Modelling of water isotopic fractionation within and below clouds

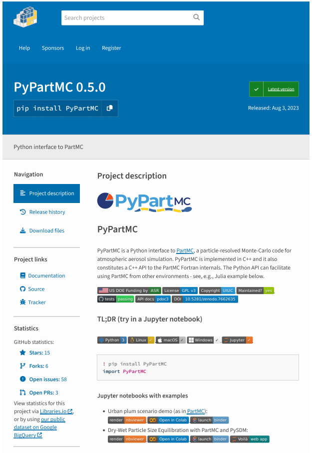
Sylwester Arabas

AGH University in Krakow

github.com/open-atmos


Search projects
Help Sponsors Log in Register
PySDM 2.20
pip install PySDM
Latest version
Released: Apr 21, 2023
Pythonic particle-based (super-droplet) warm-rain/aqueous-chemistry cloud microphysics package with box, parcel & 1D/2D prescribed-flow examples in Python, Julia and Matlab
Navigation
Project description
Release history
Download files
Project links
Homepage
Documentation
Source
Tracker
Statistics
GitHub statistics:
Stars: 40
Forks: 21
Open issues: 101
Open PRs: 13
Project description
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.7851352
EU Funding by FNP | PL Funding by NCN | US DOE Funding by ASR
License GPL v3
tests+artifacts+pypi passing | build passing | codecov 78%
pypi package 2.20 | 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


Search projects
Help Sponsors Log in Register
PyMPDATA 1.0.11
pip install PyMPDATA
Latest version
Released: Apr 26, 2023
Numba-accelerated Pythonic implementation of MPDATA with examples in Python, Julia and Matlab
Navigation
Project description
Release history
Download files
Project links
Documentation
Source
Tracker
Statistics
GitHub statistics:
Stars: 19
Forks: 10
Open issues: 25
Open PRs: 3
View statistics for this project via Libraries.io,
Project description
PyMPDATA
Python 3 | LLVM Numba | Linux | macOS | Windows | Jupyter | Maintained? yes | Open Hub PyMPDATA
JOSS 10.21105/joss.03896 | DOI 10.5281/zenodo.7864655
EU Funding by FNP | PL Funding by NCN | License GPL v3
tests+pypi passing | build passing | codecov 92%
pypi package 1.0.11 | API docs pdoc3
PyMPDATA is a high-performance Numba-accelerated Pythonic implementation of the MPDATA algorithm of Smolarkiewicz et al. used in geophysical fluid dynamics and beyond. MPDATA numerically solves generalised transport equations - partial differential equations used to model conservation/balance laws, scalar-transport problems, convection-diffusion phenomena. As of the current version, PyMPDATA supports homogeneous transport in 1D, 2D and 3D using structured meshes, optionally generalised by employment of a Jacobian of coordinate transformation. PyMPDATA includes implementation of a set of MPDATA variants including the non-oscillatory option, infinite-gauge, divergent-flow, double-pass donor cell (DPDC) and third-order-terms options. It also features support for integration of Fickian-terms in advection-diffusion problems using the pseudo-transport velocity approach. In 2D and 3D simulations, domain-decomposition is used for multi-threaded parallelism.
PyMPDATA is engineered purely in Python targeting both performance and usability, the latter encompassing research users', developers' and maintainers' perspectives. From researcher's perspective, PyMPDATA offers hassle-free


Search projects
Help Sponsors Log in Register
PyPartMC 0.5.0
pip install PyPartMC
Latest version
Released: Aug 3, 2023
Python interface to PartMC
Navigation
Project description
Release history
Download files
Project links
Documentation
Source
Tracker
Statistics
GitHub statistics:
Stars: 15
Forks: 6
Open issues: 58
Open PRs: 3
View statistics for this project via Libraries.io, or by using our public dataset on Google BigQuery
Project description
PyPartMC
PyPartMC
PyPartMC is a Python interface to PartMC, a particle-resolved Monte-Carlo code for atmospheric aerosol simulation. PyPartMC is implemented in C++ and it also constitutes a C++ API to the PartMC Fortran internals. The Python API can facilitate using PartMC from other environments - see, e.g., Julia example below.
US DOE Funding by ASR | License GPL v3 | Copyright UIUC | Maintained? yes
tests passing | API docs pdoc3 | DOI 10.5281/zenodo.7662635
TL;DR (try in a Jupyter notebook)
Python 3 | Linux | macOS | Windows | Jupyter
! pip install PyPartMC
import PyPartMC
Jupyter notebooks with examples
• Urban plum scenario demo (as in PartMC):
render nbviewer | Open in Colab | launch binder
• Dry-Wet Particle Size Equilibration with PartMC and PySDM:
render nbviewer | Open in Colab | launch binder | Voilà web app

Quarterly Journal of the Royal Meteorological Society

Research Article

The super-droplet method for the numerical simulation of clouds and precipitation: a particle-based and probabilistic microphysics model coupled with a non-hydrostatic model

S. Shima, K. Kusano, A. Kawano, T. Sugiyama, S. Kawahara

First published: 19 June 2009 | https://doi.org/10.1002/qj.441 | Citations: 150

Abstract

A novel, particle-based, probabilistic approach for the simulation of cloud microphysics is proposed, which is named the super-droplet method (SDM). This method enables the accurate simulation of cloud microphysics with a less demanding cost in computation. SDM is applied to a warm-cloud system, which incorporates sedimentation, condensation/evaporation and stochastic coalescence. The methodology to couple super-droplets and a non-hydrostatic model is also developed. It is confirmed that the result of our Monte Carlo scheme for the stochastic coalescence of super-droplets agrees fairly well with the solutions of the stochastic coalescence equation. The behaviour of the model is evaluated using a simple test problem, that of a shallow maritime cumulus formation initiated by a warm bubble. Possible extensions of SDM are briefly discussed. A theoretical analysis suggests that the computational cost of SDM becomes lower than the spectral (bin) method when the number of attributes—the variables that identify the state of each super-droplet—becomes larger than some critical value, which we estimate to be in the range $2 \sim 4$.

PySDM: open-source particle-based cloud-microphysics package

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

JOSS 10.21105/joss.03219 | DOI 10.5281/zenodo.11186158

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

License GPL v3

tests+artifacts+pypi passing | build passing | codecov 84%

pypi package 2.54 | 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.

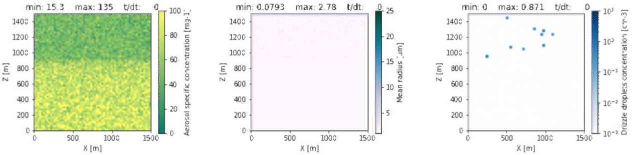
For an overview of PySDM features (and the preferred way to cite PySDM in papers), please refer to our JOSS papers:

- Bartman et al. 2022 (PySDM v1).
- de Jong, Singer et al. 2023 (PySDM v2).

PySDM includes an extension of the SDM scheme to represent collisional breakup described in de Jong, Mackay et al. 2023.
For a list of talks and other materials on PySDM as well as a list of published papers featuring PySDM simulations, see the project wiki.

Example Jupyter notebooks (reproducing results from literature):

See PySDM-examples README.



Pythonic particle-based (super-droplet) warm-rain/aqueous-chemistry cloud microphysics package with box, parcel & 1D/2D prescribed-flow examples in Python, Julia and Matlab

GPL-3.0 license

51 stars

5 watching

28 forks

Report repository

Releases 86

PySDM v2.56 Latest
4 hours ago

+ 85 releases

Contributors 19

+ 5 contributors

Languages

- Python 96.8%
- Jupyter Notebook 2.0%
- TeX 1.2%

PySDM: open-source particle-based cloud-microphysics package

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

JOSS 10.21105/joss.03219 | DOI 10.5281/zenodo.11186158

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

License GPL v3

tests+artifacts+pypi passing | build passing | codecov 84%

pypi package 2.54 | 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.

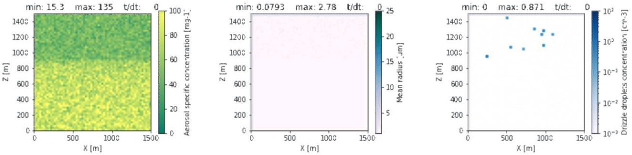
For an overview of PySDM features (and the preferred way to cite PySDM in papers), please refer to our JOSS papers:

- Bartman et al. 2022 (PySDM v1).
- de Jong, Singer et al. 2023 (PySDM v2).

PySDM includes an extension of the SDM scheme to represent collisional breakup described in de Jong, Mackay et al. 2023.
For a list of talks and other materials on PySDM as well as a list of published papers featuring PySDM simulations, see the project wiki.

Example Jupyter notebooks (reproducing results from literature):

See PySDM-examples README.



Pythonic particle-based (super-droplet) warm-rain/aqueous-chemistry cloud microphysics package with box, parcel & 1D/2D prescribed-flow examples in Python, Julia and Matlab

GPL-3.0 license

51 stars

5 watching

28 forks

Report repository

Releases 86

PySDM v2.56 Latest
4 hours ago

+ 85 releases

Contributors 19

+ 5 contributors

Languages

- Python 96.8%
- Jupyter Notebook 2.0%
- TeX 1.2%

PySDM: open-source particle-based cloud-microphysics package

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

JOSS 10.21105/joss.03219 | DOI 10.5281/zenodo.11186158

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

License GPL v3

tests+artifacts+pypi passing | build passing | codecov 84%

pypi package 2.54 | 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.

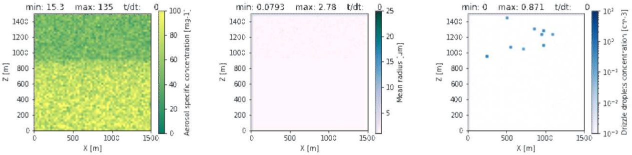
For an overview of PySDM features (and the preferred way to cite PySDM in papers), please refer to our JOSS papers:

- Bartman et al. 2022 (PySDM v1).
- de Jong, Singer et al. 2023 (PySDM v2).

PySDM includes an extension of the SDM scheme to represent collisional breakup described in de Jong, Mackay et al. 2023.
For a list of talks and other materials on PySDM as well as a list of published papers featuring PySDM simulations, see the project wiki.

Example Jupyter notebooks (reproducing results from literature):

See PySDM-examples README.



Pythonic particle-based (super-droplet) warm-rain/aqueous-chemistry cloud microphysics package with box, parcel & 1D/2D prescribed-flow examples in Python, Julia and Matlab

GPL-3.0 license

51 stars

5 watching

28 forks

Report repository

Releases 86

PySDM v2.56 Latest
4 hours ago

+ 85 releases

Contributors 19

+ 5 contributors

Languages

- Python 96.8%
- Jupyter Notebook 2.0%
- TeX 1.2%

PySDM: open-source particle-based cloud-microphysics package

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

JOSS 10.21105/joss.03219 | DOI 10.5281/zenodo.11186158

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

License GPL v3

tests+artifacts+pypi passing | build passing | codecov 84%

pypi package 2.54 | 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.

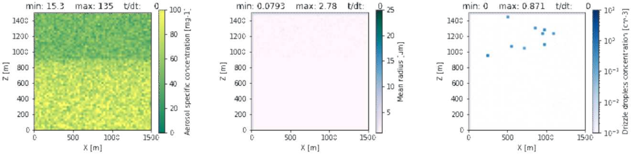
For an overview of PySDM features (and the preferred way to cite PySDM in papers), please refer to our JOSS papers:

- Bartman et al. 2022 (PySDM v1).
- de Jong, Singer et al. 2023 (PySDM v2).

PySDM includes an extension of the SDM scheme to represent collisional breakup described in de Jong, Mackay et al. 2023.
For a list of talks and other materials on PySDM as well as a list of published papers featuring PySDM simulations, see the project wiki.

Example Jupyter notebooks (reproducing results from literature):

See PySDM-examples README.



Pythonic particle-based (super-droplet) warm-rain/aqueous-chemistry cloud microphysics package with box, parcel & 1D/2D prescribed-flow examples in Python, Julia and Matlab

GPL-3.0 license

51 stars

5 watching

28 forks

Report repository

Releases 86

PySDM v2.56 Latest
4 hours ago

+ 85 releases

Contributors 19

+ 5 contributors

Languages

- Python 96.8%
- Jupyter Notebook 2.0%
- TeX 1.2%

PySDM: open-source particle-based cloud-microphysics package

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

JOSS 10.21105/joss.03219 | DOI 10.5281/zenodo.11186158

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

License GPL v3

tests+artifacts+pypi passing | build passing | codecov 84%

pypi package 2.54 | 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.

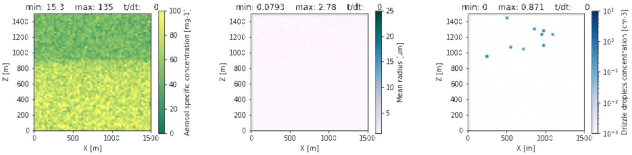
For an overview of PySDM features (and the preferred way to cite PySDM in papers), please refer to our JOSS papers:

- Bartman et al. 2022 (PySDM v1).
- de Jong, Singer et al. 2023 (PySDM v2).

PySDM includes an extension of the SDM scheme to represent collisional breakup described in de Jong, Mackay et al. 2023.
For a list of talks and other materials on PySDM as well as a list of published papers featuring PySDM simulations, see the project wiki.

Example Jupyter notebooks (reproducing results from literature):

See PySDM-examples README.



Pythonic particle-based (super-droplet) warm-rain/aqueous-chemistry cloud microphysics package with box, parcel & 1D/2D prescribed-flow examples in Python, Julia and Matlab

GPL-3.0 license

51 stars

5 watching

28 forks

Report repository

Releases 86

PySDM v2.56 Latest
4 hours ago

+ 85 releases

Contributors 19

+ 5 contributors

Languages

- Python 96.8%
- Jupyter Notebook 2.0%
- TeX 1.2%

PySDM: open-source particle-based cloud-microphysics package

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

JOSS 10.21105/joss.03219 | DOI 10.5281/zenodo.11186158

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

License GPL v3

tests+artifacts+pypi passing | build passing | codecov 84%

pypi package 2.54 | 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.

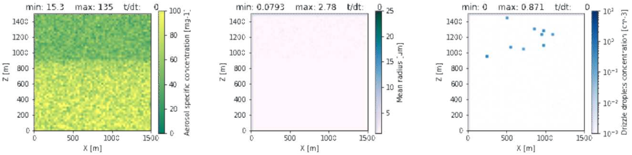
For an overview of PySDM features (and the preferred way to cite PySDM in papers), please refer to our JOSS papers:

- Bartman et al. 2022 (PySDM v1).
- de Jong, Singer et al. 2023 (PySDM v2).

PySDM includes an extension of the SDM scheme to represent collisional breakup described in de Jong, Mackay et al. 2023.
For a list of talks and other materials on PySDM as well as a list of published papers featuring PySDM simulations, see the project wiki.

Example Jupyter notebooks (reproducing results from literature):

See PySDM-examples README.



Pythonic particle-based (super-droplet) warm-rain/aqueous-chemistry cloud microphysics package with box, parcel & 1D/2D prescribed-flow examples in Python, Julia and Matlab

GPL-3.0 license

51 stars

5 watching

28 forks

Report repository

Releases 86

PySDM v2.56 Latest

4 hours ago

+ 85 releases

Contributors 19

+ 5 contributors

Languages

- Python 96.8%
- Jupyter Notebook 2.0%
- TeX 1.2%

PySDM: open-source particle-based cloud-microphysics package

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

JOSS 10.21105/joss.03219 | DOI 10.5281/zenodo.11186158

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

License GPL v3

tests+artifacts+pypi passing | build passing | codecov 84%

pypi package 2.54 | 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.

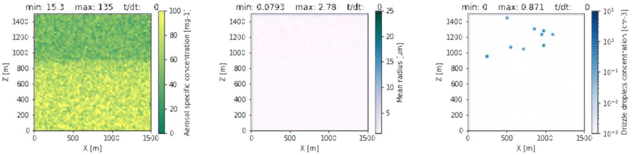
For an overview of PySDM features (and the preferred way to cite PySDM in papers), please refer to our JOSS papers:

- Bartman et al. 2022 (PySDM v1).
- de Jong, Singer et al. 2023 (PySDM v2).

PySDM includes an extension of the SDM scheme to represent collisional breakup described in de Jong, Mackay et al. 2023.
For a list of talks and other materials on PySDM as well as a list of published papers featuring PySDM simulations, see the project wiki.

Example Jupyter notebooks (reproducing results from literature):

See PySDM-examples README.



Pythonic particle-based (super-droplet) warm-rain/aqueous-chemistry cloud microphysics package with box, parcel & 1D/2D prescribed-flow examples in Python, Julia and Matlab

GPL-3.0 license

51 stars

5 watching

28 forks

Report repository

Releases 86

PySDM v2.56 Latest
4 hours ago

+ 85 releases

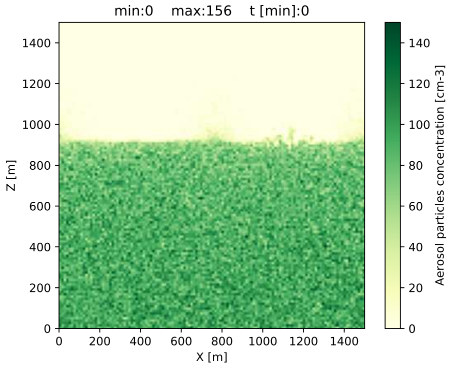
Contributors 19

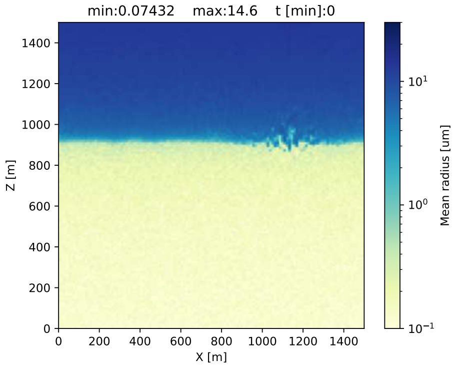
+ 5 contributors

Languages

- **Python** 96.8%
- **Jupyter Notebook** 2.0%
- **TeX** 1.2%

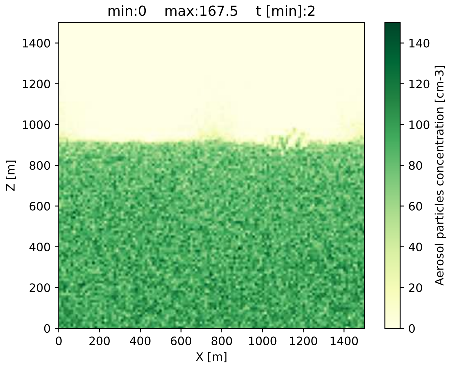
Eulerian solver (kinematic flow) grid: 128×128
Lagrangian super-particles population: 2^{21}

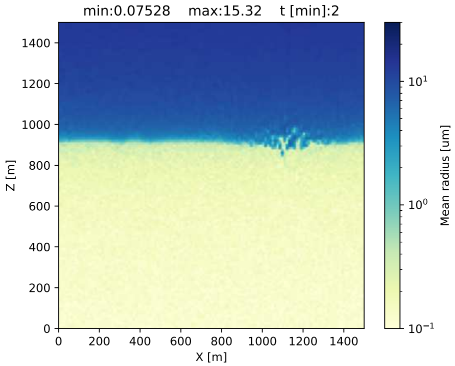




simulation & vis.: Piotr Bartman

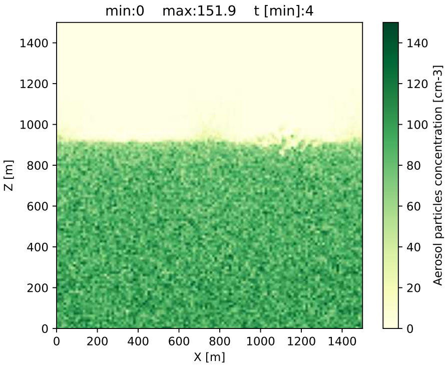
Eulerian solver (kinematic flow) grid: 128×128
Lagrangian super-particles population: 2^{21}

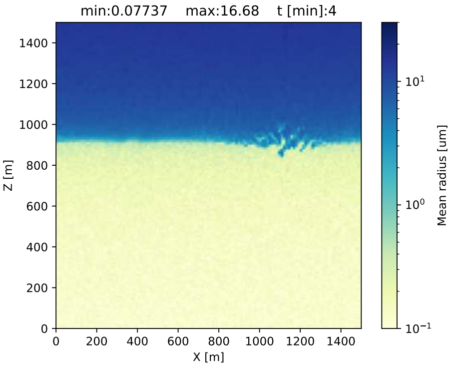




simulation & vis.: Piotr Bartman

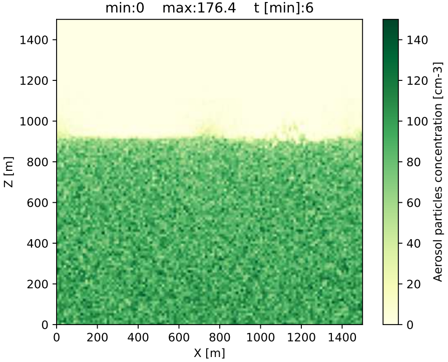
Eulerian solver (kinematic flow) grid: 128×128
Lagrangian super-particles population: 2^{21}

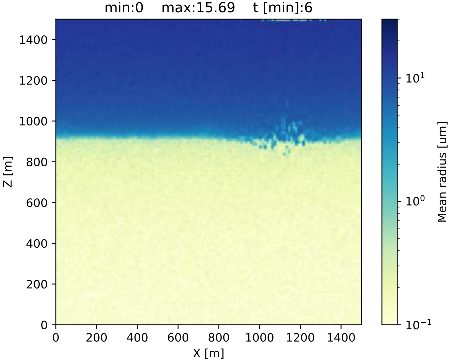




simulation & vis.: Piotr Bartman

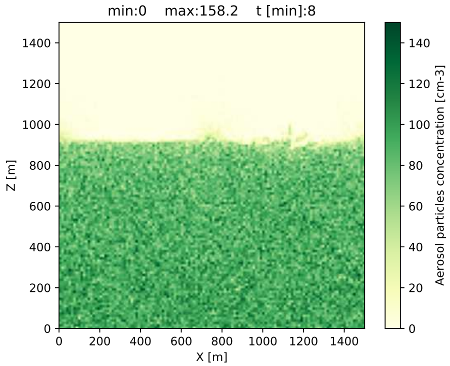
Eulerian solver (kinematic flow) grid: 128×128
Lagrangian super-particles population: 2^{21}

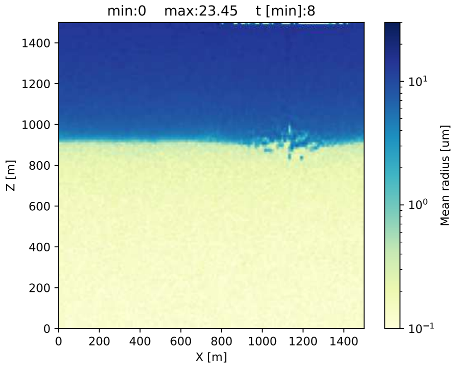




simulation & vis.: Piotr Bartman

Eulerian solver (kinematic flow) grid: 128×128
Lagrangian super-particles population: 2^{21}

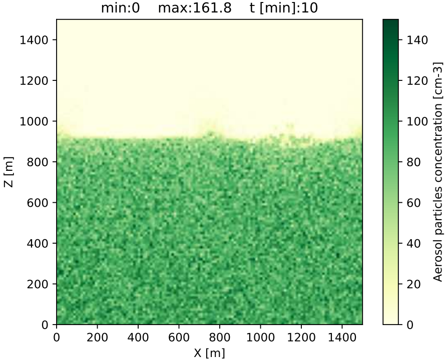


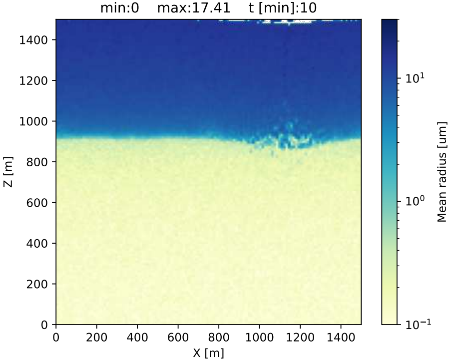


simulation & vis.: Piotr Bartman

Eulerian solver (kinematic flow) grid: 128×128

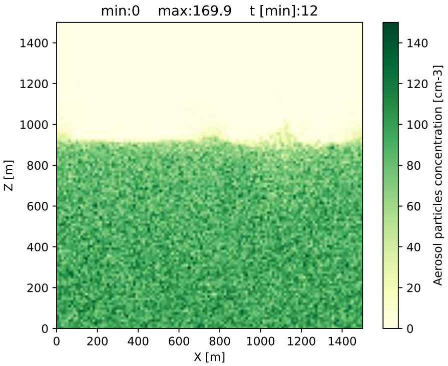
Lagrangian super-particles population: 2^{21}

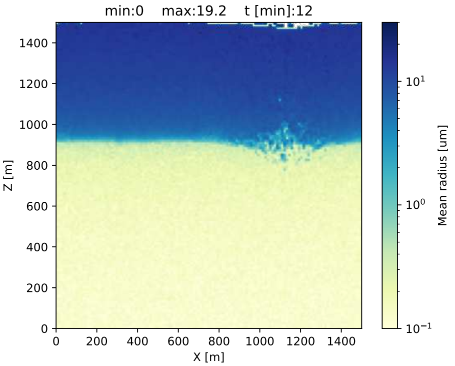




simulation & vis.: Piotr Bartman

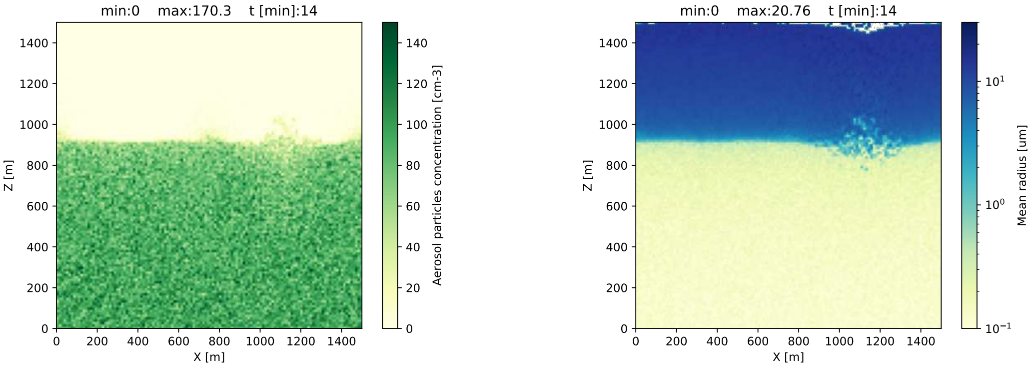
Eulerian solver (kinematic flow) grid: 128×128
Lagrangian super-particles population: 2^{21}





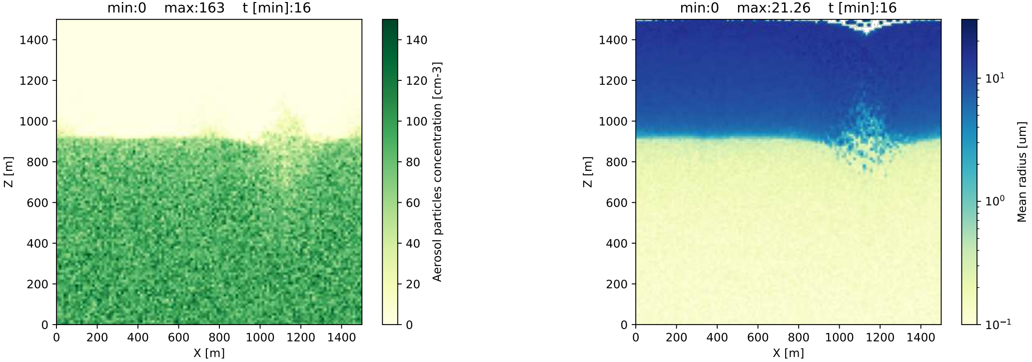
simulation & vis.: Piotr Bartman

Eulerian solver (kinematic flow) grid: 128×128
Lagrangian super-particles population: 2^{21}



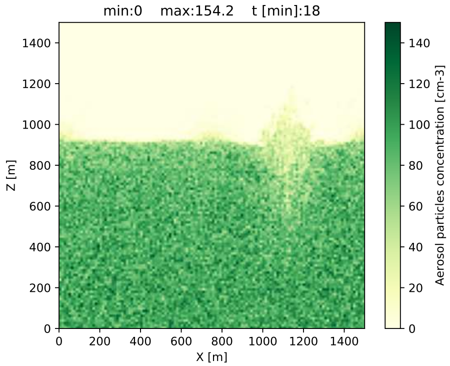
simulation & vis.: Piotr Bartman

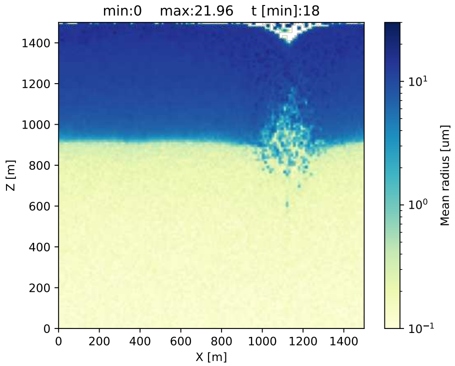
Eulerian solver (kinematic flow) grid: 128×128
Lagrangian super-particles population: 2^{21}



simulation & vis.: Piotr Bartman

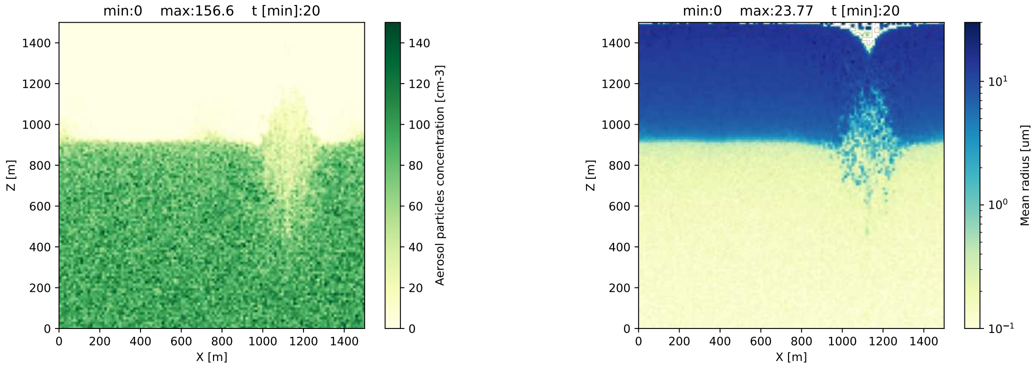
Eulerian solver (kinematic flow) grid: 128×128
Lagrangian super-particles population: 2^{21}





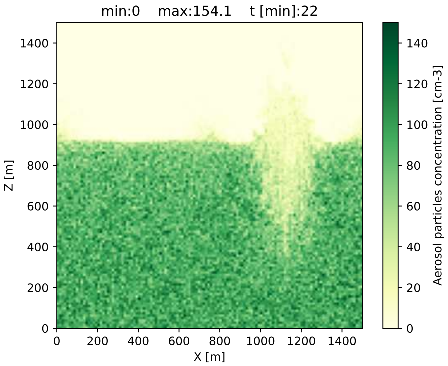
simulation & vis.: Piotr Bartman

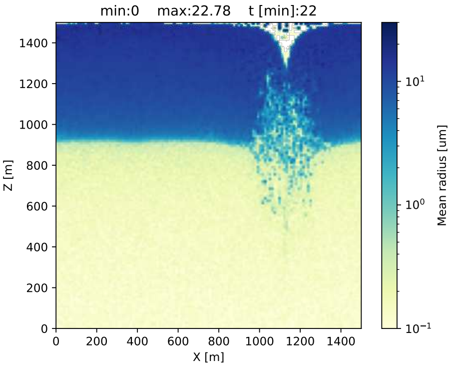
Eulerian solver (kinematic flow) grid: 128×128
Lagrangian super-particles population: 2^{21}



simulation & vis.: Piotr Bartman

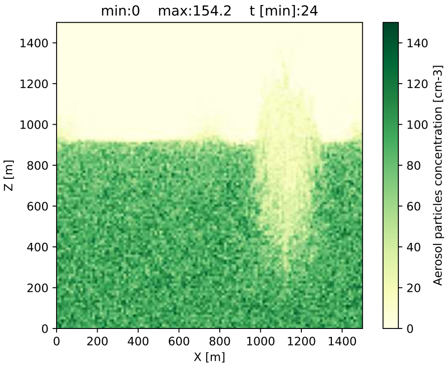
Eulerian solver (kinematic flow) grid: 128×128
Lagrangian super-particles population: 2^{21}

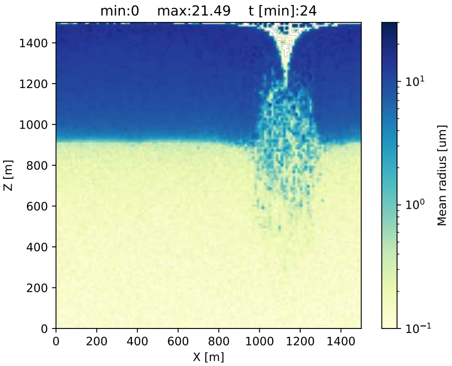




simulation & vis.: Piotr Bartman

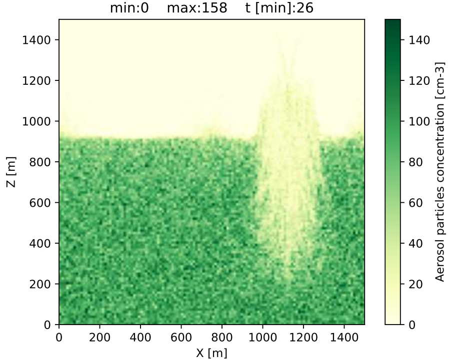
Eulerian solver (kinematic flow) grid: 128×128
Lagrangian super-particles population: 2^{21}

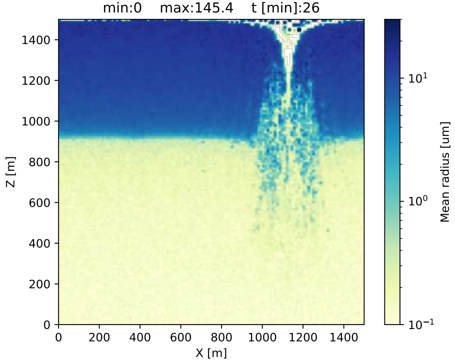




simulation & vis.: Piotr Bartman

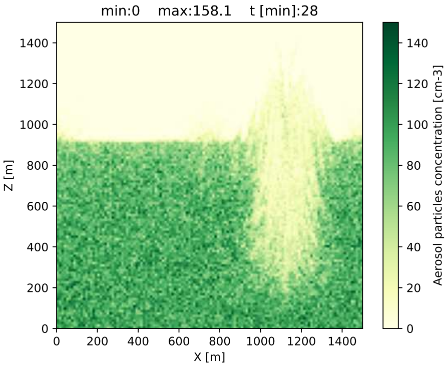
Eulerian solver (kinematic flow) grid: 128×128
Lagrangian super-particles population: 2^{21}

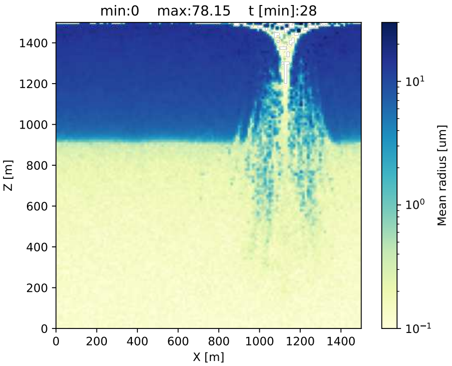




simulation & vis.: Piotr Bartman

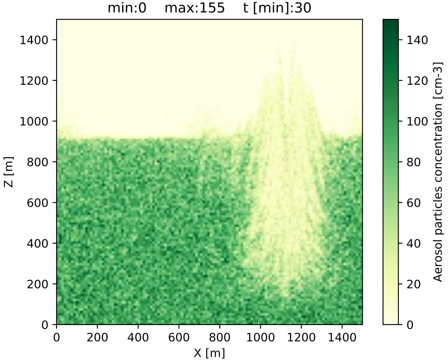
Eulerian solver (kinematic flow) grid: 128×128
Lagrangian super-particles population: 2^{21}

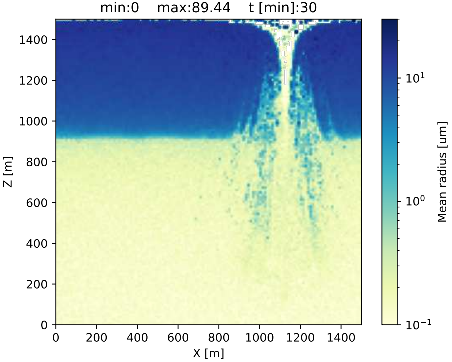




simulation & vis.: Piotr Bartman

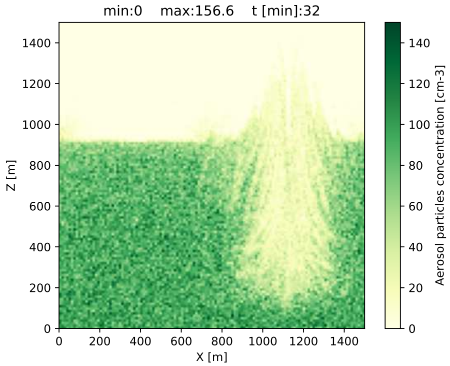
Eulerian solver (kinematic flow) grid: 128×128
Lagrangian super-particles population: 2^{21}

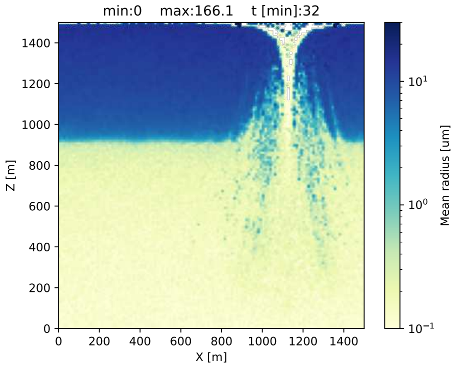




simulation & vis.: Piotr Bartman

Eulerian solver (kinematic flow) grid: 128×128
Lagrangian super-particles population: 2^{21}

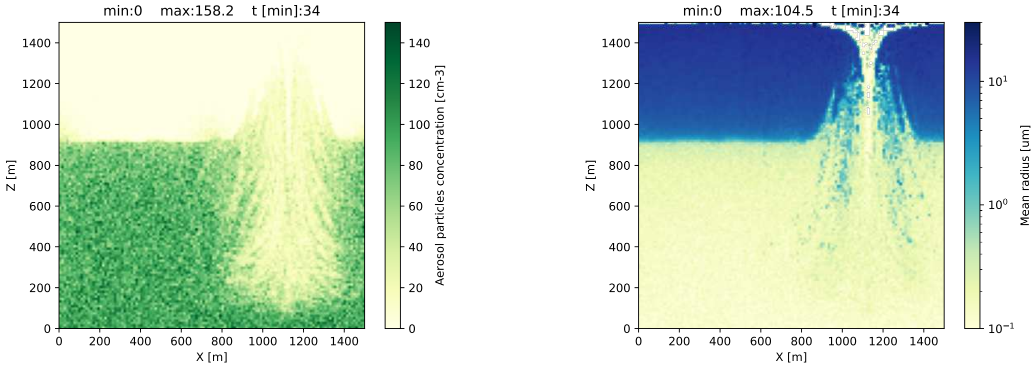




simulation & vis.: Piotr Bartman

Eulerian solver (kinematic flow) grid: 128×128

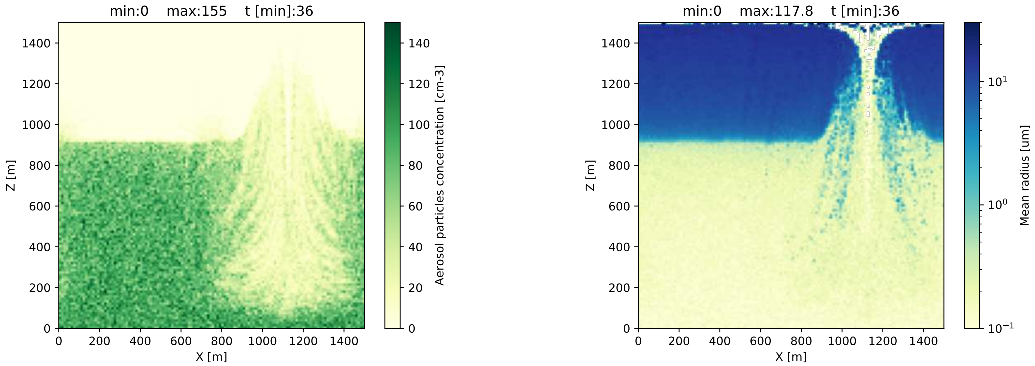
Lagrangian super-particles population: 2^{21}



simulation & vis.: Piotr Bartman

Eulerian solver (kinematic flow) grid: 128×128

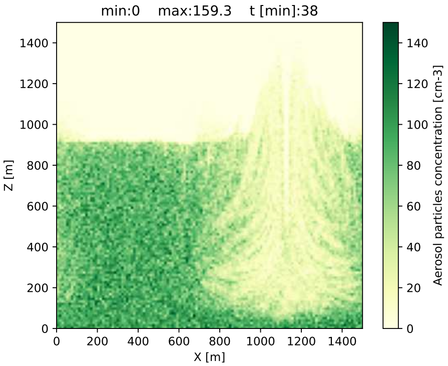
Lagrangian super-particles population: 2^{21}

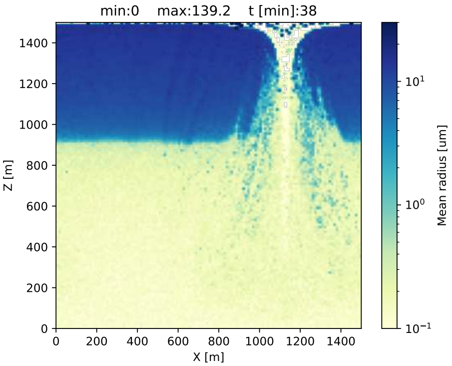


simulation & vis.: Piotr Bartman

Eulerian solver (kinematic flow) grid: 128×128

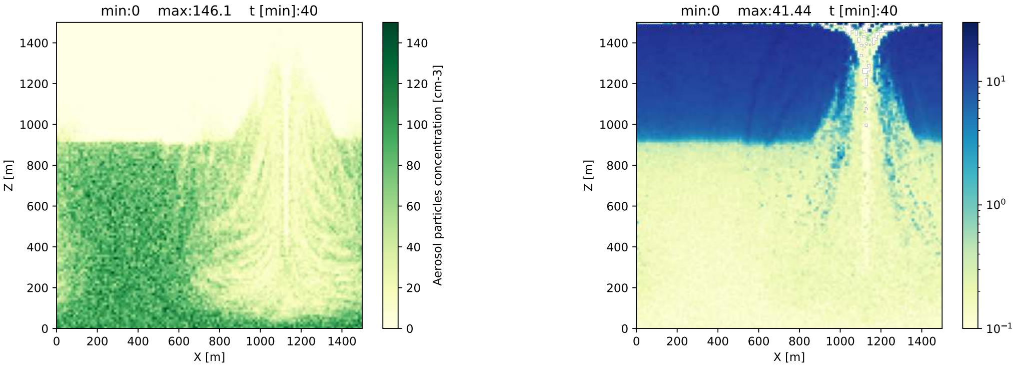
Lagrangian super-particles population: 2^{21}



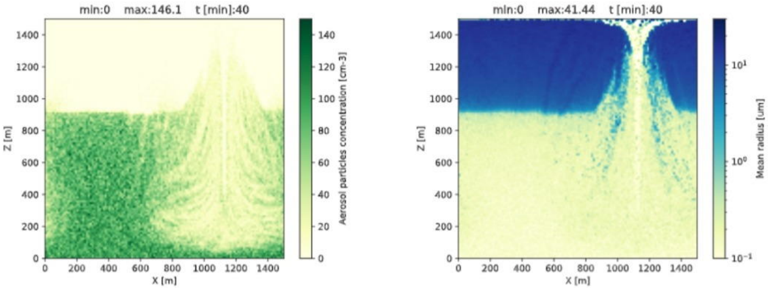


simulation & vis.: Piotr Bartman

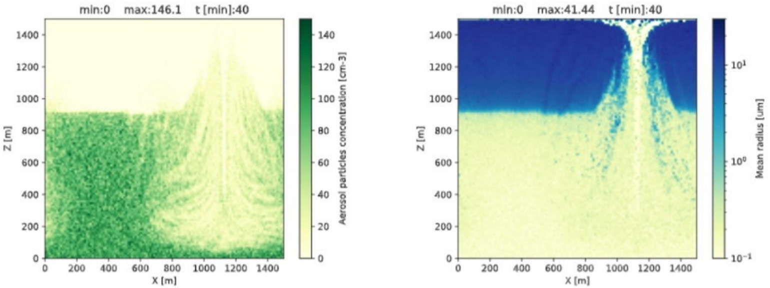
Eulerian solver (kinematic flow) grid: 128×128
Lagrangian super-particles population: 2^{21}

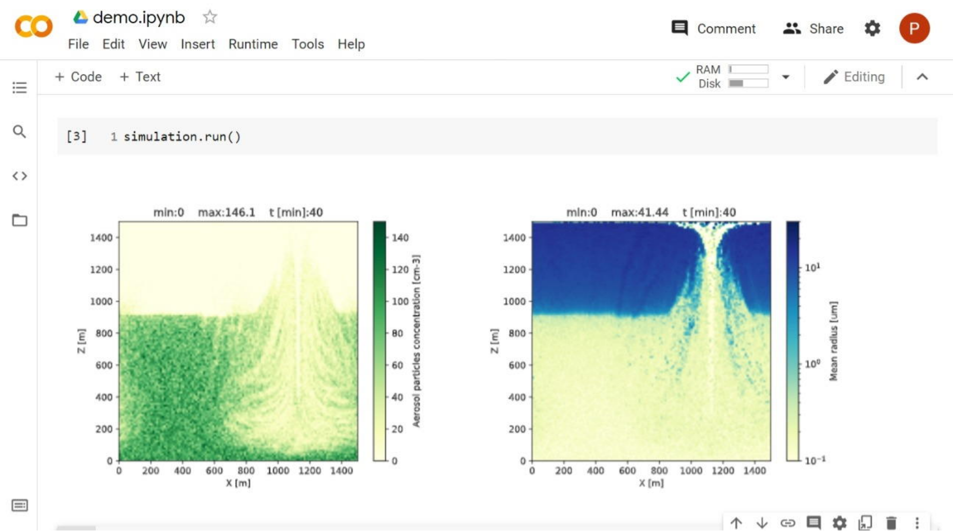


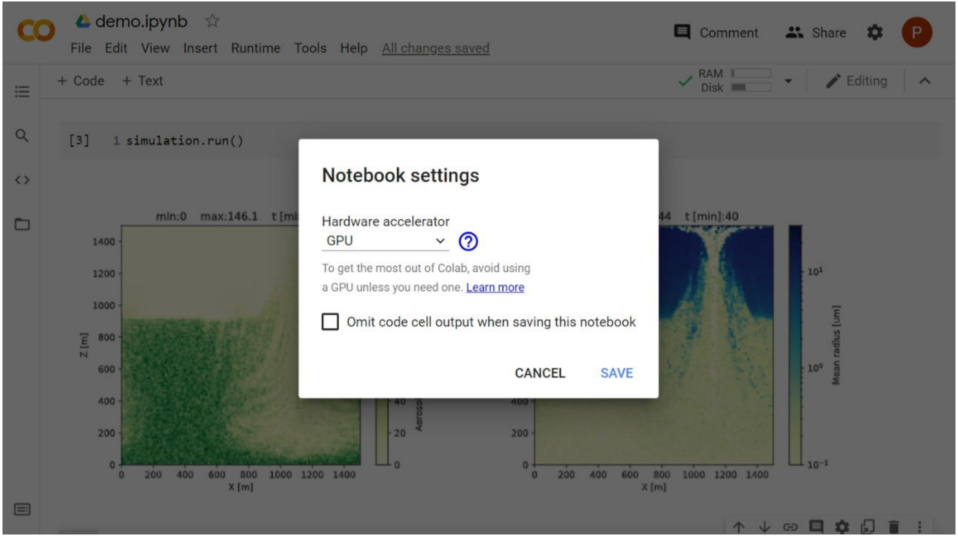
simulation & vis.: Piotr Bartman


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]

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




demo.ipynb
File Edit View Insert Runtime Tools Help
Comment
Share
+ Code + Text
RAM
Disk
Editing
[3] 1 simulation.run()
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
Mean radius [um]


demo.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing
[3] 1 simulation.run()
Notebook settings
Hardware accelerator
GPU
To get the most out of Colab, avoid using
a GPU unless you need one. Learn more
Omit code cell output when saving this notebook
CANCEL
SAVE
min:0 max:146.1
t [min]:40
Z [m]
X [m]
Mean radius [um]

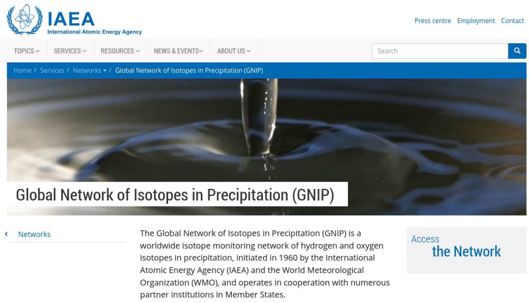
PySDM: Jupyer notebooks reproducing results from literature

pypi.org/p/PySDM-examples

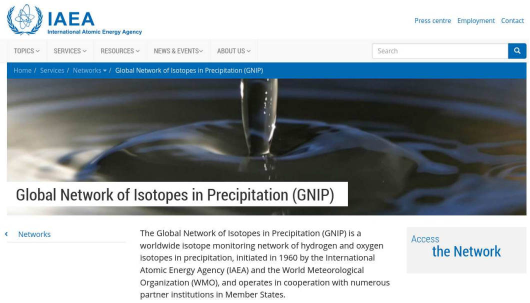
literature reference	cond/evap	coalescence	isotopes	breakup	transport	chemistry	freezing	keywords
formulae-only								
Gedzelman & Arnold 1994			×					#dynsys
Pierchala et al. 2022			×					#lab-experiment
...								
0D box environment								
Berry 1967		×						#kernels
Shima et al. 2009		×						#analytic-solution
Alpert & Knopf 2016							×	#ABIFM
de Jong et al. 2023		×		×				#analytic-solution
...								
0D parcel environment								
Rozanski & Sonntag 1982	×		×					#iterative-parcel
Abdul-Razzak & Ghan 2000	×							#pysdm-vs-gcm-param
Kreidenweis et al. 2003	×					×		#Hoppel-gap
Arabas and Shima 2017	×							#dynsys
Jensen and Nugent 2017	×							#giant-CCN
Yang et al. 2018	×							#ripening
Lowe et al. 2019	×							#surfactants
Grabowski and Pawlowska 2023	×							#ripening
...								
1D single-column kinematic env. (advection: PyMPDATA)								
Shipway & Hill 2012	×	×			×			#KiD
deJong et al. 2023 (figures 6-8)	×	×		×	×			#KiD
...								
2D prescribed-flow environment (advection: PyMPDATA)								
Arabas et al. 2015	×	×			×			#GUI
Arabas et al. 2023 (figure 11)	×	×			×		×	#Paraview

AGH

ENVIRONMENTAL
PHYSICS
GROUP



- IAEA/GNIP site in Kraków



- IAEA/GNIP site in Kraków
- 50-year precip isotopic data record

AGH

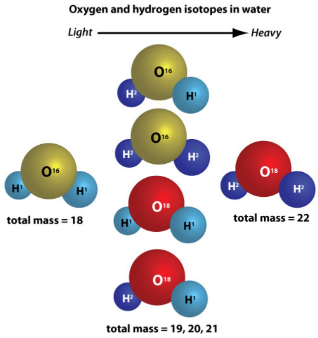
ENVIRONMENTAL PHYSICS GROUP

- IAEA/GNIP site in Kraków
- 50-year precip isotopic data record
- high-altitude lab (clouds in-situ) @Kasprowy Wierch (1987 m AMSL)

photo: naukaoklimacie.pl

clouds from a water isotopic point of view

- water isotopologues (stable): H_2O (99.7%), $H_2{}^{18}O$ (0.2%), HDO (0.03%), $H_2{}^{17}O$ (0.04%), ...

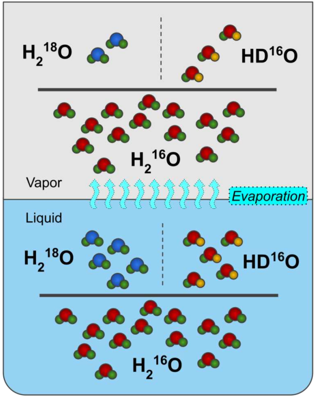


graphic: usgs.gov

$$M_v \approx 18.015 \text{ g/mol}$$

clouds from a water isotopic point of view

- water isotopologues (stable): H_2O (99.7%), $H_2{}^{18}O$ (0.2%), HDO (0.03%), $H_2{}^{17}O$ (0.04%), ...
- condensation "favors" heavy over light isotopologues (evaporation vice versa)
  $\rightsquigarrow$ equilibrium fractionation
  $\rightsquigarrow$ more pronounced in colder temperatures
  $\rightsquigarrow$ larger ($\times 8$) effect for H than O



graphic: scisnack.com

$$\alpha_{eq}^{HDO}(20^\circ C) = e_s^{light} / e_s^{heavy} \approx 1.08$$

$$\alpha_{eq}^{H_2^{18}O}(20^\circ C) \approx 1.01$$

clouds from a water isotopic point of view

- water isotopologues (stable): H_2O (99.7%), $H_2{}^{18}O$ (0.2%), HDO (0.03%), $H_2{}^{17}O$ (0.04%), ...
- condensation "favors" heavy over light isotopologues (evaporation vice versa)
  $\rightsquigarrow$ equilibrium fractionation
  $\rightsquigarrow$ more pronounced in colder temperatures
  $\rightsquigarrow$ larger ($\times 8$) effect for H than O
- differences in diffusivity in air
  $\rightsquigarrow$ non-equilibrium (kinetic) fractionation
  $\rightsquigarrow$ applies to sub- and super-saturated conditions
  $\rightsquigarrow$ more pronounced for O than H

$$\frac{\alpha_{\text{eff}}}{\alpha_{\text{eq}}} - 1 \approx n \cdot \left(1 - \frac{D^{\text{heavy}}}{D^{\text{light}}}\right) \cdot (1 - \text{RH})$$

α_{eff} effective fractionation coeff.

n turbulence parameter

D diffusion coeffs:

HDO: $(1 - D^{\text{heavy}} / D^{\text{light}})\Big|_{T=20^\circ\text{C}} \approx 2.5\%$

$H_2^{18}O$: $(1 - D^{\text{heavy}} / D^{\text{light}})\Big|_{T=20^\circ\text{C}} \approx 2.9\%$

RH rel. humidity

precipitating cloud as an isotopic distillation column

(photo: Yevgen Timashov / National Geographic; Ai-Petri, Crimea, Ukraine)

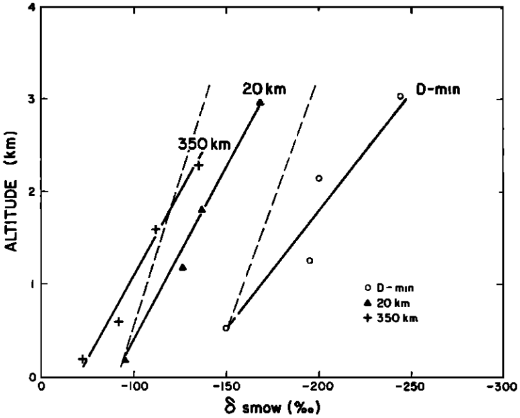
clouds from a water isotopic point of view

pypi.org/p/PySDM-examples

literature reference	cond/evap	coalescence	isotopes	breakup	transport	chemistry	freezing	keywords
formulae-only								
Gedzelman & Arnold 1994			×					#dynsys
Pierchala et al. 2022			×					#lab-experiment
...								
0D box environment								
Berry 1967		×						#kernels
Shima et al. 2009		×						#analytic-solution
Alpert & Knopf 2016							×	#ABIFM
de Jong et al. 2023		×		×				#analytic-solution
...								
0D parcel environment								
Rozanski & Sonntag 1982	×		×					#iterative-parcel
Abdul-Razzak & Ghan 2000	×							#pysdm-vs-gcm-param
Kreidenweis et al. 2003	×					×		#Hoppel-gap
Arabas and Shima 2017	×							#dynsys
Jensen and Nugent 2017	×							#giant-CCN
Yang et al. 2018	×							#ripening
Lowe et al. 2019	×							#surfactants
Grabowski and Pawlowska 2023	×							#ripening
...								
1D single-column kinematic env. (advection: PyMPDATA)								
Shipway & Hill 2012	×	×			×			#KiD
deJong et al. 2023 (figures 6-8)	×	×		×	×			#KiD
...								
2D prescribed-flow environment (advection: PyMPDATA)								
Arabas et al. 2015	×	×			×			#GUI
Arabas et al. 2023 (figure 11)	×	×			×		×	#Paraview

Rozanski & Sonntag ’82 – iterative parcel PySDM setup

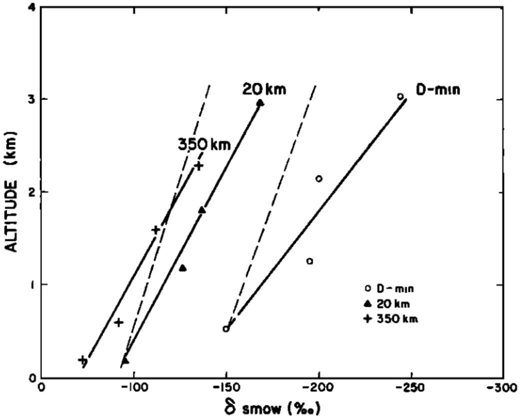
Ehhalt & Östlund 1970



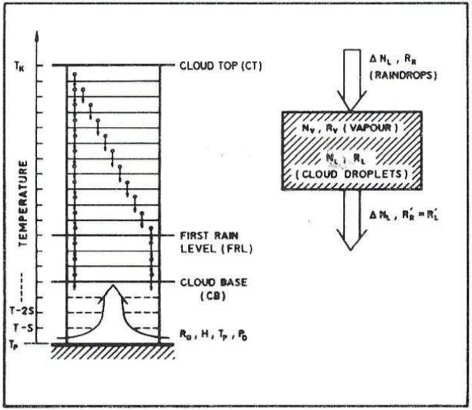
$$\delta_{\text{SMOW}} = \frac{[heavy]/[light]|_{\text{sample}}}{[heavy]/[light]|_{\text{standard}}} - 1$$

Rozanski & Sonntag '82 – iterative parcel PySDM setup

Ehhalt & Östlund 1970



Rozanski & Sonntag 1982



Rozanski & Sonntag '82 – iterative parcel PySDM setup[a]

[a]launch-in-the-cloud URL: `https://mybinder.org/v2/gh/slayoo/PySDM.git/isotopes_rozanski_and_sonntag_example?urlpath=lab/tree/examples/PySDM_examples`

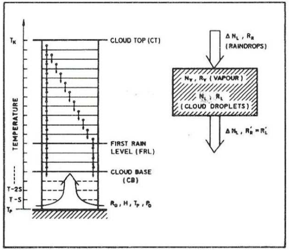


Fig. 3. Schematic diagram of the multibox cloud model. Input data: initial temperature, T_p; final temperature, T_k; initial pressure, P_0; relative humidity, H; initial isotopic composition of water vapour, R_0; cloud water mixing ratio, N_L; temperature step, S; isotope exchange factor, K.

Tellus 34 (1982), 2

Rozanski & Sonntag ’82 – iterative parcel PySDM setup[a]

[a] launch-in-the-cloud URL: https://mybinder.org/v2/gh/slayoo/PySDM.git/isotopes_rozanski_and_sonntag_example?urlpath=lab/tree/examples/PySDM_examples

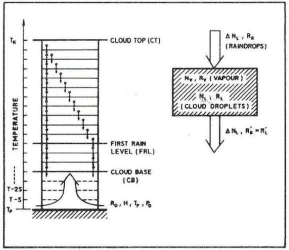


Fig. 3. Schematic diagram of the multibox cloud model. Input data: initial temperature, T_p; final temperature, T_k; initial pressure, P_0; relative humidity, H; initial isotopic composition of water vapour, R_0; cloud water mixing ratio, N_L; temperature step, S; isotope exchange factor, K.

Tellus 34 (1982), 2

- hydrostatic/adiabatic rainshaft with precip removal

Rozanski & Sonntag '82 – iterative parcel PySDM setup[a]

[a] launch-in-the-cloud URL: `https://mybinder.org/v2/gh/slayoo/PySDM.git/isotopes_rozanski_and_sonntag_example?urlpath=lab/tree/examples/PySDM_examples`

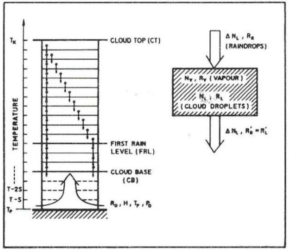


Fig. 3. Schematic diagram of the multibox cloud model. Input data: initial temperature, T_p; final temperature, T_k; initial pressure, P_0; relative humidity, H; initial isotopic composition of water vapour, R_0; cloud water mixing ratio, N_L; temperature step, S; isotope exchange factor, K.

Tellus 34 (1982), 2

- hydrostatic/adiabatic rainshaft with precip removal
- condensation: saturation adjustment

Rozanski & Sonntag '82 – iterative parcel PySDM setup[a]

[a]launch-in-the-cloud URL: `https://mybinder.org/v2/gh/slayoo/PySDM.git/isotopes_rozanski_and_sonntag_example?urlpath=lab/tree/examples/PySDM_examples`

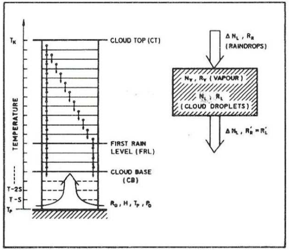


Fig. 3. Schematic diagram of the multibox cloud model. Input data: initial temperature, T_p; final temperature, T_k; initial pressure, P_0; relative humidity, H; initial isotopic composition of water vapour, R_0; cloud water mixing ratio, N_L; temperature step, S; isotope exchange factor, K.

Tellus 34 (1982), 2

- hydrostatic/adiabatic rainshaft with precip removal
- condensation: saturation adjustment
- rain formation: liquid water content threshold

Rozanski & Sonntag '82 – iterative parcel PySDM setup[a]

[a] launch-in-the-cloud URL: `https://mybinder.org/v2/gh/slayoo/PySDM.git/isotopes_rozanski_and_sonntag_example?urlpath=lab/tree/examples/PySDM_examples`

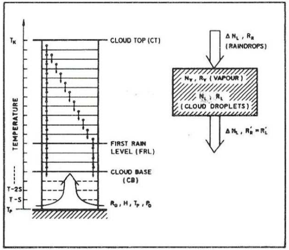


Fig. 3. Schematic diagram of the multibox cloud model. Input data: initial temperature, T_p; final temperature, T_k; initial pressure, P_0; relative humidity, H; initial isotopic composition of water vapour, R_0; cloud water mixing ratio, N_L; temperature step, S; isotope exchange factor, K.

Tellus 34 (1982), 2

- hydrostatic/adiabatic rainshaft with precip removal
- condensation: saturation adjustment
- rain formation: liquid water content threshold
- parcel-model iterations towards stationary state (no explicit role of time)

Rozanski & Sonntag '82 – iterative parcel PySDM setup[a]

[a] launch-in-the-cloud URL: `https://mybinder.org/v2/gh/slayoo/PySDM.git/isotopes_rozanski_and_sonntag_example?urlpath=lab/tree/examples/PySDM_examples`

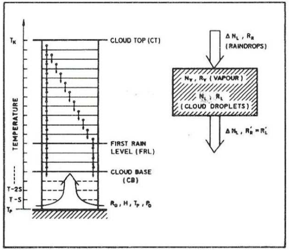


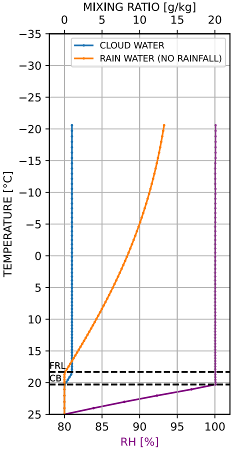
Fig. 3. Schematic diagram of the multibox cloud model. Input data: initial temperature, T_p; final temperature, T_k; initial pressure, P_0; relative humidity, H; initial isotopic composition of water vapour, R_0; cloud water mixing ratio, N_L; temperature step, S; isotope exchange factor, K.

Tellus 34 (1982), 2

- hydrostatic/adiabatic rainshaft with precip removal
- condensation: saturation adjustment
- rain formation: liquid water content threshold
- parcel-model iterations towards stationary state (no explicit role of time)
- minimal model for capturing isotope exchange between precip, ambient vapor and cloud water ($\rightsquigarrow$ hypothesis explaining observed steep δ^2H profile gradients beyond condensation-only effects)

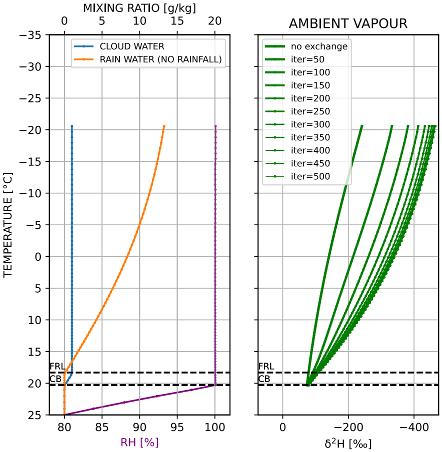
Rozanski & Sonntag '82 – iterative parcel PySDM setup[a]

[a] launch-in-the-cloud URL: `https://mybinder.org/v2/gh/slayoo/PySDM.git/isotopes_rozanski_and_sonntag_example?urlpath=lab/tree/examples/PySDM_examples`



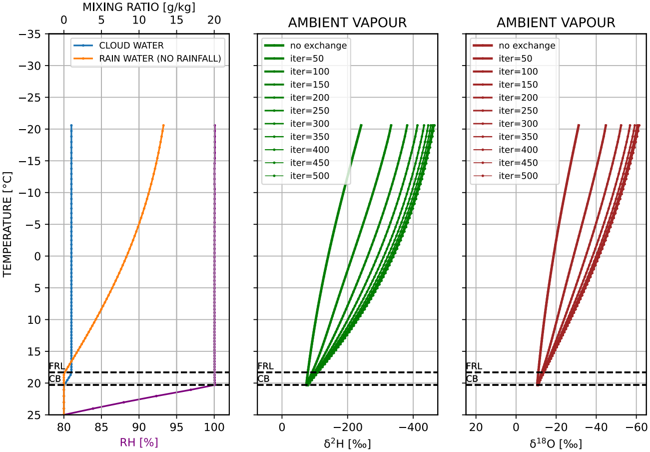
Rozanski & Sonntag '82 – iterative parcel PySDM setup[a]

[a] launch-in-the-cloud URL: `https://mybinder.org/v2/gh/slayoo/PySDM.git/isotopes_rozanski_and_sonntag_example?urlpath=lab/tree/examples/PySDM_examples`



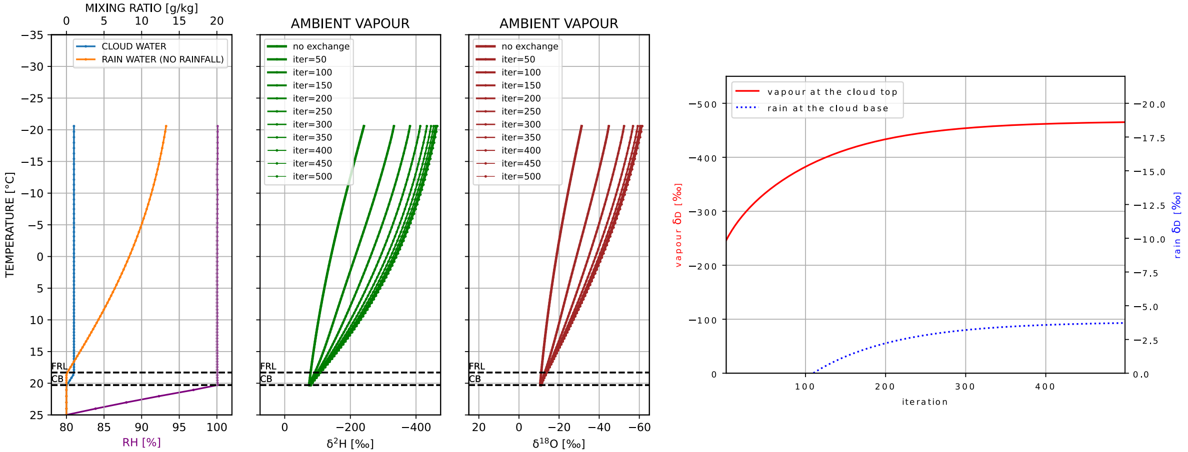
Rozanski & Sonntag ’82 – iterative parcel PySDM setup[a]

[a] launch-in-the-cloud URL: `https://mybinder.org/v2/gh/slayoo/PySDM.git/isotopes_rozanski_and_sonntag_example?urlpath=lab/tree/examples/PySDM_examples`



Rozanski & Sonntag '82 – iterative parcel PySDM setup[a]

[a]launch-in-the-cloud URL: `https://mybinder.org/v2/gh/slayoo/PySDM.git/isotopes_rozanski_and_sonntag_example?urlpath=lab/tree/examples/PySDM_examples`



Rozanski & Sonntag '82 – iterative parcel PySDM setup

pypi.org/p/PySDM-examples

literature reference	cond/evap	coalescence	isotopes	breakup	transport	chemistry	freezing	keywords
formulae-only								
Gedzelman & Arnold 1994			×					#dynsys
Pierchala et al. 2022			×					#lab-experiment
...								
0D box environment								
Berry 1967		×						#kernels
Shima et al. 2009		×						#analytic-solution
Alpert & Knopf 2016							×	#ABIFM
de Jong et al. 2023		×		×				#analytic-solution
...								
0D parcel environment								
Rozanski & Sonntag 1982	×		×					#iterative-parcel
Abdul-Razzak & Ghan 2000	×							#pysdm-vs-gcm-param
Kreidenweis et al. 2003	×					×		#Hoppel-gap
Arabas and Shima 2017	×							#dynsys
Jensen and Nugent 2017	×							#giant-CCN
Yang et al. 2018	×							#ripening
Lowe et al. 2019	×							#surfactants
Grabowski and Pawlowska 2023	×							#ripening
...								
1D single-column kinematic env. (advection: PyMPDATA)								
Shipway & Hill 2012	×	×			×			#KiD
deJong et al. 2023 (figures 6-8)	×	×		×	×			#KiD
...								
2D prescribed-flow environment (advection: PyMPDATA)								
Arabas et al. 2015	×	×			×			#GUI
Arabas et al. 2023 (figure 11)	×	×			×		×	#Paraview

Pierchala et al. '22 – triple isotope / kinetic fract. PySDM setup[a]

[a] launch-in-the-cloud URL: `https://mybinder.org/v2/gh/open-atmos/PySDM.git/main?urlpath=lab/tree/examples/PySDM_examples/Pierchala_et_al_2022`

doi:10.1016/j.gca.2022.01.020

Geochimica et Cosmochimica Acta 322 (2022) 244–259

www.elsevier.com/locate/gca

Quantification the diffusion-induced fractionation of $^{1}H_2^{17}O$ isotopologue in air accompanying the process of water evaporation

Anna Pierchala *, Kazimierz Rozanski, Marek Dulinski, Zbigniew Gorczyca

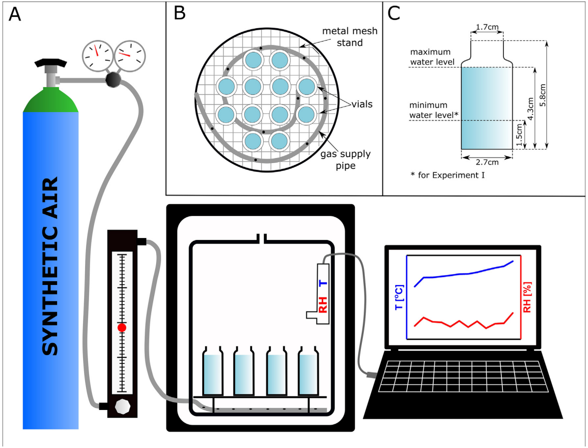
AGH University of Science and Technology, Faculty of Physics and Applied Computer Science, al. Mickiewicza 30, 30-059 Krakow, Poland

Received 25 February 2021; accepted in revised form 15 January 2022; Available online 24 January 2022

Pierchala et al. '22 – triple isotope / kinetic fract. PySDM setup[a]

[a] launch-in-the-cloud URL: `https://mybinder.org/v2/gh/open-atmos/PySDM.git/main?urlpath=lab/tree/examples/PySDM_examples/Pierchala_et_al_2022`

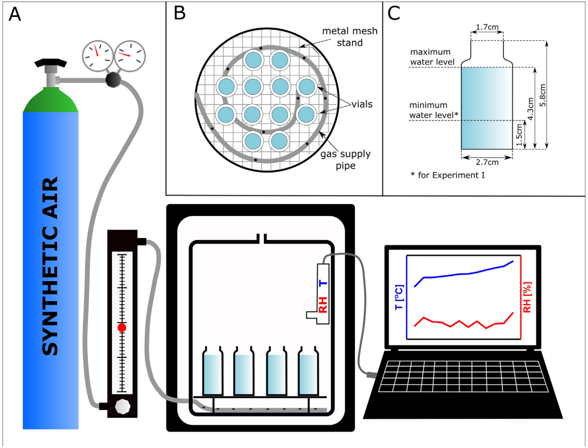
Fig. 1 (paper): lab experiment setup



Pierchala et al. ’22 – triple isotope / kinetic fract. PySDM setup[a]

[a] launch-in-the-cloud URL: `https://mybinder.org/v2/gh/open-atmos/PySDM.git/main?urlpath=lab/tree/examples/PySDM_examples/Pierchala_et_al_2022`

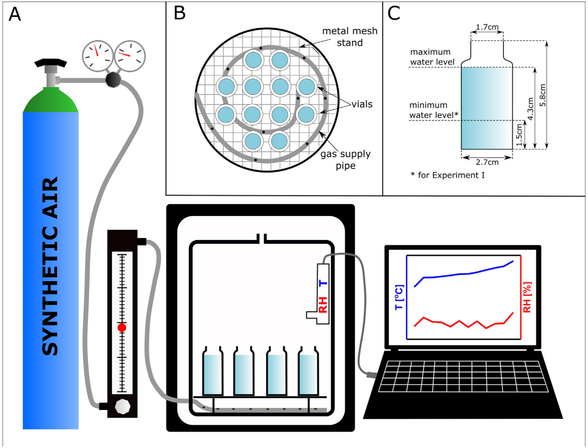
Fig. 1 (paper): lab experiment setup



- fractionation upon evaporation (incl. kinetic effects)

[a] launch-in-the-cloud URL: `https://mybinder.org/v2/gh/open-atmos/PySDM.git/main?urlpath=lab/tree/examples/PySDM_examples/Pierchala_et_al_2022`

Fig. 1 (paper): lab experiment setup

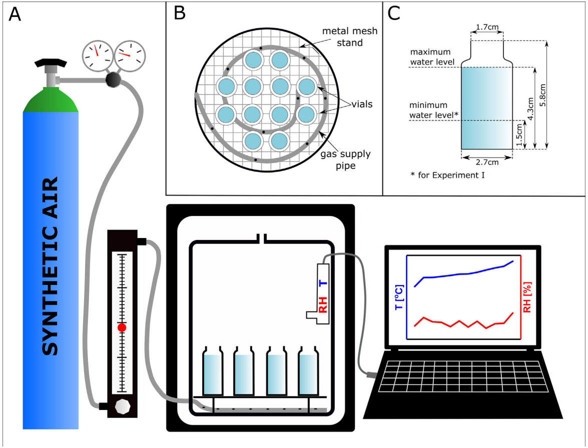


- fractionation upon evaporation (incl. kinetic effects)
- multi-day experiments (up to two weeks)

Pierchala et al. '22 – triple isotope / kinetic fract. PySDM setup[a]

[a] launch-in-the-cloud URL: `https://mybinder.org/v2/gh/open-atmos/PySDM.git/main?urlpath=lab/tree/examples/PySDM_examples/Pierchala_et_al_2022`

Fig. 1 (paper): lab experiment setup

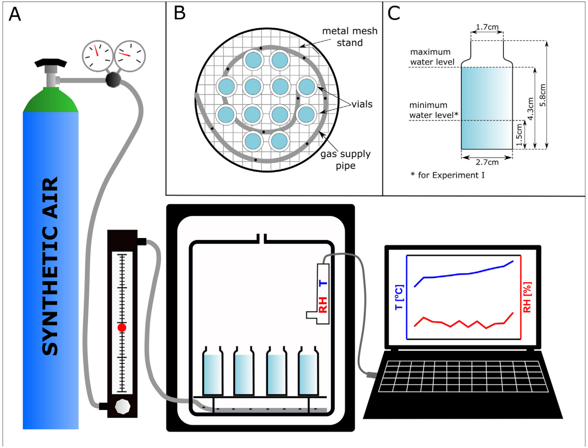


- fractionation upon evaporation (incl. kinetic effects)
- multi-day experiments (up to two weeks)
- Picarro L2140-i cavity ring-down laser spectrometer @AGH (D, ^{18}O, ^{17}O) (probing vaporized water)

Pierchala et al. '22 – triple isotope / kinetic fract. PySDM setup[a]

[a] launch-in-the-cloud URL: `https://mybinder.org/v2/gh/open-atmos/PySDM.git/main?urlpath=lab/tree/examples/PySDM_examples/Pierchala_et_al_2022`

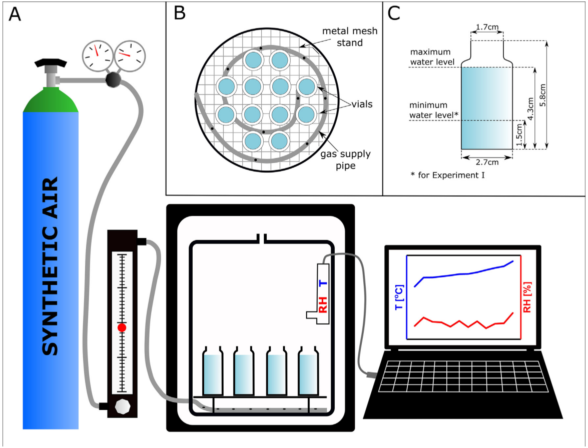
Fig. 1 (paper): lab experiment setup



- fractionation upon evaporation (incl. kinetic effects)
- multi-day experiments (up to two weeks)
- Picarro L2140-i cavity ring-down laser spectrometer @AGH (D, ^{18}O, ^{17}O) (probing vaporized water)
- constant T/RH, variable T or variable RH setups

[a] launch-in-the-cloud URL: `https://mybinder.org/v2/gh/open-atmos/PySDM.git/main?urlpath=lab/tree/examples/PySDM_examples/Pierchala_et_al_2022`

Fig. 1 (paper): lab experiment setup

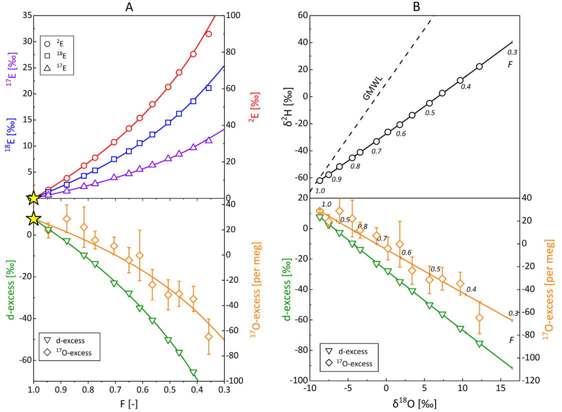


- fractionation upon evaporation (incl. kinetic effects)
- multi-day experiments (up to two weeks)
- Picarro L2140-i cavity ring-down laser spectrometer @AGH (D, ^{18}O, ^{17}O) (probing vaporized water)
- constant T/RH, variable T or variable RH setups

Pierchala et al. '22 – triple isotope / kinetic fract. PySDM setup[a]

[a] launch-in-the-cloud URL: `https://mybinder.org/v2/gh/open-atmos/PySDM.git/main?urlpath=lab/tree/examples/PySDM_examples/Pierchala_et_al_2022`

Fig. 3 (paper): measurements + model



$$E = \frac{[\text{heavy iso.}]}{[\text{light iso.}]} \Big/ \frac{[\text{heavy iso.}]}{[\text{light iso.}]}\Bigg|_{t=0} - 1 \qquad \delta = \frac{[\text{heavy iso.}]}{[\text{light iso.}]} \Big/ \frac{[\text{heavy iso.}]}{[\text{light iso.}]}\Bigg|_{\text{VSMOW}} - 1$$

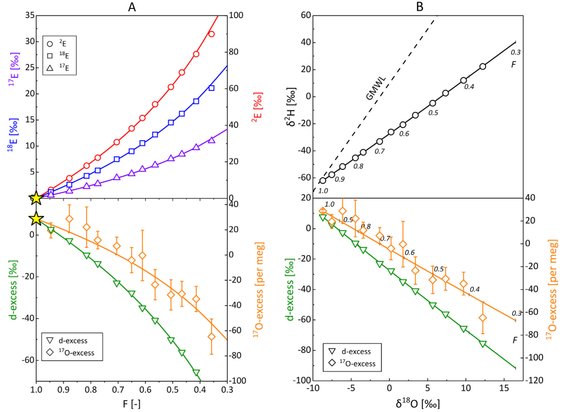
F: fraction of water remaining

d-excess: $\delta^2 H - 8 \cdot \delta^{18} O$ $\qquad$ ^{17}O-excess: $\ln(\delta^{17} O + 1) - 0.528 \cdot \ln(\delta^{18} O + 1)$

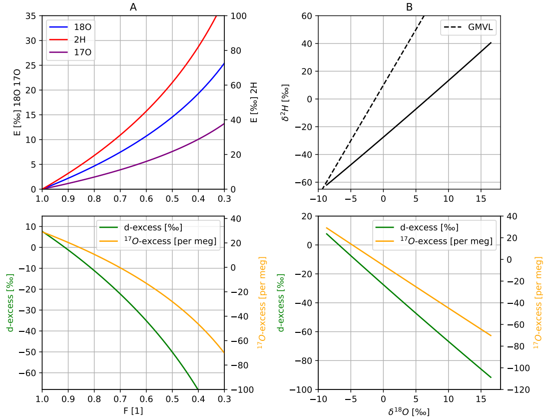
Pierchala et al. '22 – triple isotope / kinetic fract. PySDM setup[a]

[a] launch-in-the-cloud URL: `https://mybinder.org/v2/gh/open-atmos/PySDM.git/main?urlpath=lab/tree/examples/PySDM_examples/Pierchala_et_al_2022`

Fig. 3 (paper): measurements + model



PySDM: theoretical curves



$$E = \frac{[\text{heavy iso.}]}{[\text{light iso.}]} \Big/ \frac{[\text{heavy iso.}]}{[\text{light iso.}]}\Big|_{t=0} - 1 \qquad \delta = \frac{[\text{heavy iso.}]}{[\text{light iso.}]} \Big/ \frac{[\text{heavy iso.}]}{[\text{light iso.}]}\Big|_{\text{VSMOW}} - 1$$

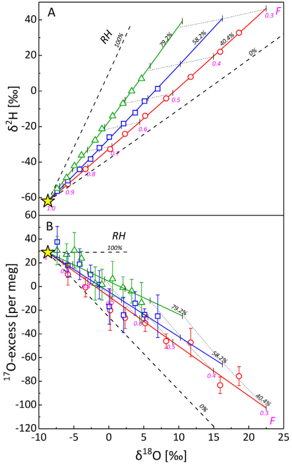
F: fraction of water remaining

d-excess: $\delta^2 H - 8 \cdot \delta^{18} O$ $\qquad$ ^{17}O-excess: $\ln(\delta^{17}O + 1) - 0.528 \cdot \ln(\delta^{18}O + 1)$

Pierchala et al. '22 – triple isotope / kinetic fract. PySDM setup[a]

[a] launch-in-the-cloud URL: `https://mybinder.org/v2/gh/open-atmos/PySDM.git/main?urlpath=lab/tree/examples/PySDM_examples/Pierchala_et_al_2022`

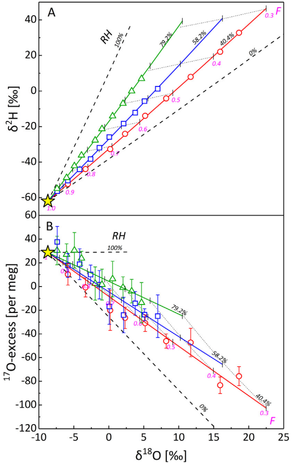
Fig. 4 (paper): RH varied



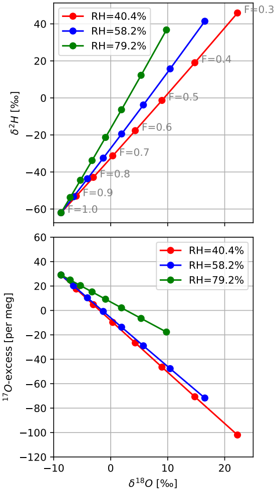
Pierchala et al. '22 – triple isotope / kinetic fract. PySDM setup[a]

[a] launch-in-the-cloud URL: `https://mybinder.org/v2/gh/open-atmos/PySDM.git/main?urlpath=lab/tree/examples/PySDM_examples/Pierchala_et_al_2022`

Fig. 4 (paper): RH varied



PySDM: model curves



Pierchala et al. '22 – triple isotope / kinetic fract. PySDM setup

pypi.org/p/PySDM-examples

literature reference	cond/evap	coalescence	isotopes	breakup	transport	chemistry	freezing	keywords
formulae-only								
Gedzelman & Arnold 1994			×					#dynsys
Pierchala et al. 2022			×					#lab-experiment
...								
0D box environment								
Berry 1967		×						#kernels
Shima et al. 2009		×						#analytic-solution
Alpert & Knopf 2016							×	#ABIFM
de Jong et al. 2023		×		×				#analytic-solution
...								
0D parcel environment								
Rozanski & Sonntag 1982	×		×					#iterative-parcel
Abdul-Razzak & Ghan 2000	×							#pysdm-vs-gcm-param
Kreidenweis et al. 2003	×					×		#Hoppel-gap
Arabas and Shima 2017	×							#dynsys
Jensen and Nugent 2017	×							#giant-CCN
Yang et al. 2018	×							#ripening
Lowe et al. 2019	×							#surfactants
Grabowski and Pawlowska 2023	×							#ripening
...								
1D single-column kinematic env. (advection: PyMPDATA)								
Shipway & Hill 2012	×	×			×			#KiD
deJong et al. 2023 (figures 6-8)	×	×		×	×			#KiD
...								
2D prescribed-flow environment (advection: PyMPDATA)								
Arabas et al. 2015	×	×			×			#GUI
Arabas et al. 2023 (figure 11)	×	×			×		×	#Paraview

Gedzelman & Arnold 1994: drop evaporation dynamics[a]

[a] https://github.com/open-atmos/PySDM/tree/main/examples/PySDM_examples/Gedzelman_and_Arnold_1994

$$\frac{d}{dt}\begin{bmatrix} m_{\text{rain}}^{\text{total}} \\ m_{\text{vap}}^{\text{total}} \\ \\ \\ \end{bmatrix} / A(m_{\text{rain}}^{\text{total}}) = F(\underbrace{T, m_{\text{vap}}^{\text{total}}}_{\text{RH}}, \quad , \underbrace{\ldots}_{\text{Re, Sc, Pr}})$$

Gedzelman & Arnold 1994: drop evaporation dynamics[a]

[a] https://github.com/open-atmos/PySDM/tree/main/examples/PySDM_examples/Gedzelman_and_Arnold_1994

$$\frac{d}{dt}\begin{bmatrix} m_{\text{rain}}^{\text{total}} \\ m_{\text{vap}}^{\text{total}} \\ m_{\text{rain}}^{\text{heavy}} \\ m_{\text{vap}}^{\text{heavy}} \\ \vdots \end{bmatrix} / A(m_{\text{rain}}^{\text{total}}) = F(\underbrace{T, m_{\text{vap}}^{\text{total}}}_{\text{RH}}, m_{\text{vap}}^{\text{heavy}}, m_{\text{rain}}^{\text{heavy}}, \underbrace{\ldots}_{\text{Re, Sc, Pr}})$$

Gedzelman & Arnold 1994: drop evaporation dynamics[a]

[a] https://github.com/open-atmos/PySDM/tree/main/examples/PySDM_examples/Gedzelman_and_Arnold_1994

$$\frac{d}{dt}\begin{bmatrix} m^{\text{total}}_{\text{rain}} \\ m^{\text{total}}_{\text{vap}} \\ m^{\text{heavy}}_{\text{rain}} \\ m^{\text{heavy}}_{\text{vap}} \\ \vdots \end{bmatrix} / A(m^{\text{total}}_{\text{rain}}) = F(\underbrace{T, m^{\text{total}}_{\text{vap}}}_{\text{RH}}, m^{\text{heavy}}_{\text{vap}}, m^{\text{heavy}}_{\text{rain}}, \underbrace{\ldots}_{\text{Re, Sc, Pr}})$$

$$T_{\text{rain}} \neq T$$
$$\text{RH} < 1$$

Gedzelman & Arnold 1994: drop evaporation dynamics[a]

[a] https://github.com/open-atmos/PySDM/tree/main/examples/PySDM_examples/Gedzelman_and_Arnold_1994

$$\frac{d}{dt}\begin{bmatrix} m^{\text{total}}_{\text{rain}} \\ m^{\text{total}}_{\text{vap}} \\ \color{red}{m^{\text{heavy}}_{\text{rain}}} \\ \color{red}{m^{\text{heavy}}_{\text{vap}}} \\ \vdots \end{bmatrix} / A(m^{\text{total}}_{\text{rain}}) = F(\underbrace{T, m^{\text{total}}_{\text{vap}}}_{\text{RH}}, \color{red}{m^{\text{heavy}}_{\text{vap}}}, \color{red}{m^{\text{heavy}}_{\text{rain}}}, \underbrace{\ldots}_{\text{Re, Sc, Pr}})$$

$$T_{\text{rain}} \neq T$$
$$\text{RH} < 1$$

$$\delta_{\text{rain}} + 1 = \frac{M^{\text{light}}}{M^{\text{heavy}}} \frac{m^{\text{heavy}}_{\text{rain}}}{m^{\text{total}}_{\text{rain}} - m^{\text{heavy}}_{\text{rain}}} / R_{\text{SMOW}}$$
$$\delta_{\text{vap}} + 1 = \ldots$$

Gedzelman & Arnold 1994: drop evaporation dynamics[a]

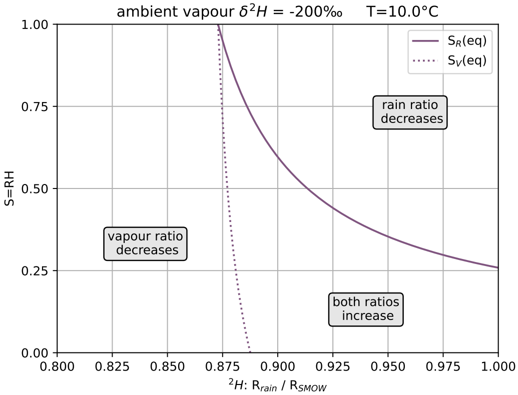
[a] https://github.com/open-atmos/PySDM/tree/main/examples/PySDM_examples/Gedzelman_and_Arnold_1994

$$\frac{d}{dt}\begin{bmatrix} m^{\text{total}}_{\text{rain}} \\ m^{\text{total}}_{\text{vap}} \\ \color{red}{m^{\text{heavy}}_{\text{rain}}} \\ \color{red}{m^{\text{heavy}}_{\text{vap}}} \\ \vdots \end{bmatrix} / A(m^{\text{total}}_{\text{rain}}) = F(\underbrace{T, m^{\text{total}}_{\text{vap}}}_{\text{RH}}, \color{red}{m^{\text{heavy}}_{\text{vap}}}, \color{red}{m^{\text{heavy}}_{\text{rain}}}, \underbrace{\ldots}_{\text{Re, Sc, Pr}})$$

$$T_{\text{rain}} \neq T$$
$$\text{RH} < 1$$

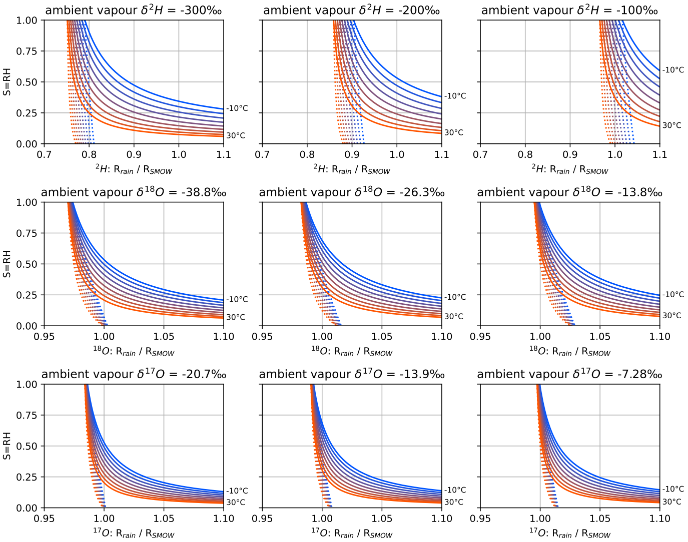
$$\delta_{\text{rain}} + 1 = \frac{M^{\text{light}}}{M^{\text{heavy}}} \frac{m^{\text{heavy}}_{\text{rain}}}{m^{\text{total}}_{\text{rain}} - m^{\text{heavy}}_{\text{rain}}} / R_{\text{SMOW}}$$
$$\delta_{\text{vap}} + 1 = \ldots$$

$$S_R \rightsquigarrow \frac{d\delta_{\text{rain}}}{dt} = 0$$
$$S_V \rightsquigarrow \frac{d\delta_{\text{vap}}}{dt} = 0$$



Gedzelman & Arnold 1994: drop evaporation dynamics[a]

[a] https://github.com/open-atmos/PySDM/tree/main/examples/PySDM_examples/Gedzelman_and_Arnold_1994



water isotopes in PySDM: summary, next steps

implemented features (incl. tests against lab and model literature data):

- base SD attributes: #moles of heavy isotopologue

water isotopes in PySDM: summary, next steps

implemented features **(incl. tests against lab and model literature data):**

- base SD attributes: #moles of heavy isotopologue
- derived SD attributed: #moles of light isotopologue, δ, ...

water isotopes in PySDM: summary, next steps

implemented features **(incl. tests against lab and model literature data):**

- base SD attributes: #moles of heavy isotopologue
- derived SD attributed: #moles of light isotopologue, δ, ...
- equilibrium & non-equilibrium fractionation coeffs. for: ^{2}H, ^{18}O & ^{17}O

water isotopes in PySDM: summary, next steps

implemented features (incl. tests against lab and model literature data):

- base SD attributes: #moles of heavy isotopologue
- derived SD attributed: #moles of light isotopologue, δ, ...
- equilibrium & non-equilibrium fractionation coeffs. for: ^{2}H, ^{18}O & ^{17}O
- dynamics: differential (Merlivat & Jouzel '79, Gedzelan & Arnold 1994)
  & integral (Rayleigh distillation)

water isotopes in PySDM: summary, next steps

implemented features **(incl. tests against lab and model literature data):**

- base SD attributes: #moles of heavy isotopologue
- derived SD attributed: #moles of light isotopologue, δ, ...
- equilibrium & non-equilibrium fractionation coeffs. for: ^{2}H, ^{18}O & ^{17}O
- dynamics: differential (Merlivat & Jouzel '79, Gedzelan & Arnold 1994) & integral (Rayleigh distillation)
- minimal framework: iterative parcel runs towards stationary state

water isotopes in PySDM: summary, next steps

implemented features **(incl. tests against lab and model literature data):**

- base SD attributes: #moles of heavy isotopologue
- derived SD attributed: #moles of light isotopologue, δ, ...
- equilibrium & non-equilibrium fractionation coeffs. for: ^{2}H, ^{18}O & ^{17}O
- dynamics: differential (Merlivat & Jouzel '79, Gedzelan & Arnold 1994) & integral (Rayleigh distillation)
- minimal framework: iterative parcel runs towards stationary state

next steps:

- more comprehensive simulation setups (e.g., 2D prescribed-flow)

water isotopes in PySDM: summary, next steps

implemented features **(incl. tests against lab and model literature data):**

- base SD attributes: #moles of heavy isotopologue
- derived SD attributed: #moles of light isotopologue, δ, ...
- equilibrium & non-equilibrium fractionation coeffs. for: ^{2}H, ^{18}O & ^{17}O
- dynamics: differential (Merlivat & Jouzel '79, Gedzelan & Arnold 1994) & integral (Rayleigh distillation)
- minimal framework: iterative parcel runs towards stationary state

next steps:

- more comprehensive simulation setups (e.g., 2D prescribed-flow)
- ventilation (mass & heat budget)

water isotopes in PySDM: summary, next steps

implemented features (incl. tests against lab and model literature data):

- base SD attributes: #moles of heavy isotopologue
- derived SD attributed: #moles of light isotopologue, δ, ...
- equilibrium & non-equilibrium fractionation coeffs. for: ^{2}H, ^{18}O & ^{17}O
- dynamics: differential (Merlivat & Jouzel '79, Gedzelan & Arnold 1994) & integral (Rayleigh distillation)
- minimal framework: iterative parcel runs towards stationary state

next steps:

- more comprehensive simulation setups (e.g., 2D prescribed-flow)
- ventilation (mass & heat budget)
- exploring dependence on droplet and precip size spectra (and hence aerosol)

water isotopes in PySDM: summary, next steps

implemented features (incl. tests against lab and model literature data):

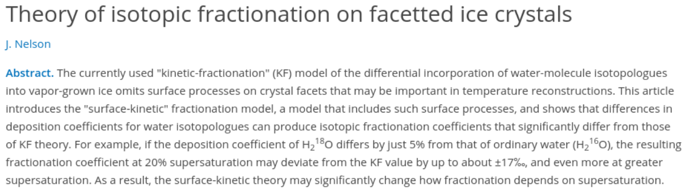
- base SD attributes: #moles of heavy isotopologue
- derived SD attributed: #moles of light isotopologue, δ, ...
- equilibrium & non-equilibrium fractionation coeffs. for: ^{2}H, ^{18}O & ^{17}O
- dynamics: differential (Merlivat & Jouzel '79, Gedzelan & Arnold 1994) & integral (Rayleigh distillation)
- minimal framework: iterative parcel runs towards stationary state

next steps:

- more comprehensive simulation setups (e.g., 2D prescribed-flow)
- ventilation (mass & heat budget)
- exploring dependence on droplet and precip size spectra (and hence aerosol)
- ice-phase processes

motivation slide!

supersaturation vs. temperature reconstructions



Theory of isotopic fractionation on facetted ice crystals

J. Nelson

Abstract. The currently used "kinetic-fractionation" (KF) model of the differential incorporation of water-molecule isotopologues into vapor-grown ice omits surface processes on crystal facets that may be important in temperature reconstructions. This article introduces the "surface-kinetic" fractionation model, a model that includes such surface processes, and shows that differences in deposition coefficients for water isotopologues can produce isotopic fractionation coefficients that significantly differ from those of KF theory. For example, if the deposition coefficient of $H_2{}^{18}O$ differs by just 5% from that of ordinary water ($H_2{}^{16}O$), the resulting fractionation coefficient at 20% supersaturation may deviate from the KF value by up to about ±17‰, and even more at greater supersaturation. As a result, the surface-kinetic theory may significantly change how fractionation depends on supersaturation.

motivation slide!

supersaturation vs. temperature reconstructions

Theory of isotopic fractionation on facetted ice crystals

J. Nelson

Abstract. The currently used "kinetic-fractionation" (KF) model of the differential incorporation of water-molecule isotopologues into vapor-grown ice omits surface processes on crystal facets that may be important in temperature reconstructions. This article introduces the "surface-kinetic" fractionation model, a model that includes such surface processes, and shows that differences in deposition coefficients for water isotopologues can produce isotopic fractionation coefficients that significantly differ from those of KF theory. For example, if the deposition coefficient of $H_2{}^{18}O$ differs by just 5% from that of ordinary water ($H_2{}^{16}O$), the resulting fractionation coefficient at 20% supersaturation may deviate from the KF value by up to about ±17‰, and even more at greater supersaturation. As a result, the surface-kinetic theory may significantly change how fractionation depends on supersaturation.

using water-isotope data to investigate turbulence

Study of Mass Transfer at the Air-Water Interface by an Isotopic Method

L. Merlivat

Departement de Recherche et Analyse, Centre d'Études Nucléaires de Saclay
Gif Sur Yvette, France

M. Coantic

Institut de Mécanique Statistique de la Turbulence, Marseille, France

The calculation of the evaporation flux is based on certain assumptions concerning processes in the vicinity of the air-water interface. Most of the recently proposed evaporation theories differ mainly in the estimated contributions of molecular and turbulent mass transfer in the vapor phase just above the liquid surface. This paper will show that, by analyzing the hydrogen and oxygen stable isotope distribution in liquid and water vapor, the processes taking place on a very small scale near the liquid can be investigated. The effect of molecular mass transfer is directly obtained without having to perform difficult measurements in the air in the immediate vicinity of the water surface. Experiments are carried out in the Institut de Mécanique Statistique de la Turbulence air-water tunnel specially designed for the simulation

github.com/open-atmos/PySDM

Acknowledgements:
PySDM contributors; prof. Kazimierz Różański (water isotopes)

Funding:
National Science Centre Poland
(grant no. 2020/39/D/ST10/01220)

github.com/open-atmos/PySDM

Acknowledgements:
PySDM contributors; prof. Kazimierz Różański (water isotopes)

Funding:
National Science Centre Poland
(grant no. 2020/39/D/ST10/01220)

merci de votre attention