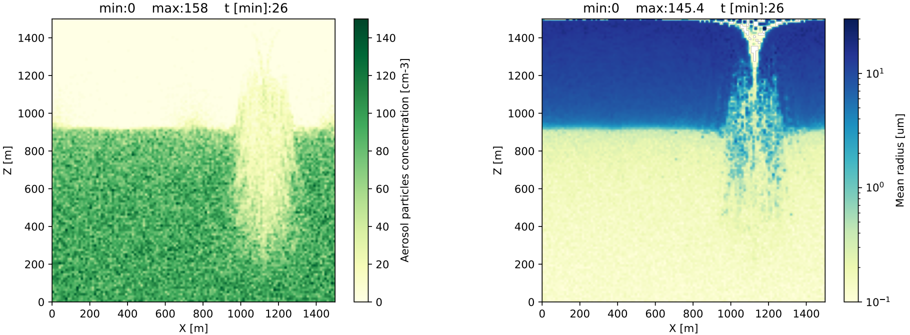
Computational grid: 128x128
Computational particles: 2^{21}

Simulation of stratocumulus and virga phenomenon



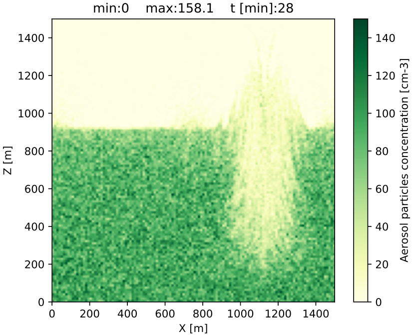
Computational grid: 128x128
Computational particles: 2^{21}

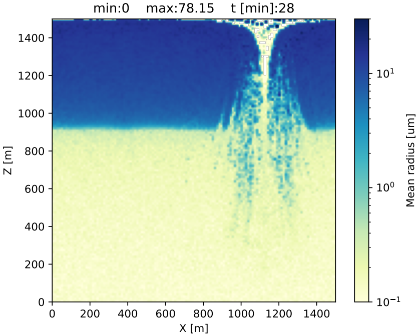
Simulation of stratocumulus and virga phenomenon



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

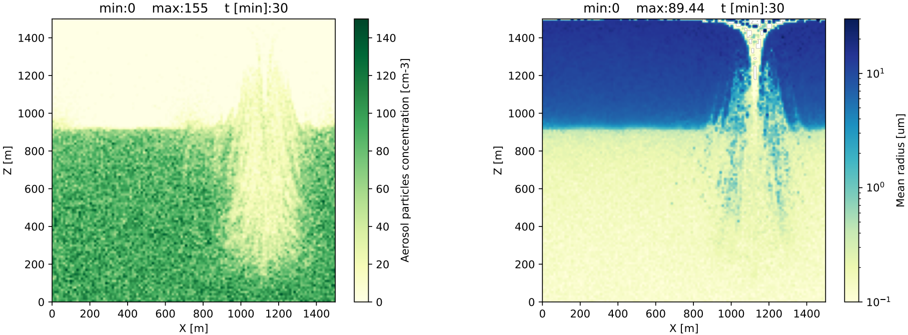
Simulation of stratocumulus and virga phenomenon





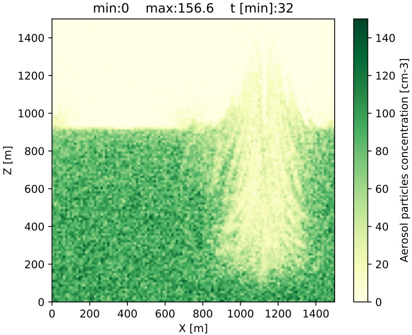
Computational grid: 128x128
Computational particles: 2^{21}

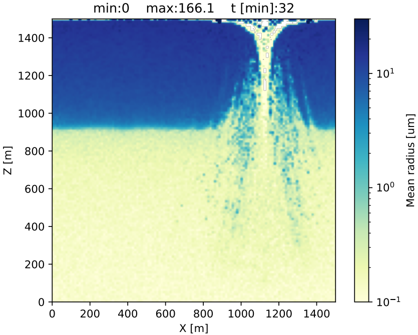
Simulation of stratocumulus and virga phenomenon



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

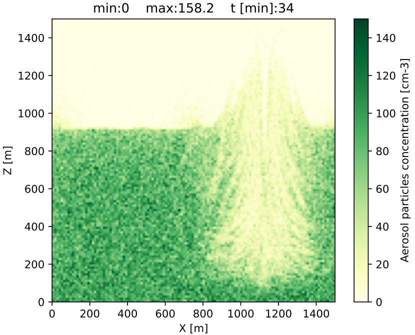
Simulation of stratocumulus and virga phenomenon

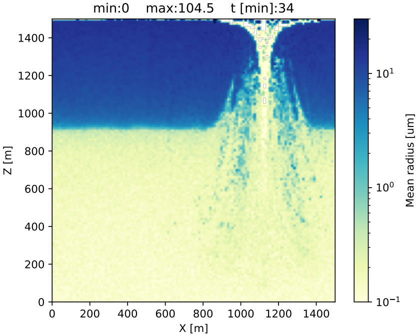




Computational grid: 128x128
Computational particles: 2^{21}

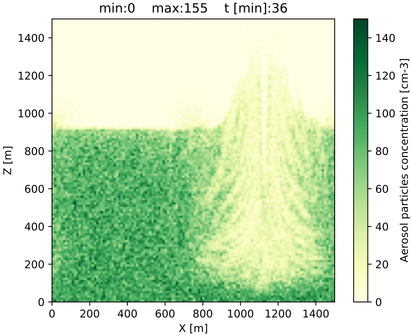
Simulation of stratocumulus and virga phenomenon

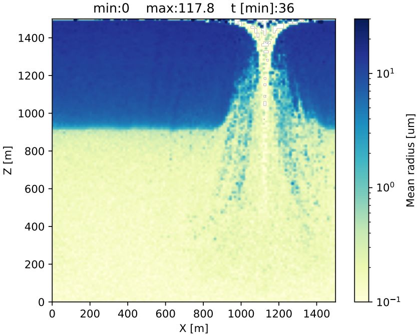




Computational grid: 128x128
Computational particles: 2^{21}

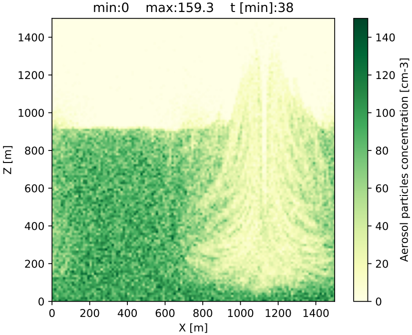
Simulation of stratocumulus and virga phenomenon

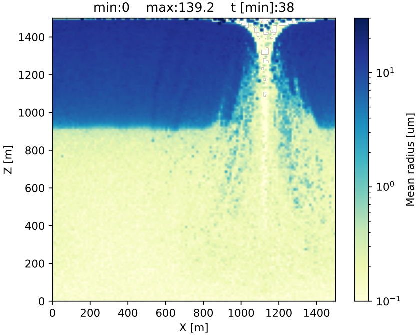




Computational grid: 128x128
Computational particles: 2^{21}

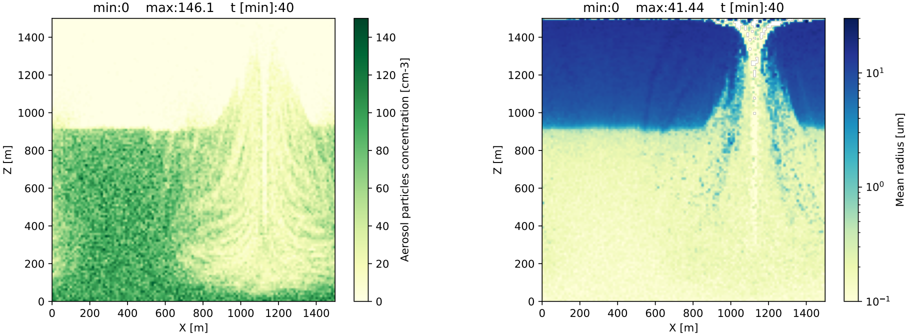
Simulation of stratocumulus and virga phenomenon





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

Simulation of stratocumulus and virga phenomenon



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

Modeled processes

- Condensation - cloud droplet condensational growth
- Advection and Sedimentation - transport of droplets due to air flow and gravity
- Coalescence - cloud droplet collisional growth

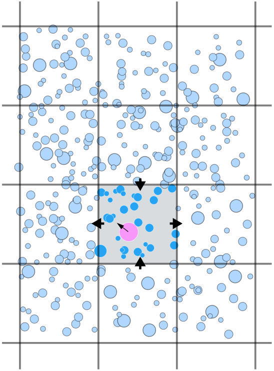
Cloud droplet condensational growth

Thermodynamic variables

q mixing ratio of water vapor (ratio of the vapor density to the dry-air density),

θ potential temperature (temperature that the air would attain if adiabatically brought to a reference pressure of 1000 hPa),

ρ_d dry-air density,



Droplet growth law

Approximation of the two equations of diffusion (heat and vapor) with a single one:

$$\dot{s} = \frac{ds}{dr}\dot{r} = \frac{ds}{dr}\frac{1}{r}\frac{(RH(q,\theta,\rho_d)) - \frac{a}{r} + \frac{b}{r^3}}{F(q,\theta,\rho_d)} \tag{1}$$

r - radius of droplet, $s = log(\frac{4}{3}\pi r^3)$
RH - relative humidity
a, b - parameters set according to the kappa-Köhler parameterization of hygroscopicity
F^{-1} - effective diffusion coefficient

Evolution of the environment (in each cell)

$$\begin{bmatrix} \dot{s}_{[i]} \\ \vdots \\ \dot{\rho}_\mathrm{d} \\ \\ \dot{q} \\ \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{s}(s_{[i]}, \theta, q, \rho_\mathrm{d}) \\ \vdots \\ 0 \\ \\ \dot{q}_{cond} + \dot{q}_{env} \\ \\ \dot{\theta}_{cond} + \dot{\theta}_{env} \end{bmatrix}$$

$$v_{[i]} = e^{s_{[i]}}$$

Evolution of the environment (in each cell)

$$\begin{bmatrix} \dot{s}_{[i]} \\ \vdots \\ \dot{\rho}_\mathrm{d} \\ \\ \dot{q} \\ \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{s}(s_{[i]}, \theta, q, \rho_\mathrm{d}) \\ \vdots \\ 0 \\ \\ \dot{q}_{cond} + \dot{q}_{env} \\ \\ \dot{\theta}_{cond} + \dot{\theta}_{env} \end{bmatrix}$$

$$\dot{q}_{cond}(x, z) = -\frac{\rho_l}{\rho_\mathrm{d}} \frac{1}{\Delta V} \sum_i \xi_{[i]} \frac{dv_{[i]}}{dt}$$

$$v_{[i]} = e^{s_{[i]}}$$

Evolution of the environment (in each cell)

$$\begin{bmatrix} \dot{s}_{[i]} \\ \vdots \\ \dot{\rho}_\mathrm{d} \\ \dot{q} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{s}(s_{[i]}, \theta, q, \rho_\mathrm{d}) \\ \vdots \\ 0 \\ \dot{q}_{cond} + \dot{q}_{env} \\ \dot{\theta}_{cond} + \dot{\theta}_{env} \end{bmatrix}$$

$$\dot{q}_{cond}(x, z) = -\frac{\rho_l}{\rho_\mathrm{d}} \frac{1}{\Delta V} \sum_i \xi_{[i]} \frac{dv_{[i]}}{dt}$$

$$\dot{\theta}_{cond}(x, z) = -\frac{l(T(\theta, q, \rho_d))\dot{q}_{cond}}{c_p T(\theta, q, \rho_d)\theta\rho_d}$$

$$v_{[i]} = e^{s_{[i]}}$$

Evolution of the environment (in each cell)

$$\begin{bmatrix} \dot{s}_{[i]} \\ \vdots \\ \dot{\rho}_{\mathrm{d}} \\ \dot{q} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{s}(s_{[i]}, \theta, q, \rho_{\mathrm{d}}) \\ \vdots \\ 0 \\ \dot{q}_{cond} + \dot{q}_{env} \\ \dot{\theta}_{cond} + \dot{\theta}_{env} \end{bmatrix}$$

$$\dot{q}_{cond}(x, z) = -\frac{\rho_l}{\rho_{\mathrm{d}}} \frac{1}{\Delta V} \sum_i \xi_{[i]} \frac{dv_{[i]}}{dt}$$

$$\dot{\theta}_{cond}(x, z) = -\frac{l(T(\theta, q, \rho_d))\dot{q}_{cond}}{c_p T(\theta, q, \rho_d)\theta\rho_d}$$

$$\dot{q}_{env}(x, z) = -\rho_{\mathrm{d}}^{-1} \nabla \cdot (\vec{u}\rho_{\mathrm{d}} q_{\mathrm{v}})$$

$$v_{[i]} = e^{s_{[i]}}$$

Evolution of the environment (in each cell)

$$\begin{bmatrix} \dot{s}_{[i]} \\ \vdots \\ \dot{\rho}_\mathrm{d} \\ \dot{q} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{s}(s_{[i]}, \theta, q, \rho_\mathrm{d}) \\ \vdots \\ 0 \\ \dot{q}_{cond} + \dot{q}_{env} \\ \dot{\theta}_{cond} + \dot{\theta}_{env} \end{bmatrix}$$

$$v_{[i]} = e^{s_{[i]}}$$

$$\dot{q}_{cond}(x, z) = -\frac{\rho_l}{\rho_\mathrm{d}} \frac{1}{\Delta V} \sum_i \xi_{[i]} \frac{dv_{[i]}}{dt}$$

$$\dot{\theta}_{cond}(x, z) = -\frac{l(T(\theta, q, \rho_d))\dot{q}_{cond}}{c_p T(\theta, q, \rho_d)\theta\rho_d}$$

$$\dot{q}_{env}(x, z) = -\rho_\mathrm{d}^{-1} \nabla \cdot (\vec{u}\rho_\mathrm{d} q_\mathrm{v})$$

$$\dot{\theta}_{env}(x, z) = -\rho_\mathrm{d}^{-1} \nabla \cdot (\vec{u}\rho_\mathrm{d} \theta_\mathrm{d})$$

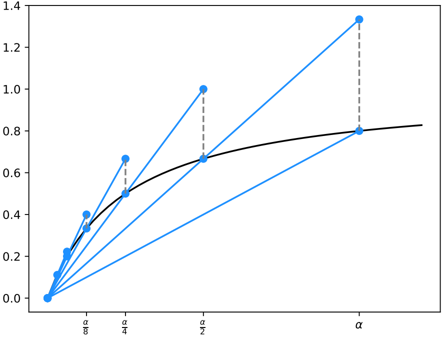
Adaptive time step (proposed in PySDM)

In each major time step n, the number of time substeps is adjusted iteratively searching for such $m \in \mathbb{N}$, and as a consequence $\alpha = \frac{1}{2^m}$, for which the following condition holds:

$$\left| \underbrace{\left(\theta^{n+\frac{1}{2\alpha}} \Big|_{2\alpha} - \theta^n \right)}_{\Delta\theta|_{2\alpha}} - 2 \underbrace{\left(\theta^{n+\frac{1}{\alpha}} \Big|_{\alpha} - \theta^n \right)}_{\Delta\theta|_{\alpha}} \right| < r_\theta |\theta^n| \qquad (2)$$

where r_θ is the relative tolerance. Note that if only θ is differentiable in the limit of $m \to \infty$, the left-hand side of ineq. (2) $\left| \Delta\theta|_{2\alpha} - 2\, \Delta\theta|_{\alpha} \right| \to 0$ assuring convergence.

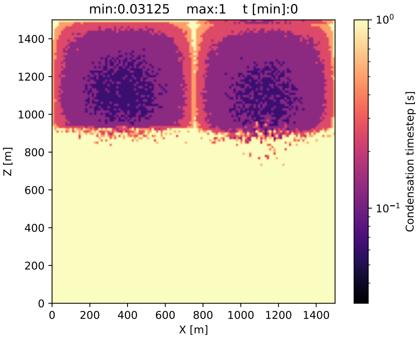
Adaptive time step (proposed in PySDM)

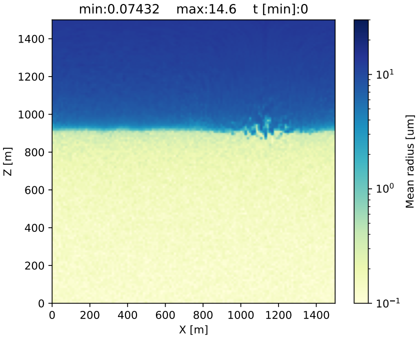


Conceptual view of inequality (2).

- no additional memory is required
- condensation step adapted:
  - once per major time step
  - in each cell independently

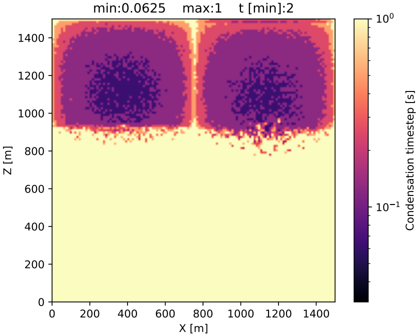
Condensation solver adaptivity

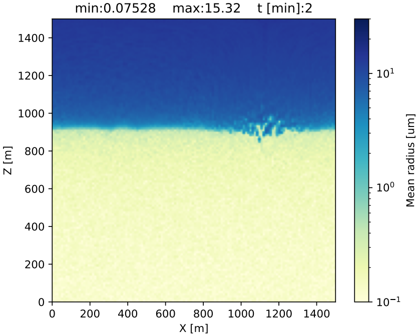




Computational grid: 128x128
Computational particles: 2^{21}

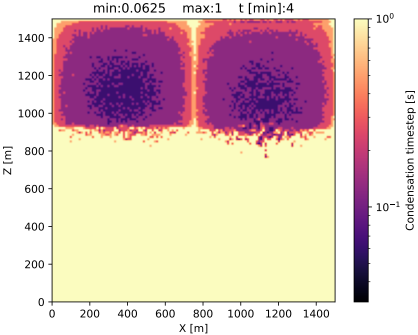
Condensation solver adaptivity

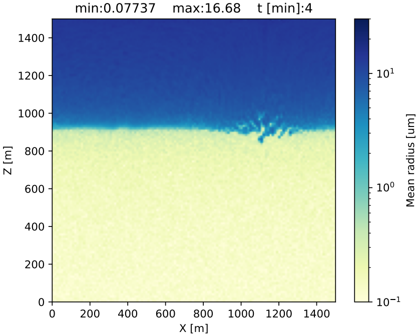




Computational grid: 128x128
Computational particles: 2^{21}

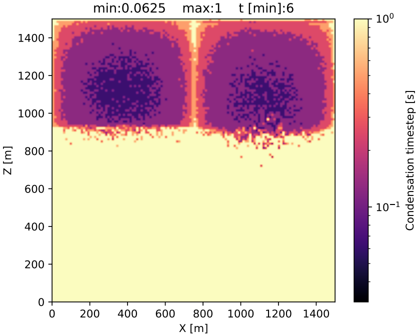
Condensation solver adaptivity

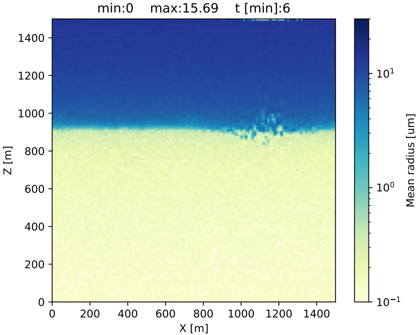




Computational grid: 128x128
Computational particles: 2^{21}

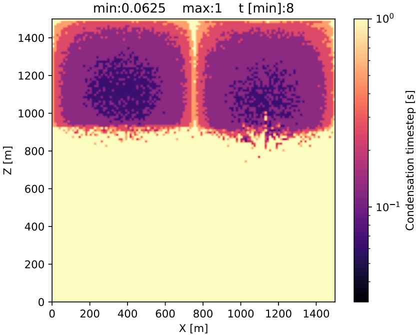
Condensation solver adaptivity

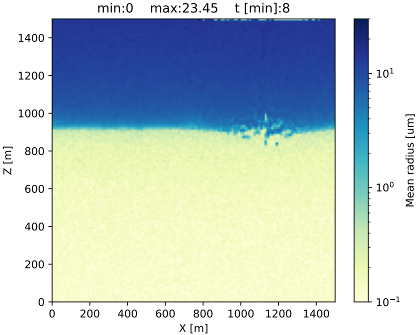




Computational grid: 128x128
Computational particles: 2^{21}

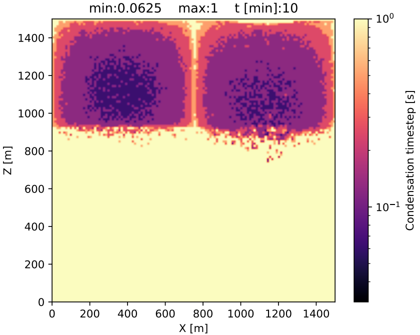
Condensation solver adaptivity

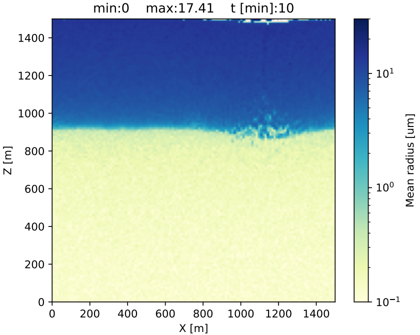




Computational grid: 128x128
Computational particles: 2^{21}

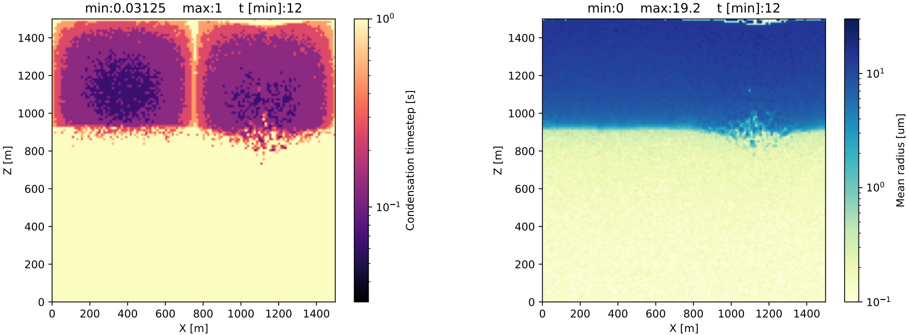
Condensation solver adaptivity





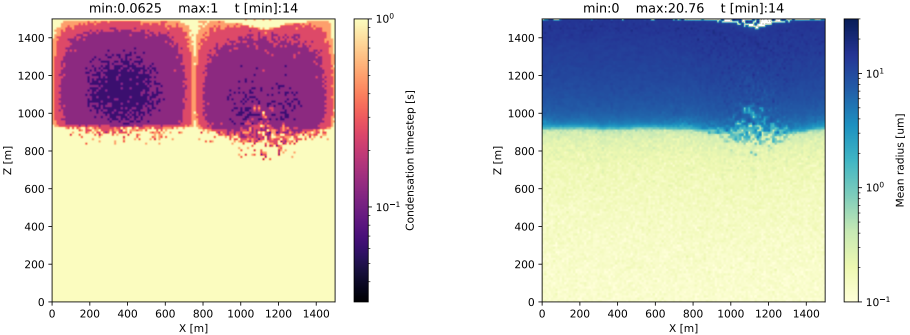
Computational grid: 128x128
Computational particles: 2^{21}

Condensation solver adaptivity



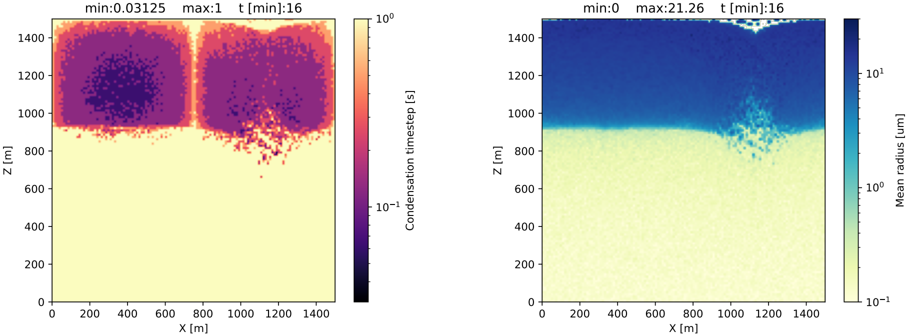
Computational grid: 128x128
Computational particles: 2^{21}

Condensation solver adaptivity



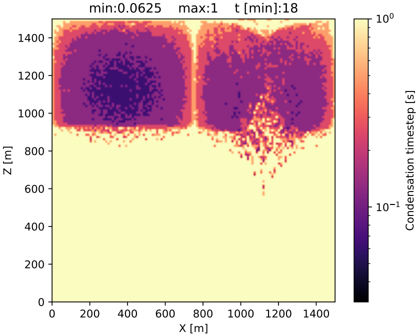
Computational grid: 128x128
Computational particles: 2^{21}

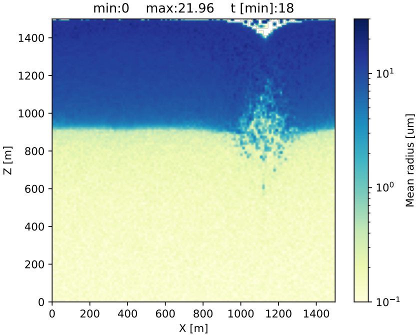
Condensation solver adaptivity



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

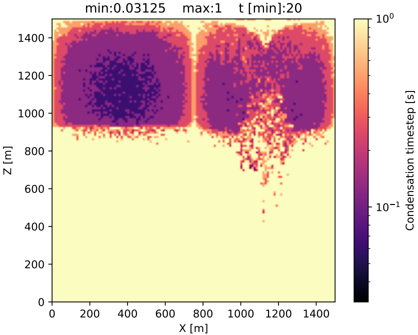
Condensation solver adaptivity

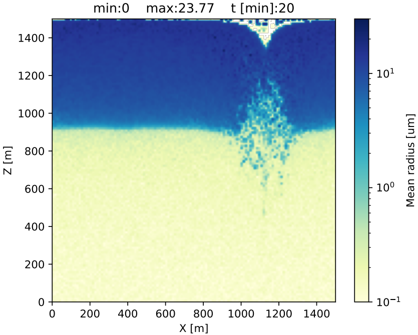




Computational grid: 128x128
Computational particles: 2^{21}

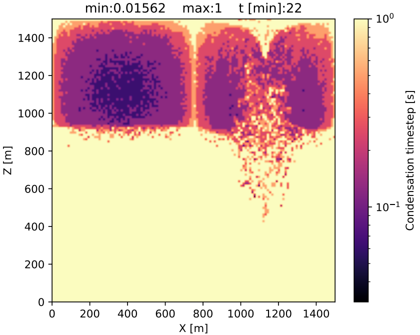
Condensation solver adaptivity

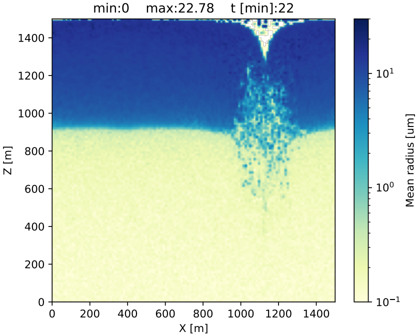




Computational grid: 128x128
Computational particles: 2^{21}

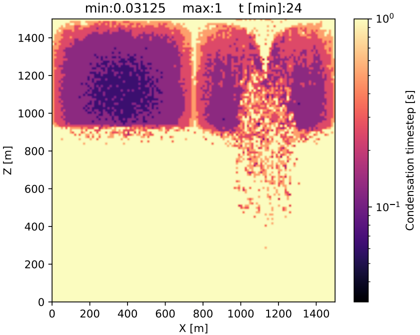
Condensation solver adaptivity

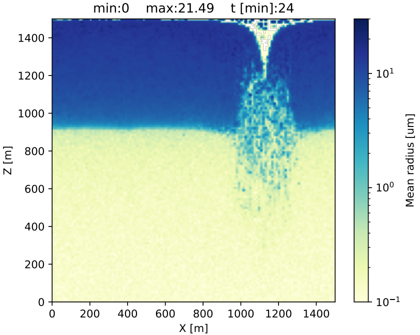




Computational grid: 128x128
Computational particles: 2^{21}

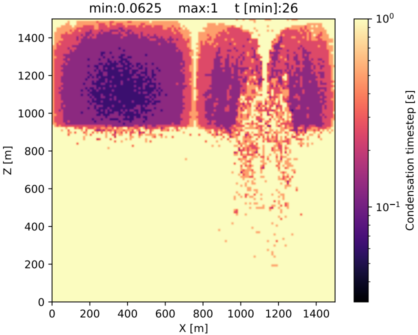
Condensation solver adaptivity

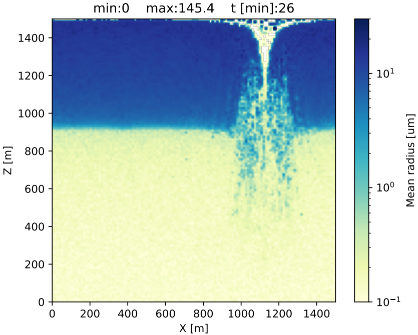




Computational grid: 128x128
Computational particles: 2^{21}

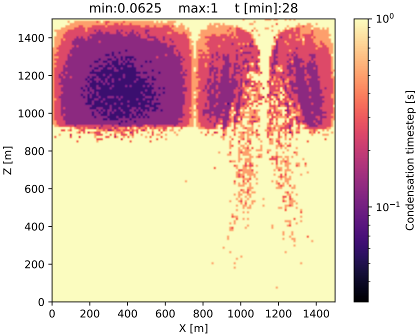
Condensation solver adaptivity

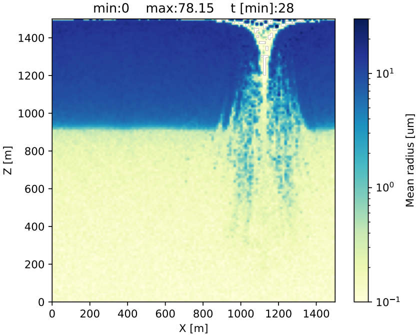




Computational grid: 128x128
Computational particles: 2^{21}

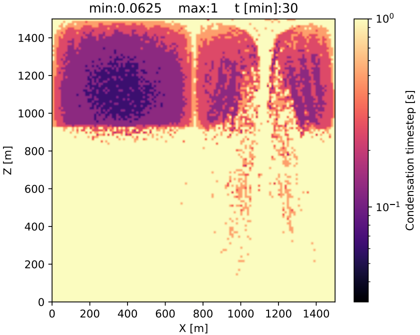
Condensation solver adaptivity

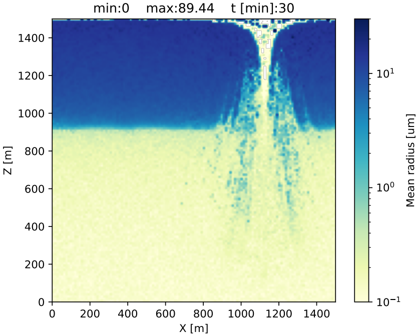




Computational grid: 128x128
Computational particles: 2^{21}

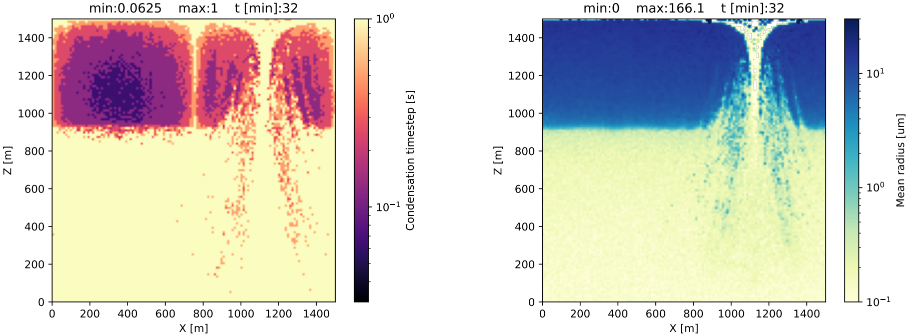
Condensation solver adaptivity





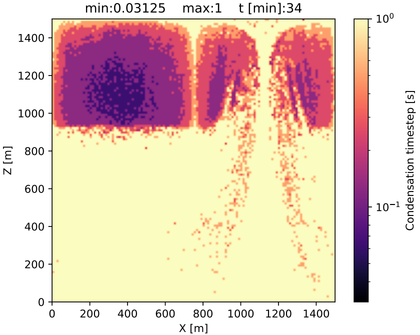
Computational grid: 128x128
Computational particles: 2^{21}

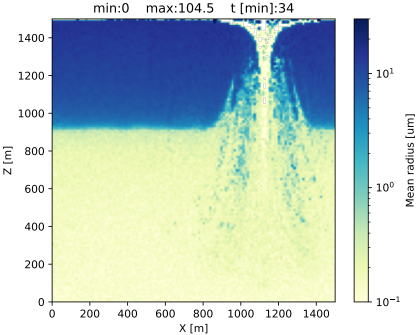
Condensation solver adaptivity



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

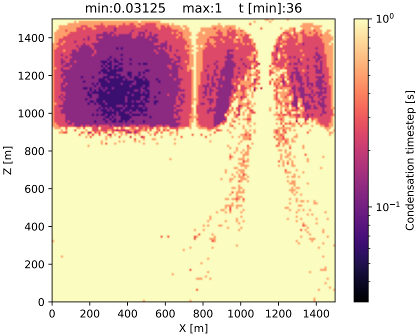
Condensation solver adaptivity

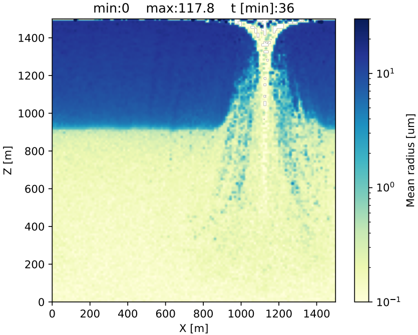




Computational grid: 128x128
Computational particles: 2^{21}

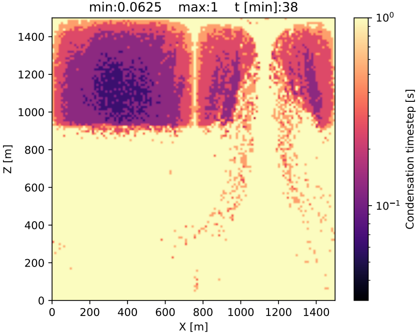
Condensation solver adaptivity

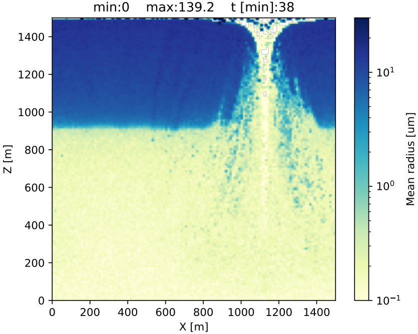




Computational grid: 128x128
Computational particles: 2^{21}

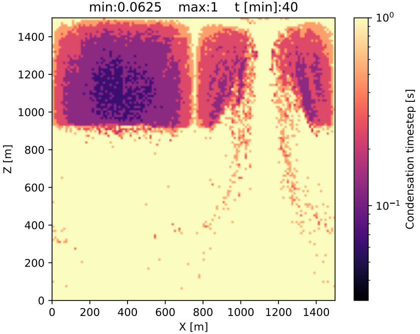
Condensation solver adaptivity

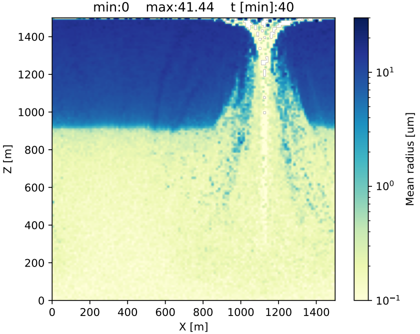




Computational grid: 128x128
Computational particles: 2^{21}

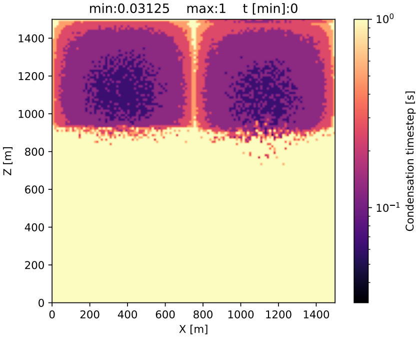
Condensation solver adaptivity

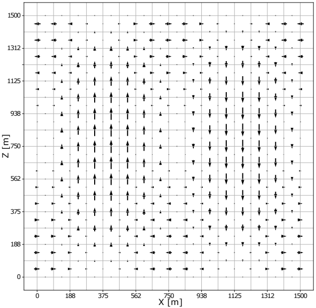




Computational grid: 128x128
Computational particles: 2^{21}

Condensation solver adaptivity





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

Cloud droplet collisional growth

Cloud droplet collisional growth

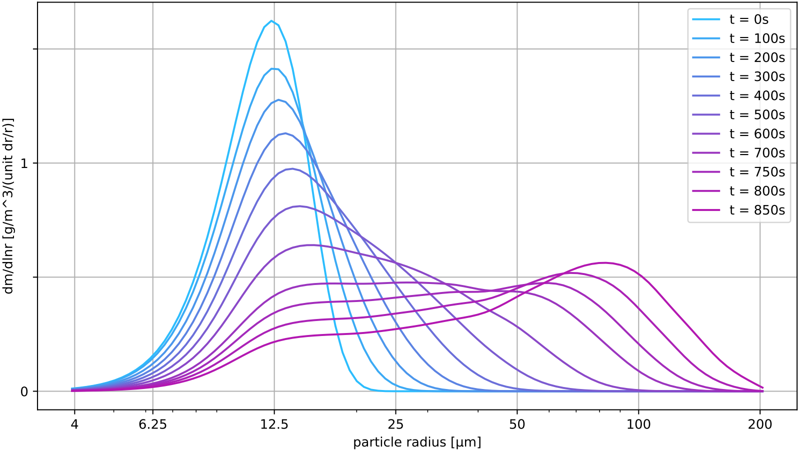
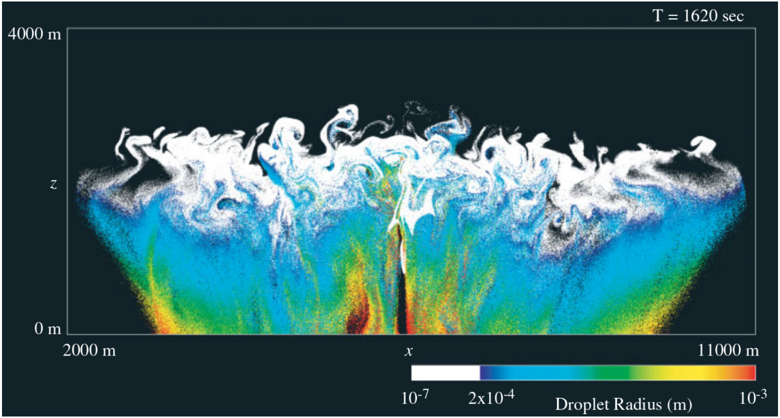


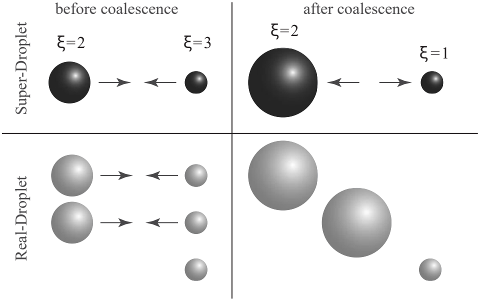
Fig. 5 from Berry 1967 reproduced by PySDM.

Probabilistic particle-based simulations



Super-droplet simulation of a shallow convective cloud
(figure: Shima et al. 2009, QJRMS)

Super-Droplet Method (SDM)



Conceptual view of collision in SDM.
(figure: Shima et al. 2009, QJRMS)

$$\gamma = \left\lceil a(v_{[j]}, v_{[k]}) \frac{\Delta t}{V} \max\{\xi_{[j]}, \xi_{[k]}\} \frac{n_{sd}(n_{sd}-1)/2}{n_{sd}/2} - \phi_\gamma \right\rceil \quad (3)$$

$$\phi_\gamma \sim Uniform[0, 1)$$

assuming $\xi_{[j]} > \xi_{[k]}$ and $\tilde{\gamma} = \min\{\gamma, \lfloor \xi_{[j]}/\xi_{[k]} \rfloor\}$

Super-Droplet Method (SDM)

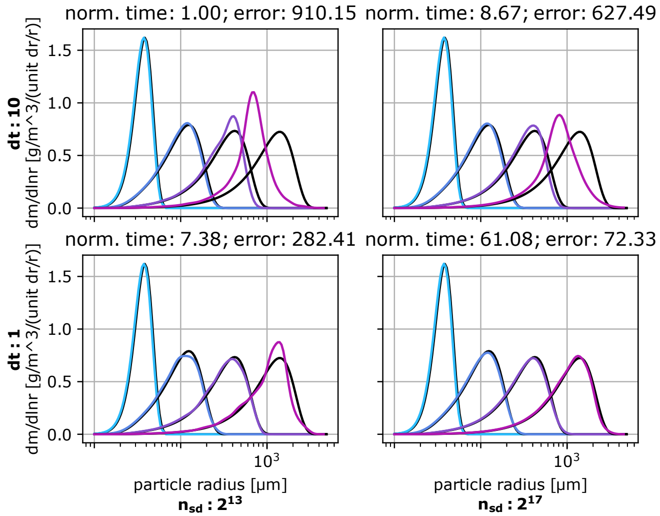
1. $\xi_{[j]} - \tilde{\gamma}\xi_{[k]} > 0$

$$
\begin{aligned}
\hat{\xi}_{[j]} &= \xi_{[j]} - \tilde{\gamma}\xi_{[k]} & \hat{\xi}_{[k]} &= \xi_{[k]} \\
\hat{A}^{ex}_{[j]} &= A^{ex}_{[j]} & \hat{A}^{ex}_{[k]} &= A^{ex}_{[k]} + \tilde{\gamma}A^{ex}_{[j]} \\
\hat{A}^{in}_{[j]} &= A^{in}_{[j]} & \hat{A}^{in}_{[k]} &= \frac{A^{in}_{[k]} v_{[k]} + \tilde{\gamma}A^{in}_{[j]} v_{[j]}}{v_{[k]} + \tilde{\gamma} v_{[j]}}
\end{aligned}
$$

2. $\xi_{[j]} - \tilde{\gamma}\xi_{[k]} = 0$

$$
\begin{aligned}
\hat{\xi}_{[j]} &= \lfloor \xi_{[k]}/2 \rfloor & \hat{\xi}_{[k]} &= \xi_{[k]} - \lfloor \xi_{[k]}/2 \rfloor \\
\hat{A}^{ex}_{[j]} &= \hat{A}^{ex}_{[k]} & \hat{A}^{ex}_{[k]} &= A^{ex}_{[k]} + \tilde{\gamma}A^{ex}_{[j]} \\
\hat{A}^{in}_{[j]} &= \hat{A}^{in}_{[k]} & \hat{A}^{in}_{[k]} &= \frac{A^{in}_{[k]} v_{[k]} + \tilde{\gamma}A^{in}_{[j]} v_{[j]}}{v_{[k]} + \tilde{\gamma} v_{[j]}}
\end{aligned}
$$

SDM: sensitivity to time step



SDM: adaptive time step (proposed in PySDM)

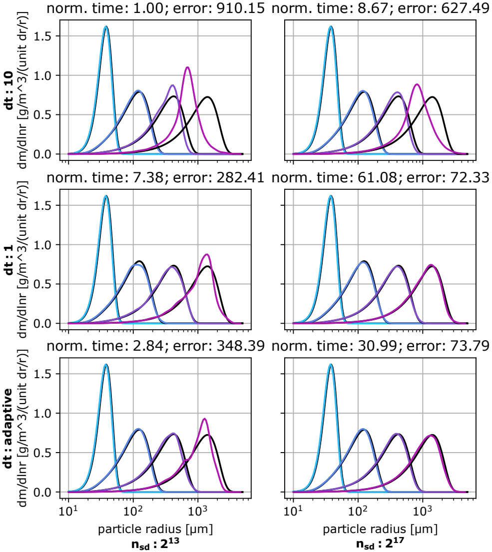
The aim of the proposed adaptivity scheme is to avoid situation when $\gamma > \lfloor \xi_{[j]}/\xi_{[k]} \rfloor$ by adjusting the number of sub-steps k^{n+1} in next major time step based on the collision rates during current major time step:

$$k^{n+1} = \frac{\frac{1}{k^n}\sum\limits_{s=0}^{k^n} w^{n_s} + \max\limits_{s}\{w^{n_s}\}}{2} \tag{4}$$

$$w^{n_s} = \max_{p}\left\{\left\lfloor k^n \gamma_p^{n_s} / \left\lfloor \frac{\xi_{[j]}^{n_s}}{\xi_{[k]}^{n_s}} \right\rfloor \right\rfloor\right\} \tag{5}$$

where $p \in \{(j, k) : j, k \in [0, n_{sd}]$ and j-th and k-th super-droplets are a colliding pair$\}$.

SDM: adaptive time step (proposed in PySDM)



PySDM

- Python `python.org`
- Numba `numba.pydata.org`
- ThrustRTC `pypi.org/project/ThrustRTC`

technological stack and workflows

- Python `python.org`
- Numba `numba.pydata.org`
- ThrustRTC `pypi.org/project/ThrustRTC`
- GitHub & GitHub Actions `github.com`
- TravisCI `travis-ci.org`
- AppVeyor `appveyor.com`

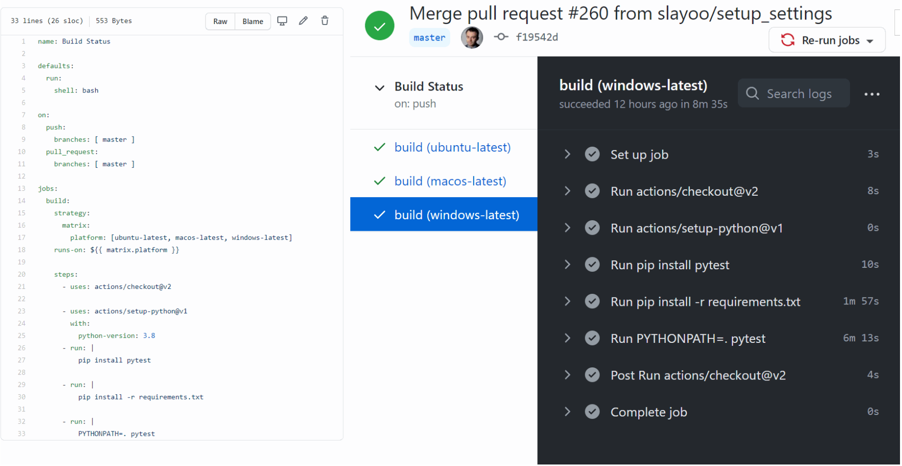
technological stack and workflows

- Python python.org
- Numba numba.pydata.org
- ThrustRTC pypi.org/project/ThrustRTC

- GitHub & GitHub Actions github.com
- TravisCI travis-ci.org
- AppVeyor appveyor.com

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

Portability and Continuous Integration



Demo (github.com/atmos-cloud-sim-uj/PySDM)

README.md

build passing coverage 61%

PySDM

PySDM is a package for simulating the dynamics of population of particles immersed in moist air based (a.k.a. super-droplet) approach to represent aerosol/cloud/rain microphysics. The package high-performance implementation of the Super-Droplet Method (SDM) Monte-Carlo algorithm f collisional growth (Shima et al. 2009), hence the name. PySDM has two alternative parallel numb available: multi-threaded CPU backend based on Numba and GPU-resident backend built on top

Demos:

- Shima et al. 2009 Fig. 2 launch binder Open in Colab
  (Box model, coalescence only, test case employing Golovin analytical solution)
- Berry 1967 Figs. 6, 8, 10 launch binder Open in Colab
  (Box model, coalescence only, test cases for realistic kernels)

Highlights

- New pythonic implementation of SDM
- CPU/GPU parallelization
- Adaptive time stepping schemes
  for coalescence and condensation
- Low entry threshold:
  – for users (Jupyter Notebooks "in the cloud")
  – for developers (2.1 kLOC of tests/ 3.8 kLOC of source)

MORE:

www.ap.uj.edu.pl/diplomas/141204

www.github.com/atmos-cloud-sim-uj/PySDM

www.github.com/piotrbartman

piotr.bartman@doctoral.uj.edu.pl

funding acknowledgment:

Foundation for Polish Science / European Union