

PySDM: Pythonic particle-based cloud microphysics package

Piotr Bartman

Jagiellonian University in Kraków

Introduction



Atmospheric Cloud Simulation Group @ Jagiellonian University

Repositories 4 Packages People 4 Teams Projects 1

Find a repository...

Type: All

Language: All

New

PySDM

Forked from piotrbartman/PySDM

Pythonic particle-based (super-droplet) cloud microphysics modelling with Jupyter examples

atmospheric-modelling monte-carlo-simulation gpu-computing

physics-simulation numba thrust pint

Jupyter Notebook GPL-3.0 11 9 57 2 Updated 5 minutes ago



PyMPDATA

Forked from piotrbartman/PyMPDATA

Numba-accelerated Pythonic implementation of MPDATA with Jupyter examples

atmospheric-modelling numba numerical-integration advection

pde-solver advection-diffusion

Jupyter Notebook GPL-3.0 8 3 44 2 Updated 27 minutes ago



Top languages

Jupyter Notebook

Most used topics

Manage

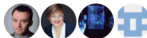
atmospheric-modelling

atmospheric-physics gpu-computing

monte-carlo-simulation numba

People

4 >



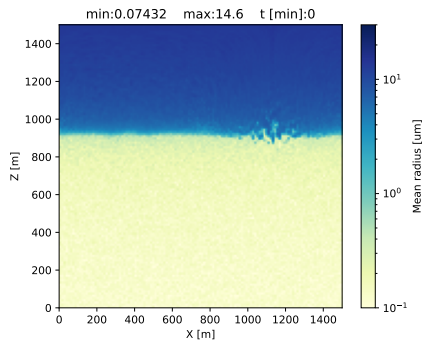
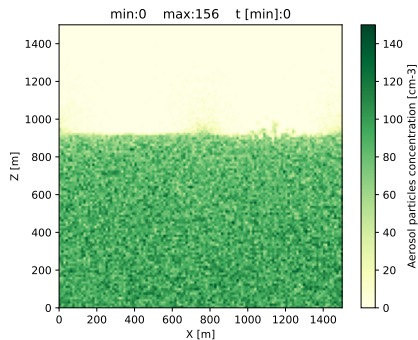
ACSG: Sylwester Arabas, Piotr Bartman, Michael Olesik

Aerosol-cloud-precipitation interactions



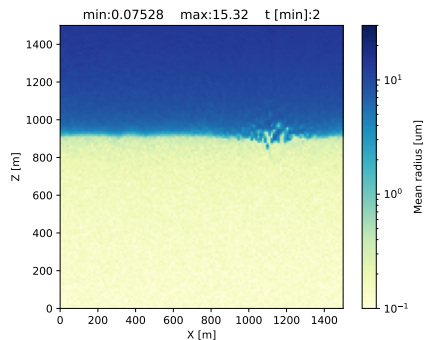
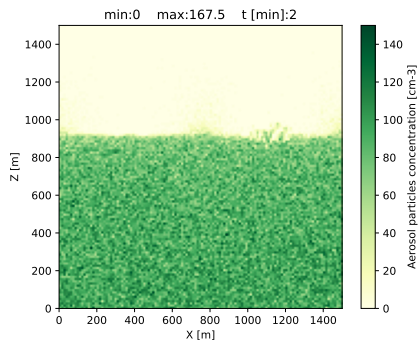
“Cloud and ship. Ukraine, Crimea, Black sea, view from Ai-Petri mountain”
(photo: Yevgen Timashov / National Geographic)

Simulation of stratocumulus and virga phenomenon



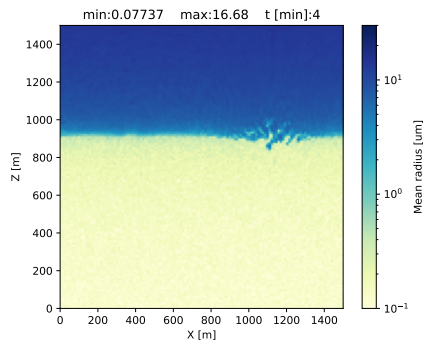
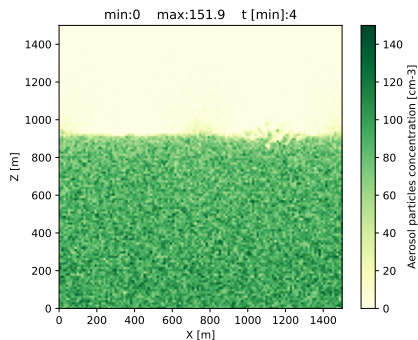
Computational grid: 128x128
Computational particles: 2²¹

Simulation of stratocumulus and virga phenomenon



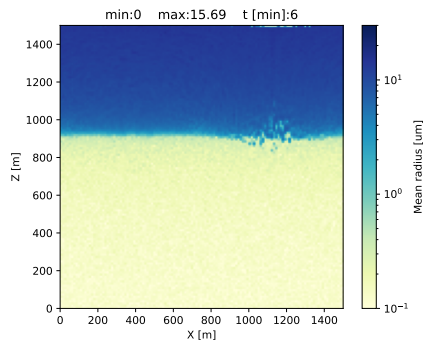
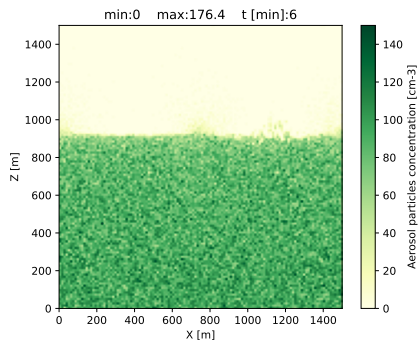
Computational grid: 128x128
Computational particles: 2²¹

Simulation of stratocumulus and virga phenomenon



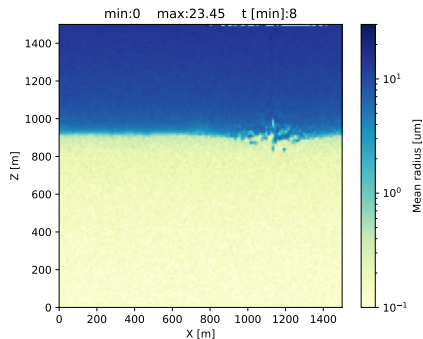
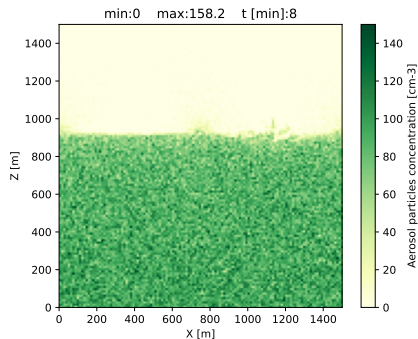
Computational grid: 128x128
Computational particles: 2²¹

Simulation of stratocumulus and virga phenomenon



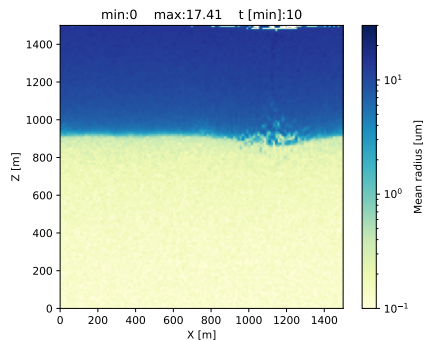
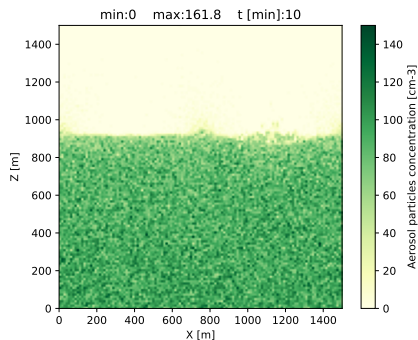
Computational grid: 128x128
Computational particles: 2²¹

Simulation of stratocumulus and virga phenomenon



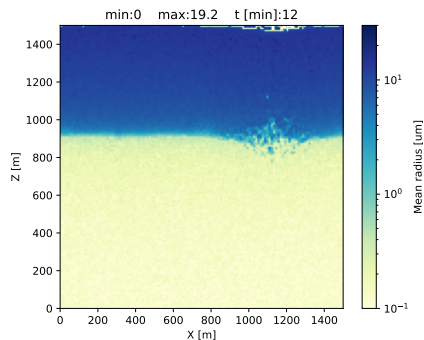
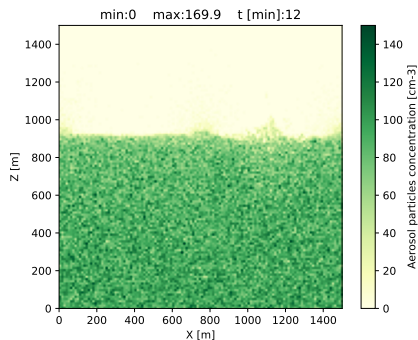
Computational grid: 128x128
Computational particles: 2^{21}

Simulation of stratocumulus and virga phenomenon



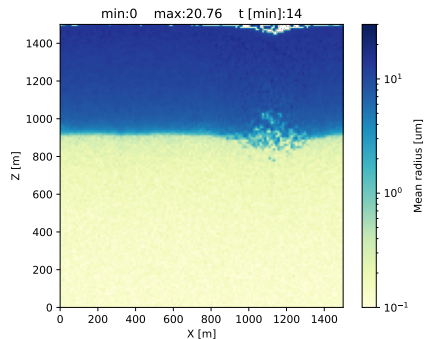
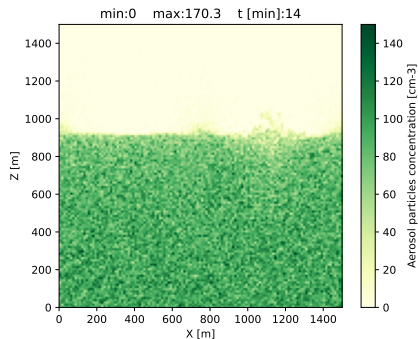
Computational grid: 128x128
Computational particles: 2²¹

Simulation of stratocumulus and virga phenomenon



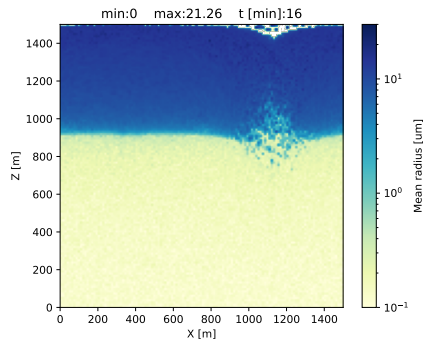
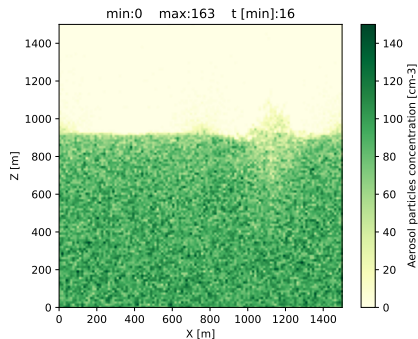
Computational grid: 128x128
Computational particles: 2²¹

Simulation of stratocumulus and virga phenomenon



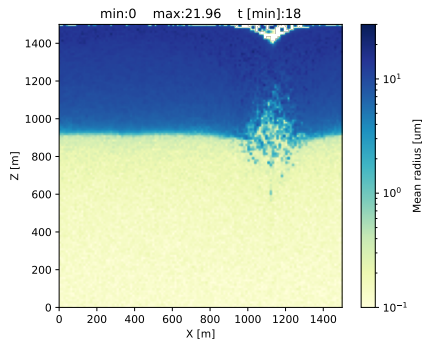
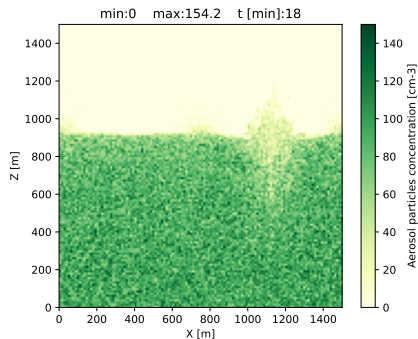
Computational grid: 128x128
Computational particles: 2²¹

Simulation of stratocumulus and virga phenomenon



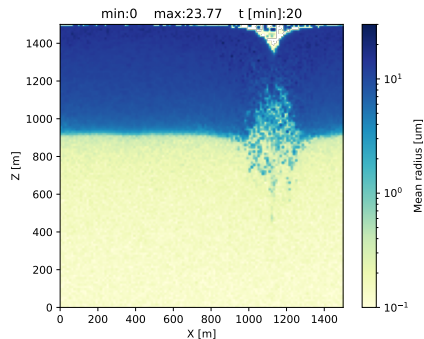
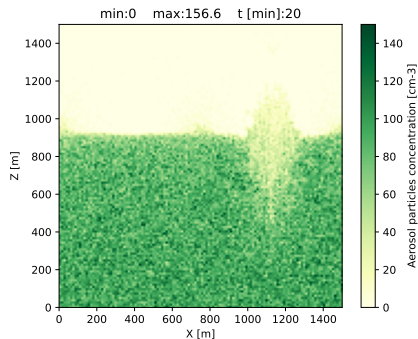
Computational grid: 128x128
Computational particles: 2²¹

Simulation of stratocumulus and virga phenomenon



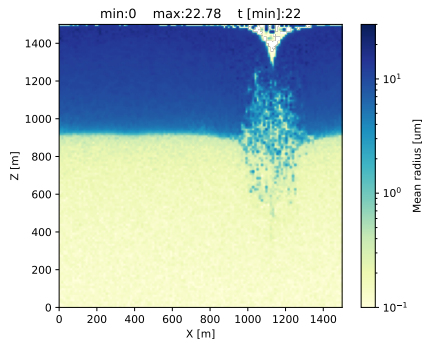
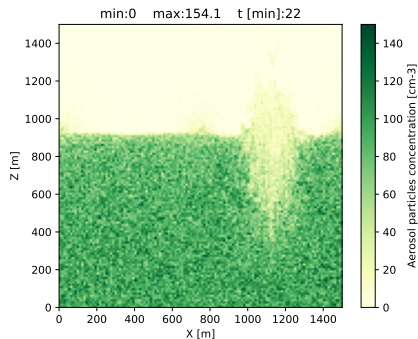
Computational grid: 128x128
Computational particles: 2²¹

Simulation of stratocumulus and virga phenomenon



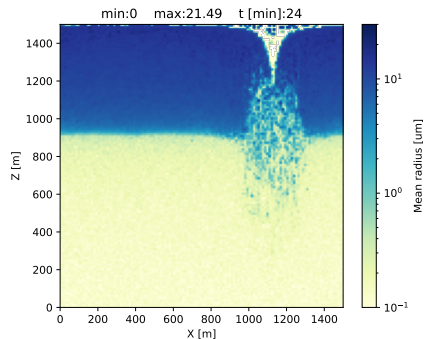
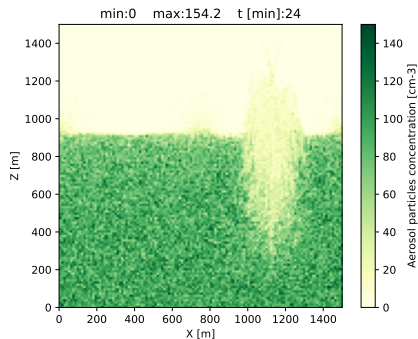
Computational grid: 128x128
Computational particles: 2²¹

Simulation of stratocumulus and virga phenomenon



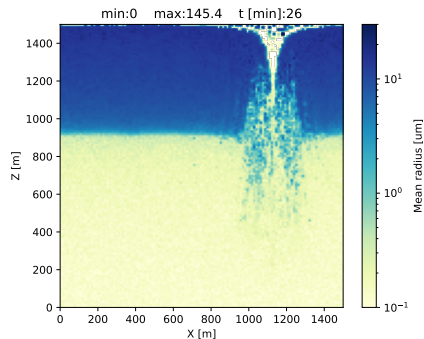
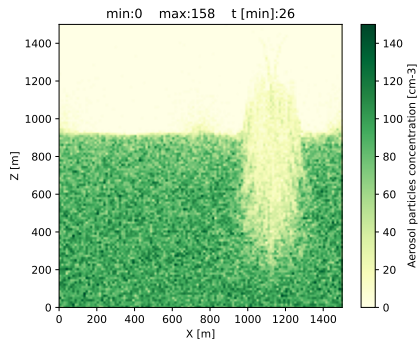
Computational grid: 128x128
Computational particles: 2²¹

Simulation of stratocumulus and virga phenomenon



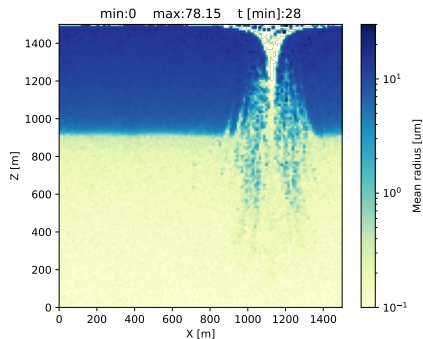
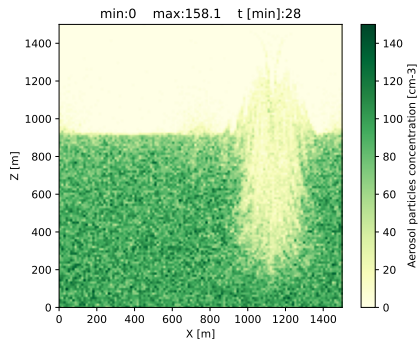
Computational grid: 128x128
Computational particles: 2^{21}

Simulation of stratocumulus and virga phenomenon



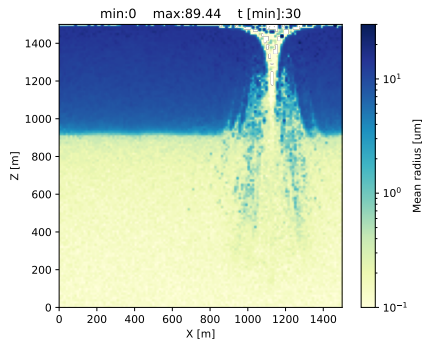
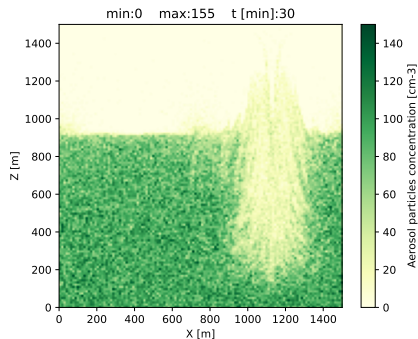
Computational grid: 128x128
Computational particles: 2^{21}

Simulation of stratocumulus and virga phenomenon



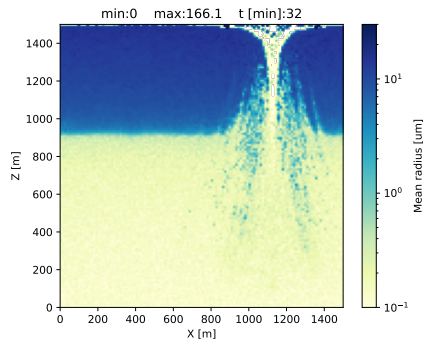
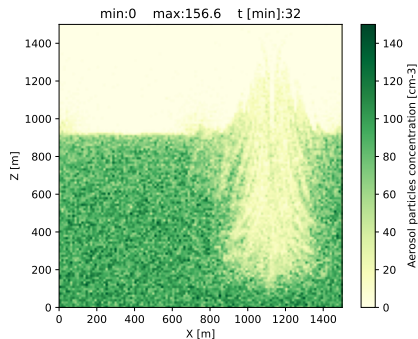
Computational grid: 128x128
Computational particles: 2²¹

Simulation of stratocumulus and virga phenomenon



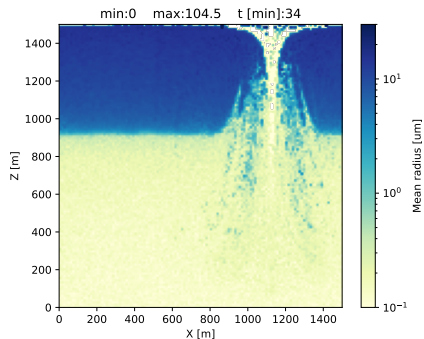
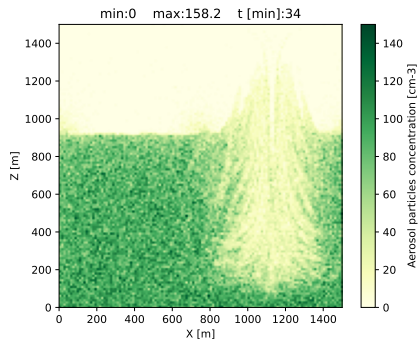
Computational grid: 128x128
Computational particles: 2²¹

Simulation of stratocumulus and virga phenomenon



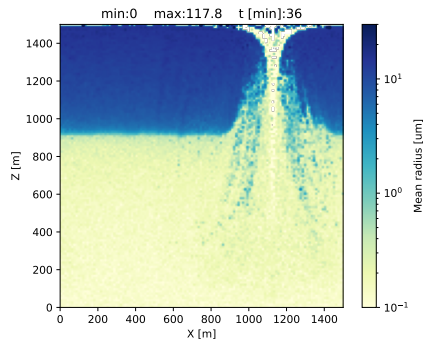
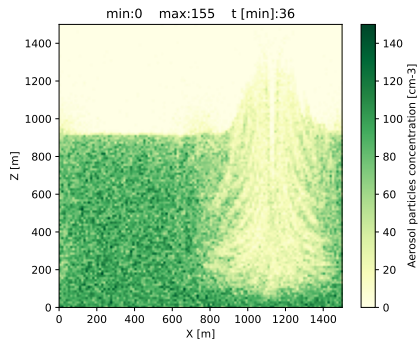
Computational grid: 128x128
Computational particles: 2²¹

Simulation of stratocumulus and virga phenomenon



Computational grid: 128x128
Computational particles: 2²¹

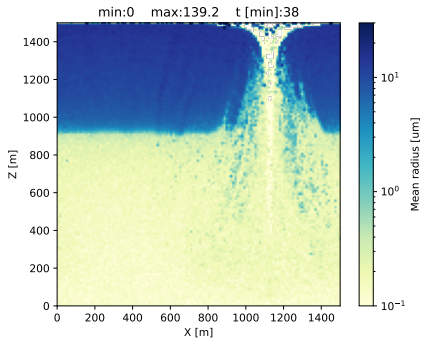
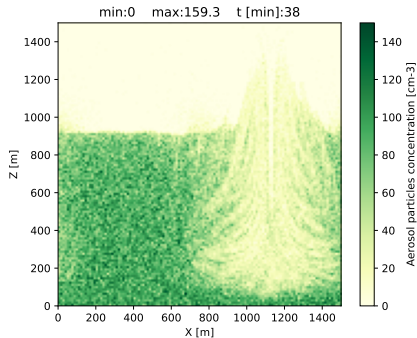
Simulation of stratocumulus and virga phenomenon



Computational grid: 128x128

Computational particles: 2²¹

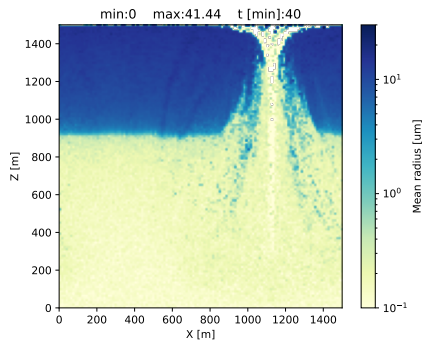
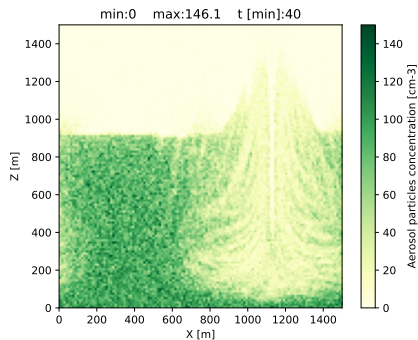
Simulation of stratocumulus and virga phenomenon



Computational grid: 128x128

Computational particles: 2²¹

Simulation of stratocumulus and virga phenomenon



Computational grid: 128x128

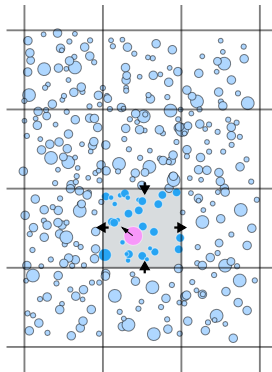
Computational particles: 2²¹

- ❖ 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:

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

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)

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

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

Evolution of the environment (in each cell)

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

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

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

Evolution of the environment (in each cell)

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

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

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

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

Evolution of the environment (in each cell)

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

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

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

$$\dot{q}_{env}(x, z) = -\rho_d^{-1} \nabla \cdot (\vec{u}\rho_d q_v)$$

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

Evolution of the environment (in each cell)

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

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

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

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

$$\dot{q}_{env}(x, z) = -\rho_d^{-1} \nabla \cdot (\vec{u} \rho_d q_v)$$

$$\dot{\theta}_{env}(x, z) = -\rho_d^{-1} \nabla \cdot (\vec{u} \rho_d \theta_d)$$

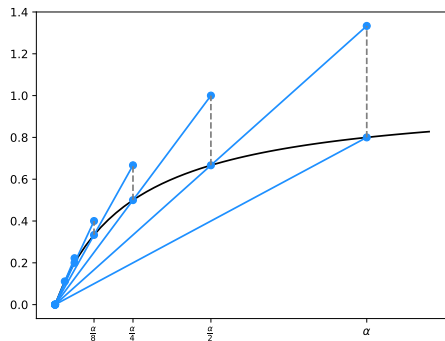
Adaptive time step (proposed in PySDM)

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:

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

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

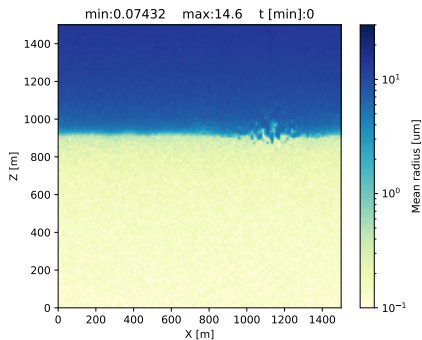
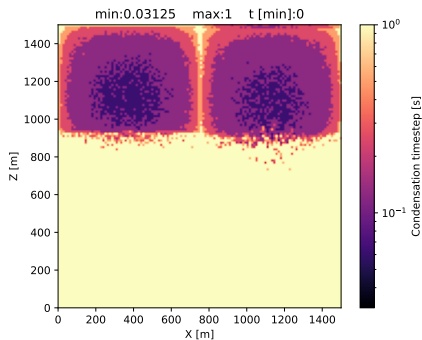
Adaptive time step (proposed in PySDM)



Conceptual view of inequality (2).

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

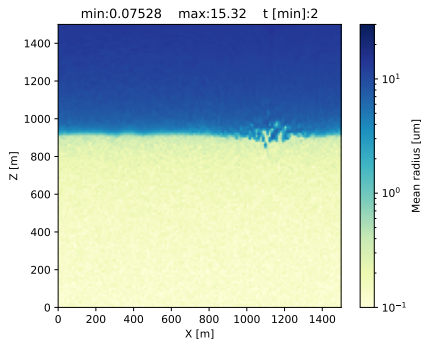
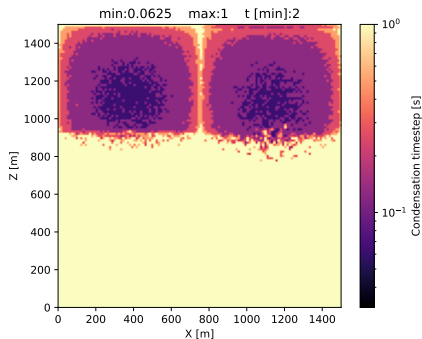
Condensation solver adaptivity



Computational grid: 128x128

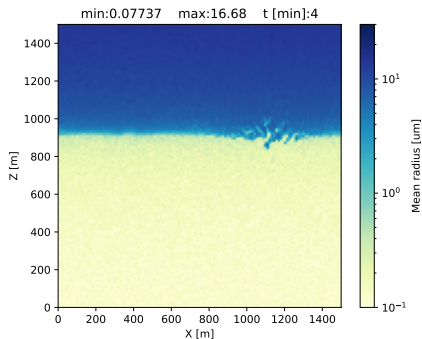
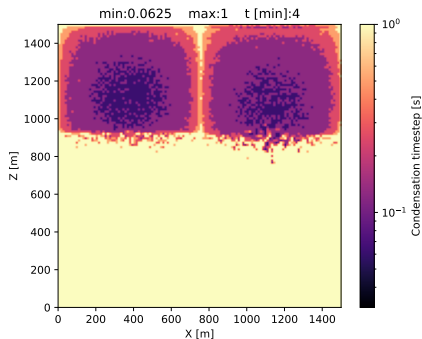
Computational particles: 2^{21}

Condensation solver adaptivity



Computational grid: 128x128
Computational particles: 2²¹

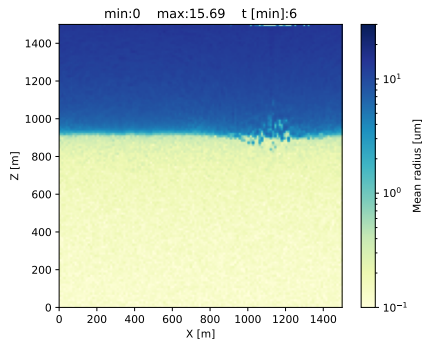
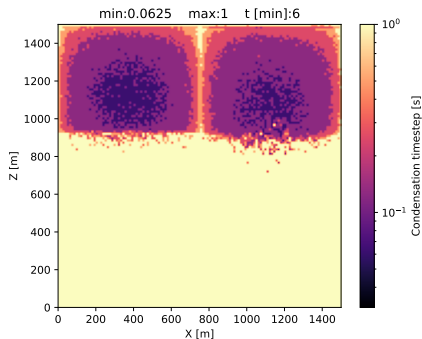
Condensation solver adaptivity



Computational grid: 128x128

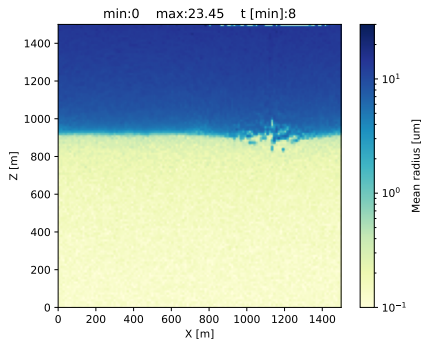
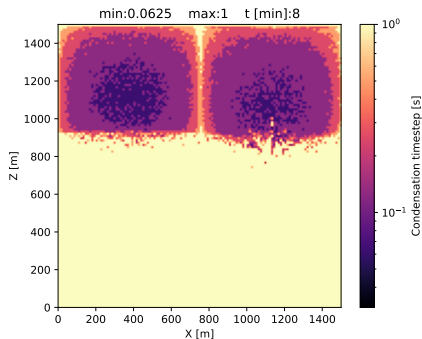
Computational particles: 2^{21}

Condensation solver adaptivity



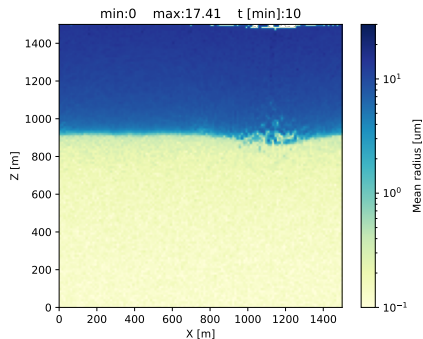
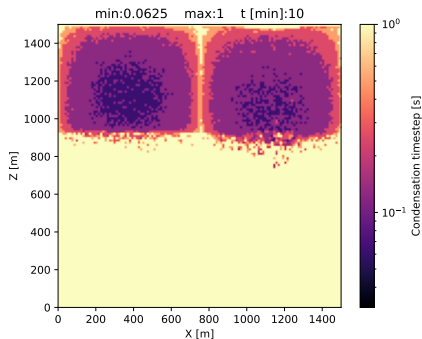
Computational grid: 128x128
Computational particles: 2²¹

Condensation solver adaptivity



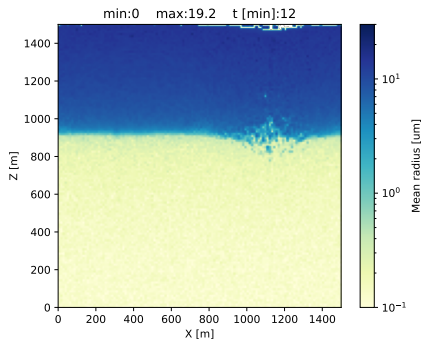
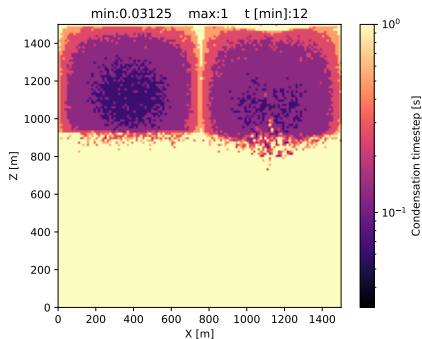
Computational grid: 128x128
Computational particles: 2^{21}

Condensation solver adaptivity



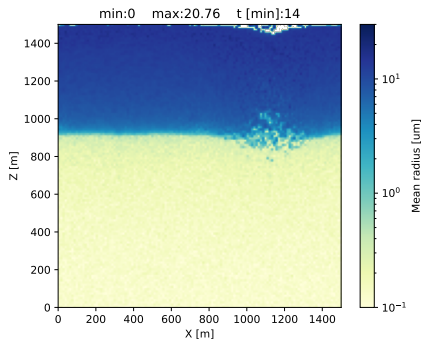
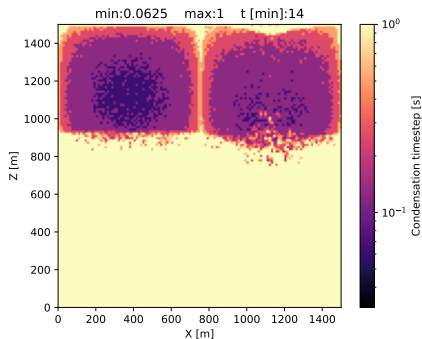
Computational grid: 128x128
Computational particles: 2²¹

Condensation solver adaptivity



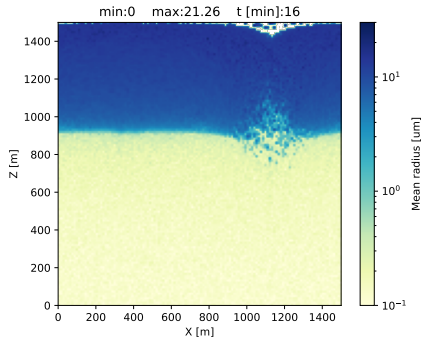
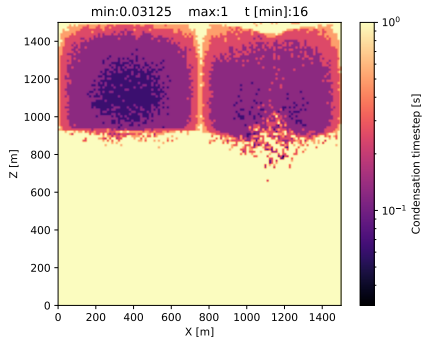
Computational grid: 128x128
Computational particles: 2^{21}

Condensation solver adaptivity



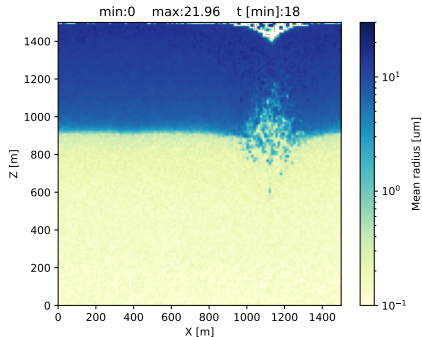
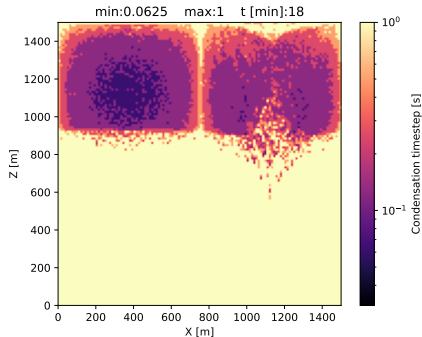
Computational grid: 128x128
Computational particles: 2²¹

Condensation solver adaptivity



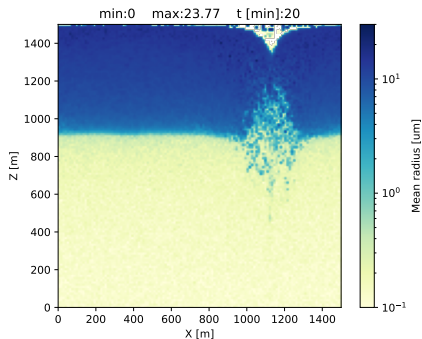
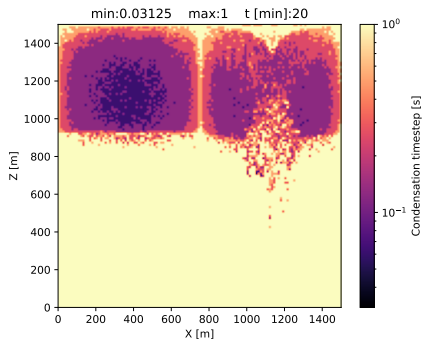
Computational grid: 128x128
Computational particles: 2²¹

Condensation solver adaptivity



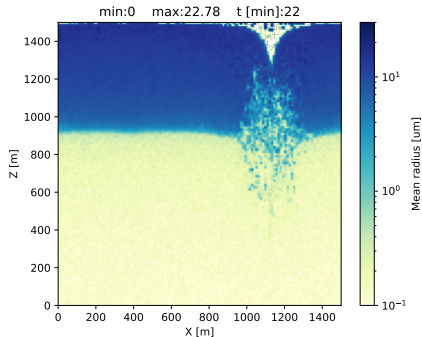
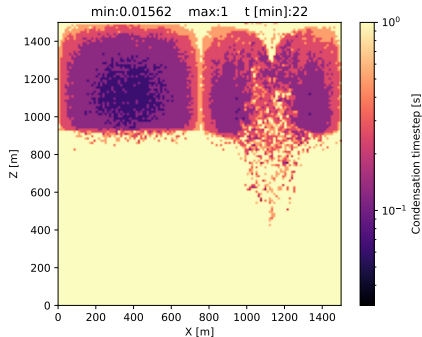
Computational grid: 128x128
Computational particles: 2²¹

Condensation solver adaptivity



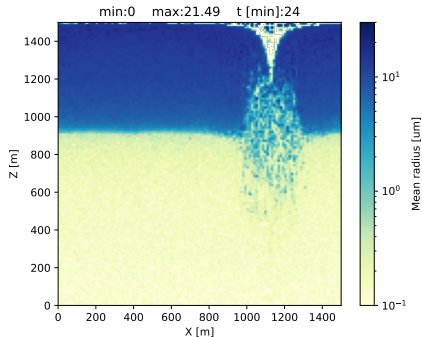
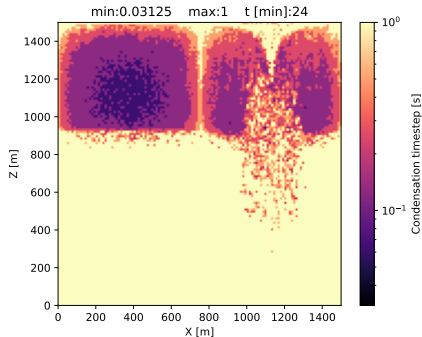
Computational grid: 128x128
Computational particles: 2²¹

Condensation solver adaptivity



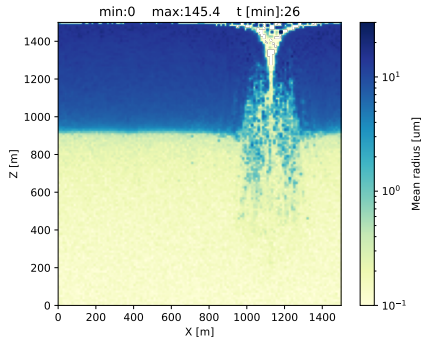
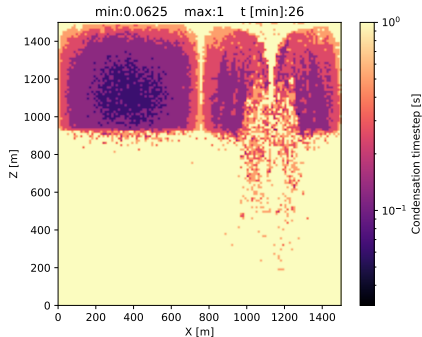
Computational grid: 128x128
Computational particles: 2^{21}

Condensation solver adaptivity



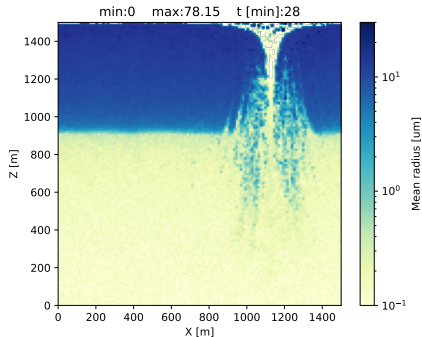
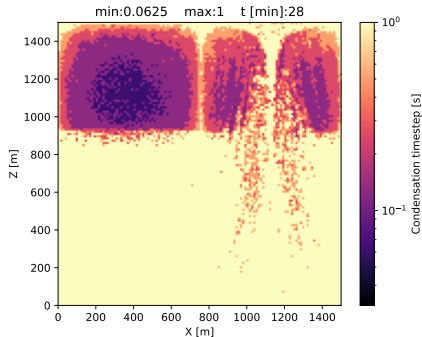
Computational grid: 128x128
Computational particles: 2^{21}

Condensation solver adaptivity



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

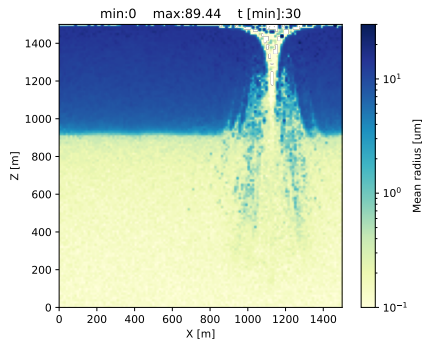
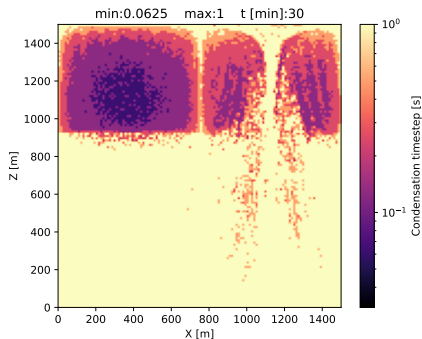
Condensation solver adaptivity



Computational grid: 128x128

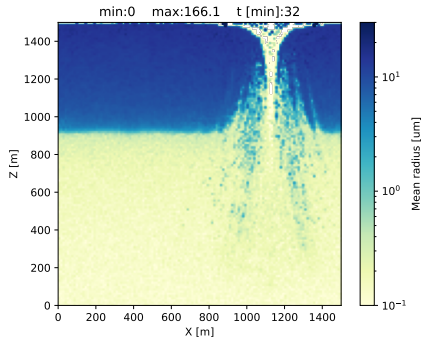
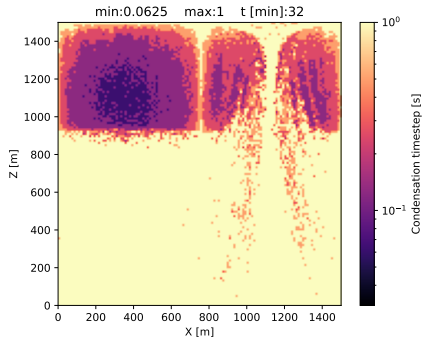
Computational particles: 2^{21}

Condensation solver adaptivity



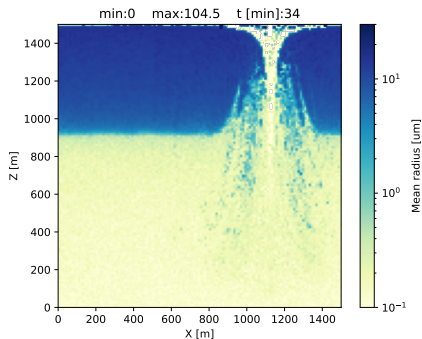
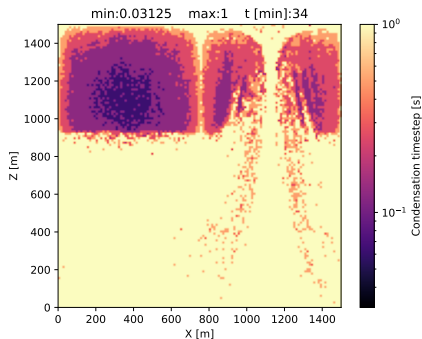
Computational grid: 128x128
Computational particles: 2^{21}

Condensation solver adaptivity



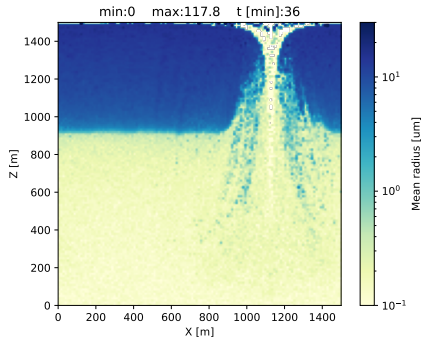
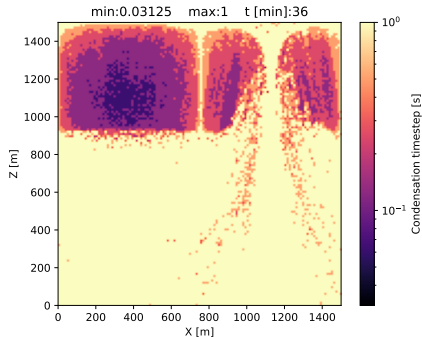
Computational grid: 128x128
Computational particles: 2^{21}

Condensation solver adaptivity



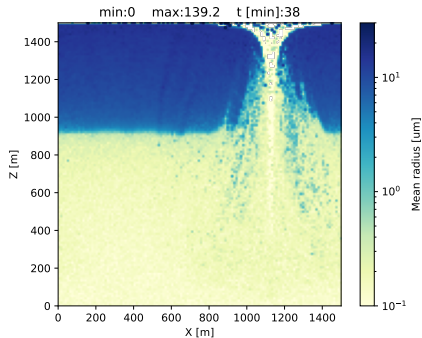
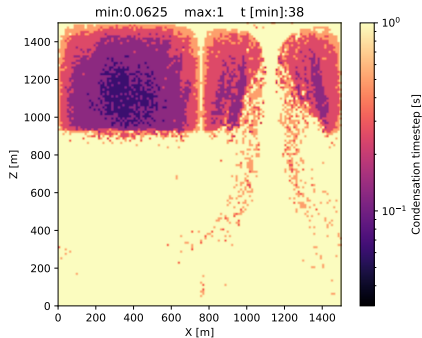
Computational grid: 128x128
Computational particles: 2^{21}

Condensation solver adaptivity



Computational grid: 128x128
Computational particles: 2²¹

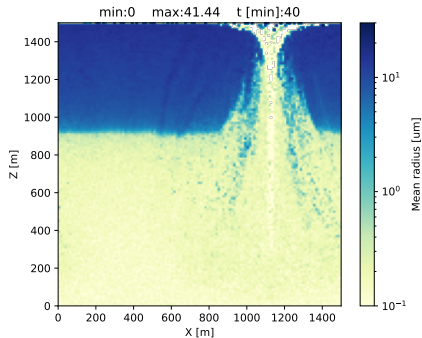
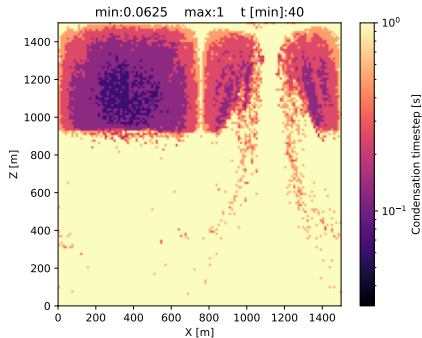
Condensation solver adaptivity



Computational grid: 128x128

Computational particles: 2^{21}

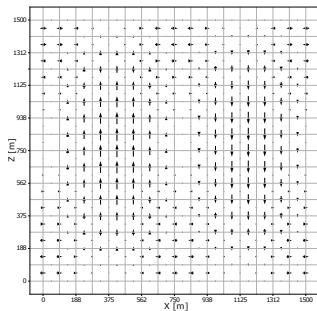
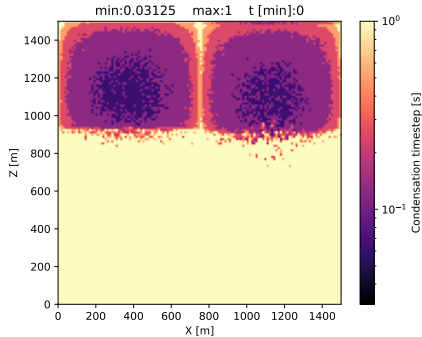
Condensation solver adaptivity



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

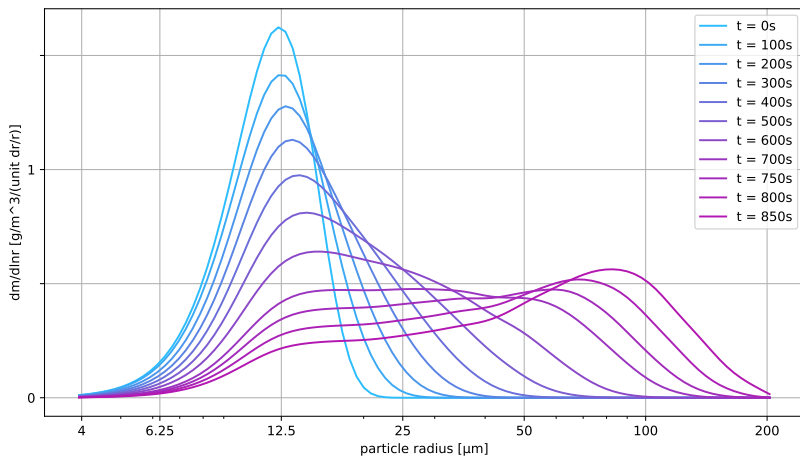
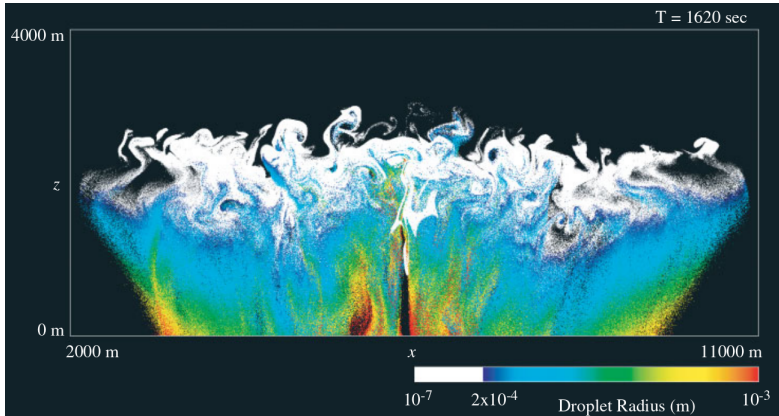


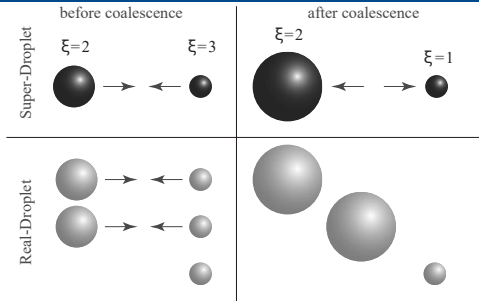
Fig. 5 from Berry 1967 reproduced by PySDM.

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)

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

$$\phi_\gamma \sim \text{Uniform}[0, 1)$$

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

Super-Droplet Method (SDM)

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

$$\hat{\xi}_{[j]} = \xi_{[j]} - \tilde{\gamma}\xi_{[k]}$$

$$\hat{A}_{[j]}^{\text{ex}} = A_{[j]}^{\text{ex}}$$

$$\hat{A}_{[j]}^{\text{in}} = A_{[j]}^{\text{in}}$$

$$\hat{\xi}_{[k]} = \xi_{[k]}$$

$$\hat{A}_{[k]}^{\text{ex}} = A_{[k]}^{\text{ex}} + \tilde{\gamma}A_{[j]}^{\text{ex}}$$

$$\hat{A}_{[k]}^{\text{in}} = \frac{A_{[k]}^{\text{in}}v_{[k]} + \tilde{\gamma}A_{[j]}^{\text{in}}v_{[j]}}{v_{[k]} + \tilde{\gamma}v_{[j]}}$$

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

$$\hat{\xi}_{[j]} = \lfloor \xi_{[k]}/2 \rfloor$$

$$\hat{A}_{[j]}^{\text{ex}} = \hat{A}_{[k]}^{\text{ex}}$$

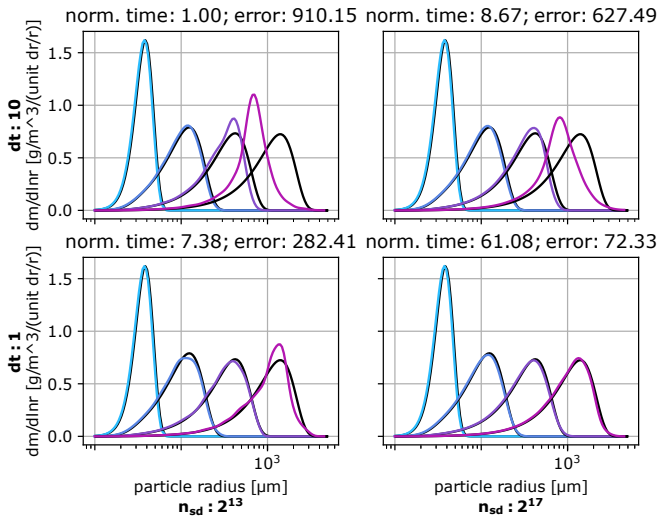
$$\hat{A}_{[j]}^{\text{in}} = \hat{A}_{[k]}^{\text{in}}$$

$$\hat{\xi}_{[k]} = \xi_{[k]} - \lfloor \xi_{[k]}/2 \rfloor$$

$$\hat{A}_{[k]}^{\text{ex}} = A_{[k]}^{\text{ex}} + \tilde{\gamma}A_{[j]}^{\text{ex}}$$

$$\hat{A}_{[k]}^{\text{in}} = \frac{A_{[k]}^{\text{in}}v_{[k]} + \tilde{\gamma}A_{[j]}^{\text{in}}v_{[j]}}{v_{[k]} + \tilde{\gamma}v_{[j]}}$$

SDM: sensitivity to time step



SDM: adaptive time step (proposed in PySDM)

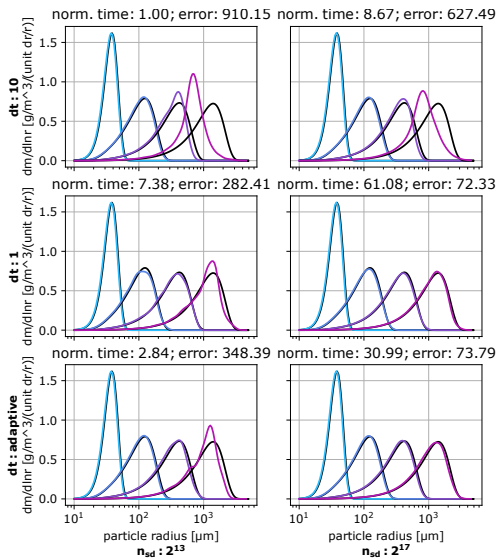
The aim of the proposed adaptivity scheme is to avoid situation when $\gamma > \lfloor \xi_{[j]} / \xi_{[k]} \rfloor$ by adjusting the number of sub-steps k^{n+1} in next major time step based on the collision rates during current major time step:

$$k^{n+1} = \frac{\frac{1}{k^n} \sum_{s=0}^{k^n} w^{n_s} + \max_s \{w^{n_s}\}}{2} \quad (4)$$

$$w^{n_s} = \max_p \left\{ \left[k^n \gamma_p^{n_s} / \left[\frac{\xi_{[j]}^{n_s}}{\xi_{[k]}^{n_s}} \right] \right] \right\} \quad (5)$$

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

SDM: adaptive time step (proposed in PySDM)



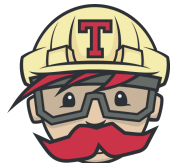
PySDM

- ❖ 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



- ❖ 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

```
33 lines (26 sloc) 553 Bytes
Raw Blame
1 name: Build Status
2
3 defaults:
4 run:
5   shell: bash
6
7 on:
8   push:
9     branches: [ master ]
10  pull_request:
11    branches: [ master ]
12
13 jobs:
14  build:
15    strategy:
16      matrix:
17        platform: [ubuntu-latest, macos-latest, windows-latest]
18    runs-on: ${matrix.platform}
19
20 steps:
21   - uses: actions/checkout@v2
22
23   - uses: actions/setup-python@v1
24     with:
25       python-version: 3.8
26   - run: |
27     pip install pytest
28
29   - run: |
30     pip install -r requirements.txt
31
32   - run: |
33     PYTHONPATH=. pytest
```

Merge pull request #260 from slayoo/setup_settings

master f19542d

Re-run jobs

Build Status
on: push

- build (ubuntu-latest)
- build (macos-latest)
- build (windows-latest)**

build (windows-latest)
succeeded 12 hours ago in 8m 35s

Search logs

- Set up job 3s
- Run actions/checkout@v2 8s
- Run actions/setup-python@v1 0s
- Run pip install pytest 10s
- Run pip install -r requirements.txt 1m 57s
- Run PYTHONPATH=. pytest 6m 13s
- Post Run actions/checkout@v2 4s
- Complete job 0s

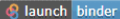

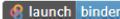
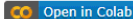
README.md

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 for collisional growth (Shima et al. 2009), hence the name. PySDM has two alternative parallel backends available: multi-threaded CPU backend based on [Numba](#) and GPU-resident backend built on top

Demos:

- [Shima et al. 2009 Fig. 2](#)  
(Box model, coalescence only, test case employing Golovin analytical solution)
- [Berry 1967 Figs. 6, 8, 10](#)  
(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