PyMPDATA: an open-source, example-rich, just-in-time compiled implementation of MPDATA finite-difference scheme

---

Sylwester Arabas
AGH University of Krakow, Poland

ESCO 2026 Software Afternoon

- **MPDATA scheme and its implementations**

MPDATA key concepts: UPWIND discretisation

- advection equation / scalar conservation law:

$$\partial_t(G\psi) + \nabla \cdot (\mathbf{v}\psi) = GR$$

$\psi(\mathbf{x}, t)$: advected scalar field (advectee),
$\mathbf{v} = \{u, \ldots\} = G\dot{\mathbf{x}}$: flow velocity vector field (advector),
$G(\mathbf{x})$: fluid density, Jacobian of coordinate transformation, or their product

MPDATA key concepts: UPWIND discretisation

- advection equation / scalar conservation law:

$$\partial_t(G\psi) + \nabla \cdot (\mathbf{v}\psi) = GR$$

$\psi(\mathbf{x}, t)$: advected scalar field (advectee),
$\mathbf{v} = \{u, \ldots\} = G\dot{\mathbf{x}}$: flow velocity vector field (advector),
$G(\mathbf{x})$: fluid density, Jacobian of coordinate transformation, or their product

- UPWIND discretisation on a staggered grid (n numbers time steps, i grid steps):

$$\frac{\psi_i^{n+1} - \psi_i^n}{\Delta t} + \frac{\overbrace{f(\psi_i^n, \psi_{i+1}^n, u_{i+1/2}^n)}^{\text{right-hand wall flux}} - \overbrace{f(\psi_{i-1}^n, \psi_i^n, u_{i-1/2}^n)}^{\text{left-hand wall flux}}}{\Delta x} = 0$$

MPDATA key concepts: UPWIND discretisation

- advection equation / scalar conservation law:

$$\partial_t(G\psi) + \nabla \cdot (\mathbf{v}\psi) = GR$$

$\psi(\mathbf{x}, t)$: advected scalar field (advectee),
$\mathbf{v} = \{u, \ldots\} = G\dot{\mathbf{x}}$: flow velocity vector field (advector),
$G(\mathbf{x})$: fluid density, Jacobian of coordinate transformation, or their product

- UPWIND discretisation on a staggered grid (n numbers time steps, i grid steps):

$$\frac{\psi_i^{n+1} - \psi_i^n}{\Delta t} + \frac{\overbrace{f(\psi_i^n, \psi_{i+1}^n, u_{i+1/2}^n)}^{\text{right-hand wall flux}} - \overbrace{f(\psi_{i-1}^n, \psi_i^n, u_{i-1/2}^n)}^{\text{left-hand wall flux}}}{\Delta x} = 0$$

$$\partial_t \psi|_i^n + u \partial_x \psi|_i^n = \underbrace{\left(|u| \frac{\Delta x}{2} - u^2 \frac{\Delta t}{2} \right)}_{k\ -\ \text{numerical diffusion}} \partial_x^2 \psi|_i^n$$

(e.g., Roberts & Weiss 1966, doi:10.2307/2003507)

MPDATA key concepts: antidiffusive pseudo-velocities

- diffusion can be cast as advection with a pseudo-velocity:

$$\partial_t \psi = k \partial_x^2 \psi + \ldots \quad \rightsquigarrow \quad \partial_t \psi = \partial_x (k \underbrace{\frac{\partial_x \psi}{\psi}}_{\text{pseudo-velocity}} \psi) + \ldots$$

(e.g., Lange 1973, doi:10.2172/4308175)

MPDATA key concepts: antidiffusive pseudo-velocities

- diffusion can be cast as advection with a pseudo-velocity:

$$\partial_t \psi = k \partial_x^2 \psi + \ldots \quad \rightsquigarrow \quad \partial_t \psi = \partial_x (\underbrace{k \frac{\partial_x \psi}{\psi}}_{\text{pseudo-velocity}} \psi) + \ldots$$

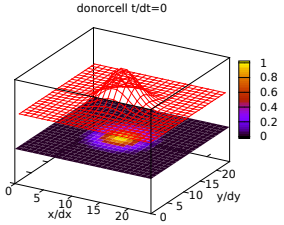
(e.g., Lange 1973, doi:10.2172/4308175)

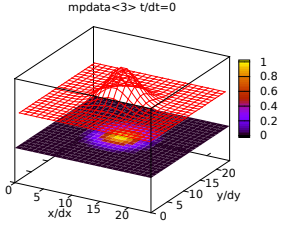
- "**Smolarkiewicz** algorithm" (MPDATA): upwind-integrate backwards-in-time, with an anti-diffusive pseudo velocity to reverse the effects of numerical diffusion, iteratively (m numbers iteration)

$$C^{m+1}_{i-1/2} = \frac{\Delta t}{\Delta x} k^m_{i-1/2} \left. \frac{\partial_x \psi}{\psi} \right|^m_{i-1/2} \approx \begin{cases} 0 & \text{if } \psi^m_i + \psi^m_{i-1} = 0 \\ \left[|C^m_{i-1/2}| - (C^m_{i-1/2})^2 \right] \frac{\psi^m_i - \psi^m_{i-1}}{\psi^m_i + \psi^m_{i-1}} & \text{otherwise} \end{cases}$$

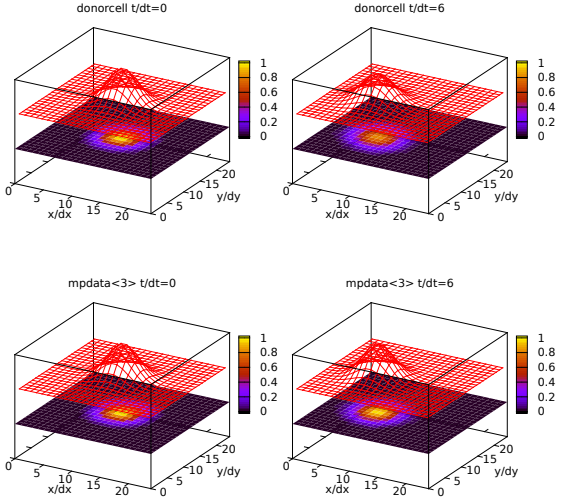
(Smolarkiewicz 1983 MWR, 1984 JCP: doi:10.1016/0021-9991(84)90121-9)

MPDATA: ”M” for multi-dimensional (2D is not a sum of two 1D MPDATAs)

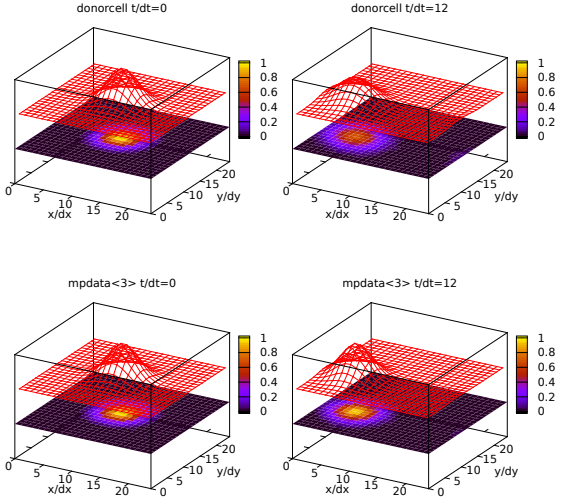




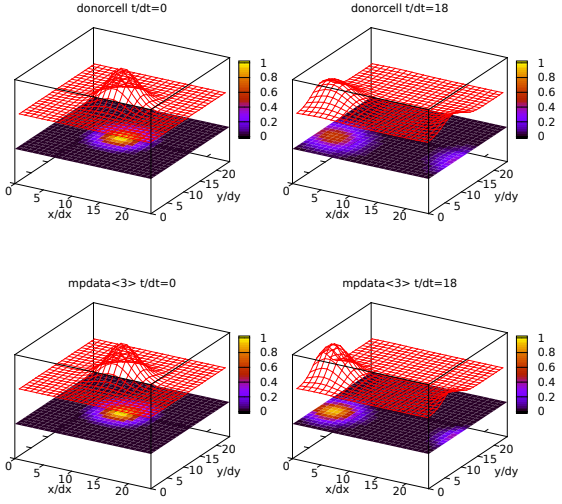
MPDATA: ”M” for multi-dimensional (2D is not a sum of two 1D MPDATAs)



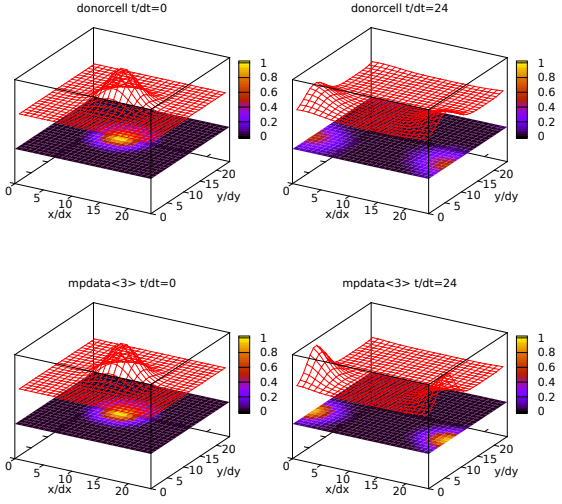
MPDATA: ”M” for multi-dimensional (2D is not a sum of two 1D MPDATAs)



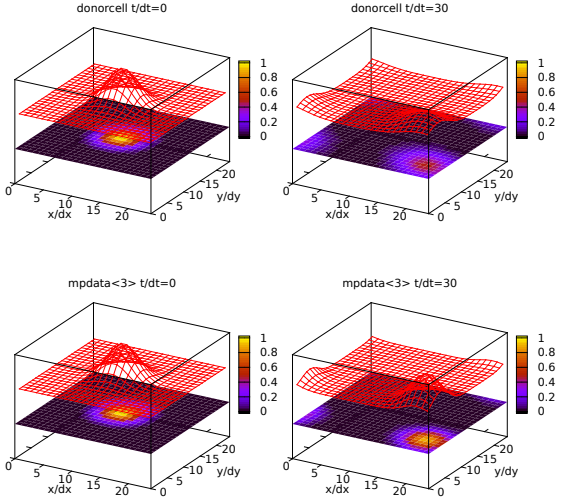
MPDATA: ”M” for multi-dimensional (2D is not a sum of two 1D MPDATAs)



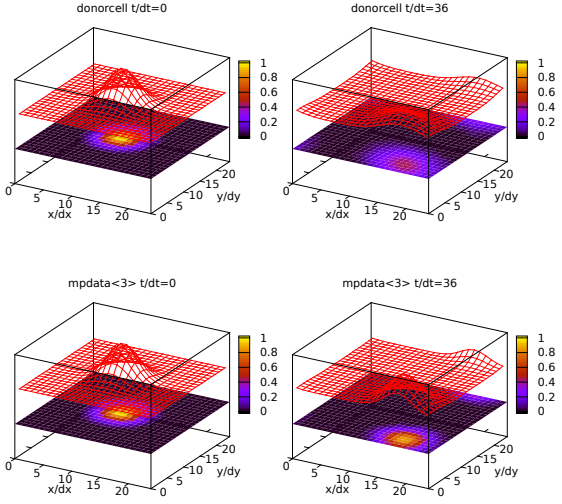
MPDATA: ”M” for multi-dimensional (2D is not a sum of two 1D MPDATAs)



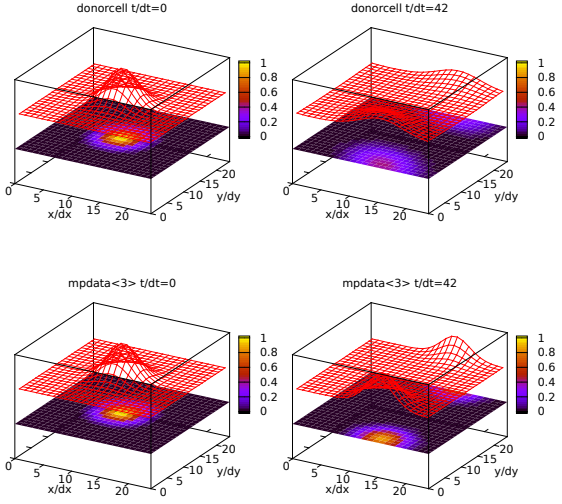
MPDATA: ”M” for multi-dimensional (2D is not a sum of two 1D MPDATAs)



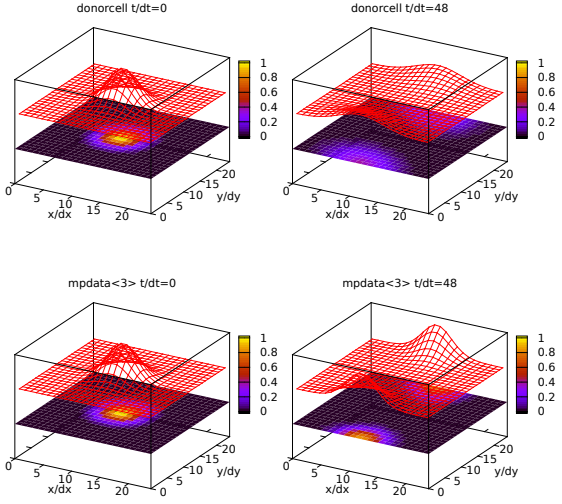
MPDATA: ”M” for multi-dimensional (2D is not a sum of two 1D MPDATAs)



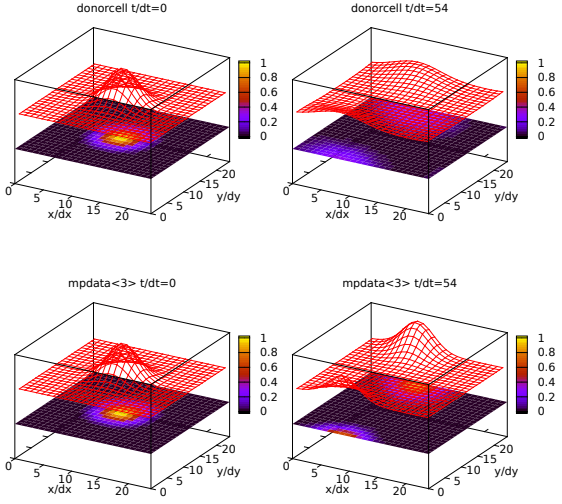
MPDATA: ”M” for multi-dimensional (2D is not a sum of two 1D MPDATAs)



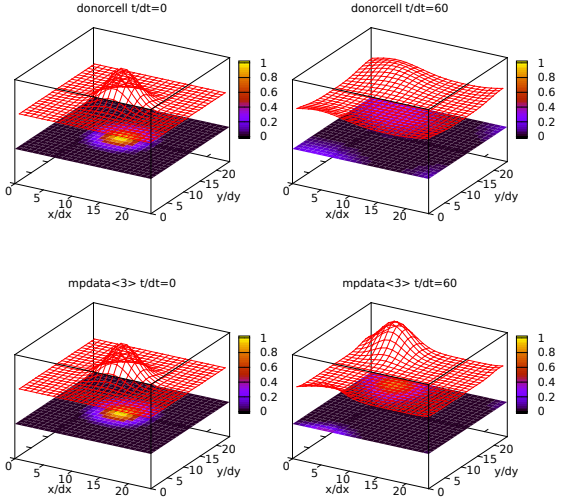
MPDATA: ”M” for multi-dimensional (2D is not a sum of two 1D MPDATAs)



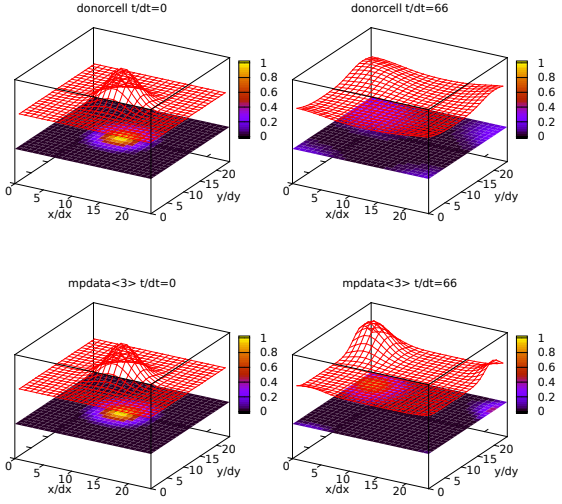
MPDATA: ”M” for multi-dimensional (2D is not a sum of two 1D MPDATAs)



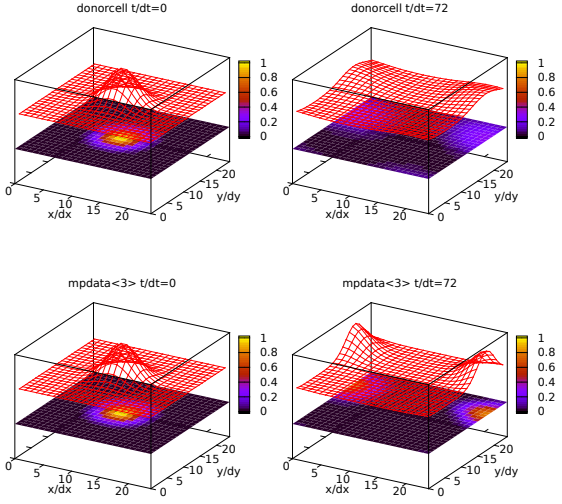
MPDATA: ”M” for multi-dimensional (2D is not a sum of two 1D MPDATAs)



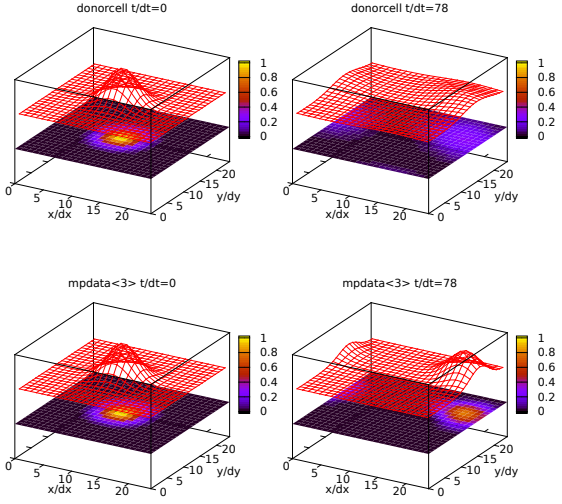
MPDATA: ”M” for multi-dimensional (2D is not a sum of two 1D MPDATAs)



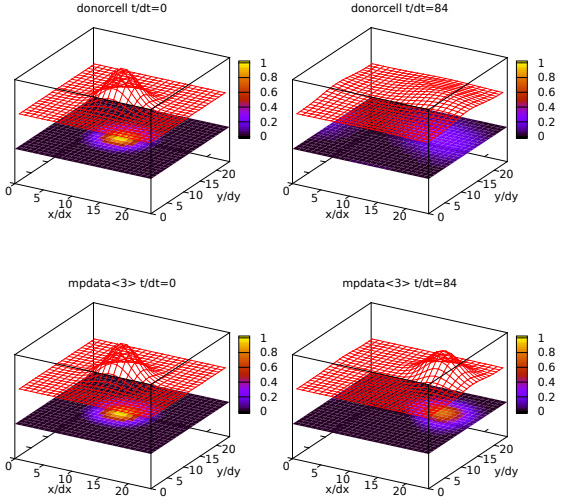
MPDATA: ”M” for multi-dimensional (2D is not a sum of two 1D MPDATAs)



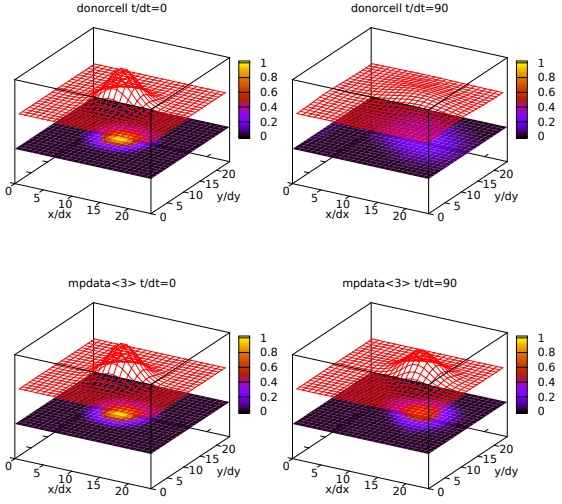
MPDATA: ”M” for multi-dimensional (2D is not a sum of two 1D MPDATAs)



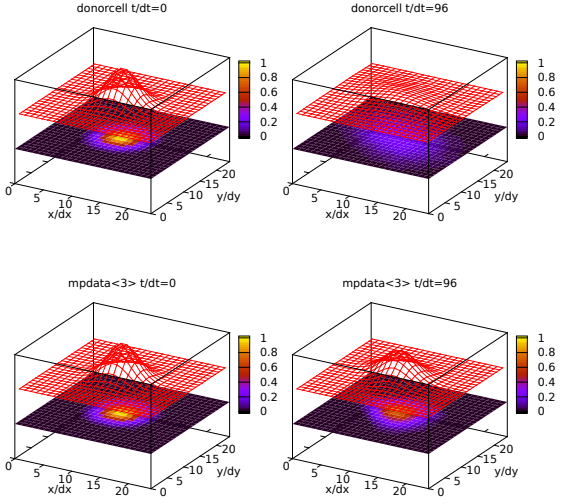
MPDATA: ”M” for multi-dimensional (2D is not a sum of two 1D MPDATAs)



MPDATA: ”M” for multi-dimensional (2D is not a sum of two 1D MPDATAs)



MPDATA: ”M” for multi-dimensional (2D is not a sum of two 1D MPDATAs)



MPDATA variants (structured grid)

- basic (+iterations): Smolarkiewicz 1983

MPDATA variants (structured grid)

- basic (+iterations): Smolarkiewicz 1983
- n-dimensions, diffusion, divergent flow corrections: Smolarkiewicz 1984

MPDATA variants (structured grid)

- basic (+iterations): Smolarkiewicz 1983
- n-dimensions, diffusion, divergent flow corrections: Smolarkiewicz 1984
- coordinate transformation: Smolarkiewicz and Clark 1986, Smolarkiewicz and Margolin 1993

MPDATA variants (structured grid)

- basic (+iterations): Smolarkiewicz 1983
- n-dimensions, diffusion, divergent flow corrections: Smolarkiewicz 1984
- coordinate transformation: Smolarkiewicz and Clark 1986, Smolarkiewicz and Margolin 1993
- resursive anti-diffusive velocities: Beason & Margolin 1988

MPDATA variants (structured grid)

- basic (+iterations): Smolarkiewicz 1983
- n-dimensions, diffusion, divergent flow corrections: Smolarkiewicz 1984
- coordinate transformation: Smolarkiewicz and Clark 1986, Smolarkiewicz and Margolin 1993
- resursive anti-diffusive velocities: Beason & Margolin 1988
- flux-corrected transport: Smolarkiewicz and Grabowski 1990

MPDATA variants (structured grid)

- basic (+iterations): Smolarkiewicz 1983
- n-dimensions, diffusion, divergent flow corrections: Smolarkiewicz 1984
- coordinate transformation: Smolarkiewicz and Clark 1986, Smolarkiewicz and Margolin 1993
- resursive anti-diffusive velocities: Beason & Margolin 1988
- flux-corrected transport: Smolarkiewicz and Grabowski 1990
- third-order terms: Smolarkiewicz and Margolin 1998

MPDATA variants (structured grid)

- basic (+iterations): Smolarkiewicz 1983
- n-dimensions, diffusion, divergent flow corrections: Smolarkiewicz 1984
- coordinate transformation: Smolarkiewicz and Clark 1986, Smolarkiewicz and Margolin 1993
- resursive anti-diffusive velocities: Beason & Margolin 1988
- flux-corrected transport: Smolarkiewicz and Grabowski 1990
- third-order terms: Smolarkiewicz and Margolin 1998
- infinite-gauge variant: Smolarkiewicz 2006

MPDATA variants (structured grid)

- basic (+iterations): Smolarkiewicz 1983
- n-dimensions, diffusion, divergent flow corrections: Smolarkiewicz 1984
- coordinate transformation: Smolarkiewicz and Clark 1986, Smolarkiewicz and Margolin 1993
- resursive anti-diffusive velocities: Beason & Margolin 1988
- flux-corrected transport: Smolarkiewicz and Grabowski 1990
- third-order terms: Smolarkiewicz and Margolin 1998
- infinite-gauge variant: Smolarkiewicz 2006
- fully third-order variant: Waruszewski et al. 2018

MPDATA implementations

known closed-source:

- NCAR EULAG
- COSMO (alternative dynamical core)
- ECMWF IFS-FVM

open-source:

integrated into CFD packages:

MPDATA implementations

known closed-source:

- NCAR EULAG
- COSMO (alternative dynamical core)
- ECMWF IFS-FVM

open-source:

integrated into CFD packages:

AtmosFOAM	C++ & Python/Numba	3D	/AtmosFOAM	U. Reading

MPDATA implementations

known closed-source:

- NCAR EULAG
- COSMO (alternative dynamical core)
- ECMWF IFS-FVM

open-source:

integrated into CFD packages:

AtmosFOAM	C++ & Python/Numba	3D	/AtmosFOAM	U. Reading
ROMS	FORTRAN	3D	/myroms	UCLA (?)

MPDATA implementations

known closed-source:

- NCAR EULAG
- COSMO (alternative dynamical core)
- ECMWF IFS-FVM

open-source:

integrated into CFD packages:

AtmosFOAM	C++ & Python/Numba	3D	/AtmosFOAM	U. Reading
ROMS	FORTRAN	3D	/myroms	UCLA (?)
PISM	C++	2D	/pism	U. Alaska Fairbanks

MPDATA implementations

known closed-source:

- NCAR EULAG
- COSMO (alternative dynamical core)
- ECMWF IFS-FVM

open-source:

integrated into CFD packages:

AtmosFOAM	C++ & Python/Numba	3D	/AtmosFOAM	U. Reading
ROMS	FORTRAN	3D	/myroms	UCLA (?)
PISM	C++	2D	/pism	U. Alaska Fairbanks
babyEULAG	FORTRAN	2D	/igfuw/bE_SDs	NCAR

MPDATA implementations

known closed-source:

- NCAR EULAG
- COSMO (alternative dynamical core)
- ECMWF IFS-FVM

open-source:

integrated into CFD packages:

AtmosFOAM	C++ & Python/Numba	3D	/AtmosFOAM	U. Reading
ROMS	FORTRAN	3D	/myroms	UCLA (?)
PISM	C++	2D	/pism	U. Alaska Fairbanks
babyEULAG	FORTRAN	2D	/igfuw/bE_SDs	NCAR
apc-llc/mpdata	C/CUDA	3D	/apc-llc	Rus. Acad. Sci.

MPDATA implementations

known closed-source:

- NCAR EULAG
- COSMO (alternative dynamical core)
- ECMWF IFS-FVM

open-source:

integrated into CFD packages:

AtmosFOAM	C++ & Python/Numba	3D	/AtmosFOAM	U. Reading
ROMS	FORTRAN	3D	/myroms	UCLA (?)
PISM	C++	2D	/pism	U. Alaska Fairbanks
babyEULAG	FORTRAN	2D	/igfuw/bE_SDs	NCAR
apc-llc/mpdata	C/CUDA	3D	/apc-llc	Rus. Acad. Sci.
ICAR	FORTRAN	3D	/NCAR/icar	NCAR

MPDATA implementations

known closed-source:

- NCAR EULAG
- COSMO (alternative dynamical core)
- ECMWF IFS-FVM

open-source:

integrated into CFD packages:

AtmosFOAM	C++ & Python/Numba	3D	/AtmosFOAM	U. Reading
ROMS	FORTRAN	3D	/myroms	UCLA (?)
PISM	C++	2D	/pism	U. Alaska Fairbanks
babyEULAG	FORTRAN	2D	/igfuw/bE_SDs	NCAR
apc-llc/mpdata	C/CUDA	3D	/apc-llc	Rus. Acad. Sci.
ICAR	FORTRAN	3D	/NCAR/icar	NCAR
sbPOM	FORTRAN	2D	/RinceWND/extPOM	Stony Brook

MPDATA implementations

known closed-source:

- NCAR EULAG
- COSMO (alternative dynamical core)
- ECMWF IFS-FVM

open-source:

integrated into CFD packages:

AtmosFOAM	C++ & Python/Numba	3D	/AtmosFOAM	U. Reading
ROMS	FORTRAN	3D	/myroms	UCLA (?)
PISM	C++	2D	/pism	U. Alaska Fairbanks
babyEULAG	FORTRAN	2D	/igfuw/bE_SDs	NCAR
apc-llc/mpdata	C/CUDA	3D	/apc-llc	Rus. Acad. Sci.
ICAR	FORTRAN	3D	/NCAR/icar	NCAR
sbPOM	FORTRAN	2D	/RinceWND/extPOM	Stony Brook

reusable:

libmpdata++	C++/Blitz++	1,2,3D	/igfuw	UWarsaw

MPDATA implementations

known closed-source:

- NCAR EULAG
- COSMO (alternative dynamical core)
- ECMWF IFS-FVM

open-source:

integrated into CFD packages:

AtmosFOAM	C++ & Python/Numba	3D	/AtmosFOAM	U. Reading
ROMS	FORTRAN	3D	/myroms	UCLA (?)
PISM	C++	2D	/pism	U. Alaska Fairbanks
babyEULAG	FORTRAN	2D	/igfuw/bE_SDs	NCAR
apc-llc/mpdata	C/CUDA	3D	/apc-llc	Rus. Acad. Sci.
ICAR	FORTRAN	3D	/NCAR/icar	NCAR
sbPOM	FORTRAN	2D	/RinceWND/extPOM	Stony Brook

reusable:

libmpdata++	C++/Blitz++	1,2,3D	/igfuw	UWarsaw
PyMPDATA	Python/Numba	1,2,3D	/open-atmos	UJ/AGH.edu.pl

- **PyMPDATA: pure-Python just-in-time compiled MPDATA**

- Numba JIT $\rightsquigarrow$ pure-Python code with compiled-language performance (plus OpenMP-like multi-threading)

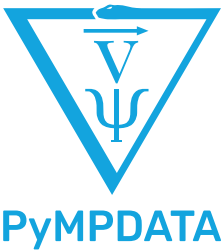
PyMPDATA: 100% Python codebase

```python
@numba.njit(**options.jit_flags)
def a_term(psi):
    """eq. 13 in [Smolarkiewicz 1984](https://doi.org/10.1016/0021-9991(84)90121-9);
    eq. 17a in [Smolarkiewicz & Margolin 1998](https://doi.org/10.1006/jcph.1998.5901)"""
    result = ats(*psi, 1) - ats(*psi, 0)
    if infinite_gauge:
        return result / 2
    return result / (ats(*psi, 1) + ats(*psi, 0) + epsilon)



@numba.njit(**options.jit_flags)
def b_term(psi):
    """eq. 13 in [Smolarkiewicz 1984](https://doi.org/10.1016/0021-9991(84)90121-9);
    eq. 17b in [Smolarkiewicz & Margolin 1998](https://doi.org/10.1006/jcph.1998.5901)"""
    result = ats(*psi, 1, 1) + ats(*psi, 0, 1) - ats(*psi, 1, -1) - ats(*psi, 0, -1)
    if infinite_gauge:
        return result / 4

    return result / (
        ats(*psi, 1, 1)
        + ats(*psi, 0, 1)
        + ats(*psi, 1, -1)
        + ats(*psi, 0, -1)
        + epsilon
    )
```



- Numba JIT $\rightsquigarrow$ pure-Python code with compiled-language performance (plus OpenMP-like multi-threading)
- runs on Linux, macOS & Windows (no compilation, just "pip install")

PyMPDATA: design goals, tech stack, features

- Numba JIT $\rightsquigarrow$ pure-Python code with compiled-language performance (plus OpenMP-like multi-threading)
- runs on Linux, macOS & Windows (no compilation, just "pip install")
- $> 90\%$ unit-test coverage (codecov) and growing...

PyMPDATA: design goals, tech stack, features

- Numba JIT $\rightsquigarrow$ pure-Python code with compiled-language performance (plus OpenMP-like multi-threading)
- runs on Linux, macOS & Windows (no compilation, just "pip install")
- $> 90\%$ unit-test coverage (codecov) and growing...
- API: 5 classes & single "advance" method

PyMPDATA: design goals, tech stack, features

- Numba JIT $\rightsquigarrow$ pure-Python code with compiled-language performance (plus OpenMP-like multi-threading)
- runs on Linux, macOS & Windows (no compilation, just "pip install")
- $> 90\%$ unit-test coverage (codecov) and growing...
- API: 5 classes & single "advance" method
- array-traversal abstractions avoiding explicit fill-halo or barrier calls

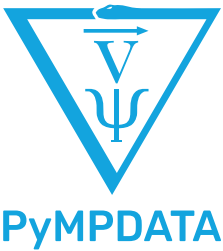
PyMPDATA: design goals, tech stack, features

- Numba JIT $\rightsquigarrow$ pure-Python code with compiled-language performance (plus OpenMP-like multi-threading)
- runs on Linux, macOS & Windows (no compilation, just "pip install")
- $> 90\%$ unit-test coverage (codecov) and growing...
- API: 5 classes & single "advance" method
- array-traversal abstractions avoiding explicit fill-halo or barrier calls
- docs (and CI) covers usage from Python, Julia, Matlab and Rust

PyMPDATA: design goals, tech stack, features

- Numba JIT $\rightsquigarrow$ pure-Python code with compiled-language performance (plus OpenMP-like multi-threading)
- runs on Linux, macOS & Windows (no compilation, just "pip install")
- $> 90\%$ unit-test coverage (codecov) and growing...
- API: 5 classes & single "advance" method
- array-traversal abstractions avoiding explicit fill-halo or barrier calls
- docs (and CI) covers usage from Python, Julia, Matlab and Rust
- suite of 20+ Jupyter notebook examples maintained with the project all with badges enabling **single-click execution on Colab**

PyMPDATA: design goals, tech stack, features



- Numba JIT $\rightsquigarrow$ pure-Python code with compiled-language performance (plus OpenMP-like multi-threading)
- runs on Linux, macOS & Windows (no compilation, just "pip install")
- $> 90\%$ unit-test coverage (codecov) and growing...
- API: 5 classes & single "advance" method
- array-traversal abstractions avoiding explicit fill-halo or barrier calls
- docs (and CI) covers usage from Python, Julia, Matlab and Rust
- suite of 20+ Jupyter notebook examples maintained with the project all with badges enabling **single-click execution on Colab**
- examples in 1D, 2D & 3D: advection-diffusion, bin cloud μ-physics, spherical coordinates, shallow-water, Black-Scholes, Burgers, Boussinesq

- **PyMPDATA-examples, documentation & usage in research**

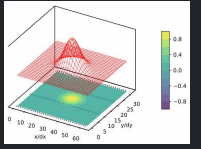
PyMPDATA & PyMPDATA-examples docs: open-atmos.io/PyMPDATA

PyMPDATA

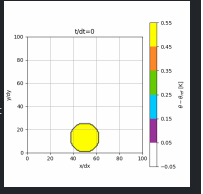
Documentation

What is PyMPDATA?

PyMPDATA is a **Numba-accelerated** multi-threaded Pythonic implementation of the **MPDATA algorithm** of Smolarkiewicz et al. used in geophysical fluid dynamics and beyond for **numerically solving generalised convection-diffusion PDEs**. PyMPDATA supports integration in 1D, 2D and 3D structured meshes with optional coordinate transformations. The first animation shown depicts a "hello-world" 2D advection-only simulation with dotted lines indicating domain decomposition across three threads. The second animation depicts an MPDATA solution to coupled mass and momentum conservation equations for a buoyancy-driven flow in Boussinesq approximation (see Jaruga et al. 2015 example).



A separate project called **PyMPDATA-MPI** depicts how **numba-mpi** can be used to enable distributed memory parallelism in PyMPDATA.



What is the difference between PyMPDATA and PyMPDATA-examples?

PyMPDATA is a Python package that provides the MPDATA algorithm implementation. It is a library that can be used in your own projects.

PyMPDATA-examples is a Python package that provides examples of how to use PyMPDATA. It includes common Python modules used in PyMPDATA smoke tests and in example Jupyter notebooks (but the package wheels do not include the notebooks, only .py files imported from the notebooks and PyMPDATA tests).

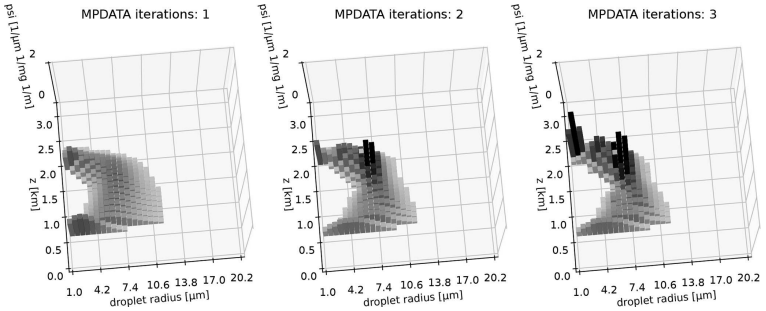
MPDATA for condensational growth in bin μ-physics (Olesik et al. 2022)

Geosci. Model Dev., 15, 3879–3899, 2022
https://doi.org/10.5194/gmd-15-3879-2022

Geoscientific Model Development
Open Access
EGU

On numerical broadening of particle-size spectra: a condensational growth study using PyMPDATA 1.0

Michael A. Olesik[1], **Jakub Banaśkiewicz**[2], **Piotr Bartman**[2], **Manuel Baumgartner**[3,4], **Simon Unterstrasser**[5], **and Sylwester Arabas**[6,2]



- spectro-spatial advection (single-column model)
- spectral broadening vs. MPDATA options

Monte-Carlo collisional breakup with “super-droplets” (de Jong et al. 2023)

Geosci. Model Dev., 16, 4193–4211, 2023
https://doi.org/10.5194/gmd-16-4193-2023

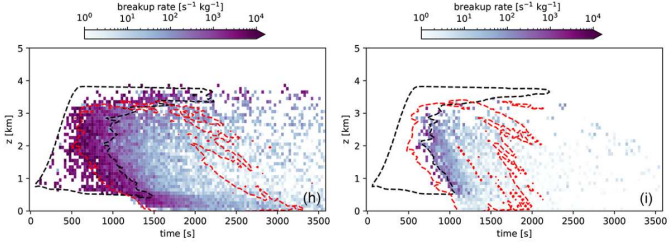
Geoscientific Model Development
Open Access
EGU

Development and technical paper

Breakups are complicated: an efficient representation of collisional breakup in the superdroplet method

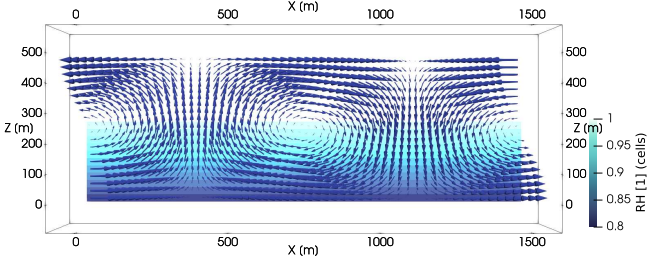
Emily de Jong[1], **John Ben Mackay**[2,a], **Oleksii Bulenok**[3], **Anna Jaruga**[4], **and Sylwester Arabas**[5,b,c]

[1]Department of Mechanical and Civil Engineering, California Institute of Technology, Pasadena, CA, USA
[2]Scripps Institution of Oceanography, San Diego, CA, USA
[3]Faculty of Mathematics and Computer Science, Jagiellonian University, Kraków, Poland
[4]Department of Environmental Science and Engineering, California Institute of Technology, Pasadena, CA, USA
[5]Faculty of Physics and Applied Computer Science, AGH University of Krakow, Kraków, Poland
[a]formerly at: Department of Environmental Science and Engineering, California Institute of Technology, Pasadena, CA, USA
[b]formerly at: Department of Atmospheric Sciences, University of Illinois Urbana-Champaign, Urbana, IL, USA
[c]formerly at: Faculty of Mathematics and Computer Science, Jagiellonian University, Kraków, Poland

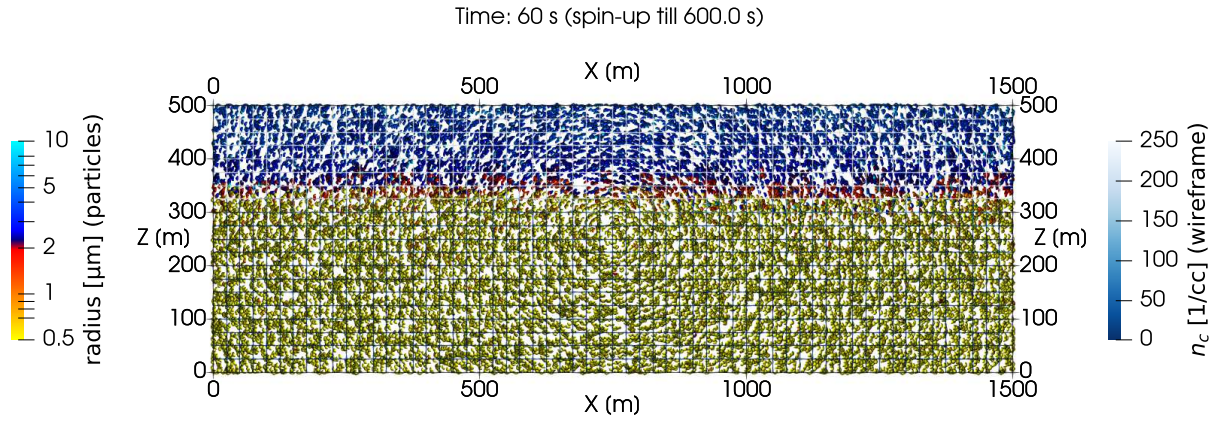


immersion freezing in clouds (Arabas et al. 2025, JAMES)

https://www.reuters.com/markets/commodities/
making-snow-stick-wind-challenges-winter-games-slope-makers-2021-11-29/



immersion freezing in clouds (Arabas et al. 2025, JAMES)

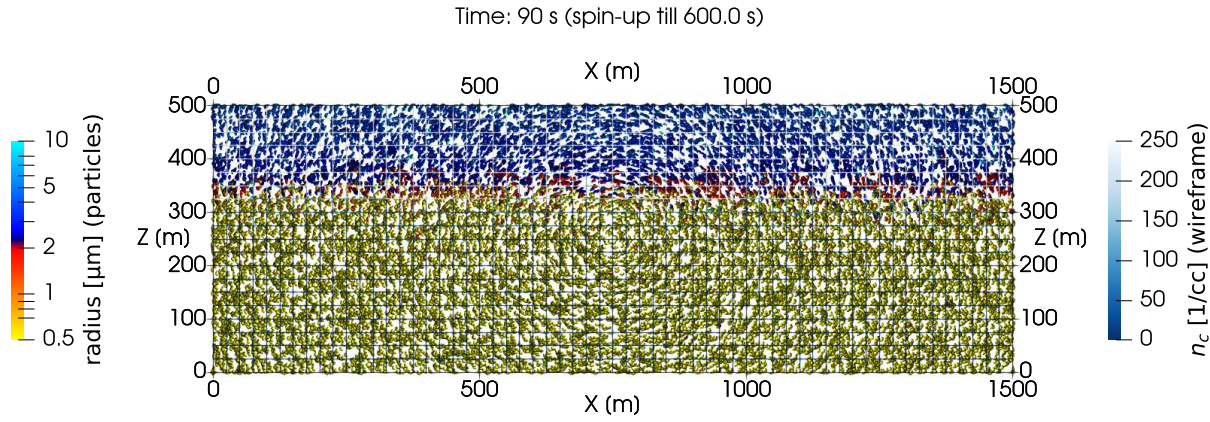


16+16 super-particles/cell for INP-rich + INP-free particles

$N_{\text{aer}} = 300/cc$ (two-mode lognormal) $N_{\text{INP}} = 150/L$ (lognormal, $D_g = 0.74$ μm, $\sigma_g = 2.55$)

spin-up = freezing off; subsequently frozen particles act as tracers

immersion freezing in clouds (Arabas et al. 2025, JAMES)

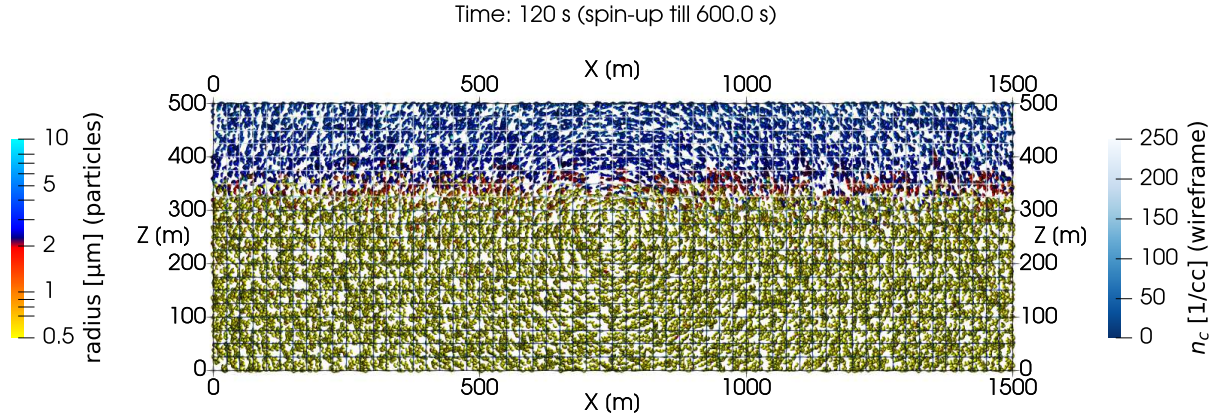


16+16 super-particles/cell for INP-rich + INP-free particles

$N_{\text{aer}} = 300/cc$ (two-mode lognormal) $N_{\text{INP}} = 150/L$ (lognormal, $D_g = 0.74$ μm, $\sigma_g = 2.55$)

spin-up = freezing off; subsequently frozen particles act as tracers

immersion freezing in clouds (Arabas et al. 2025, JAMES)

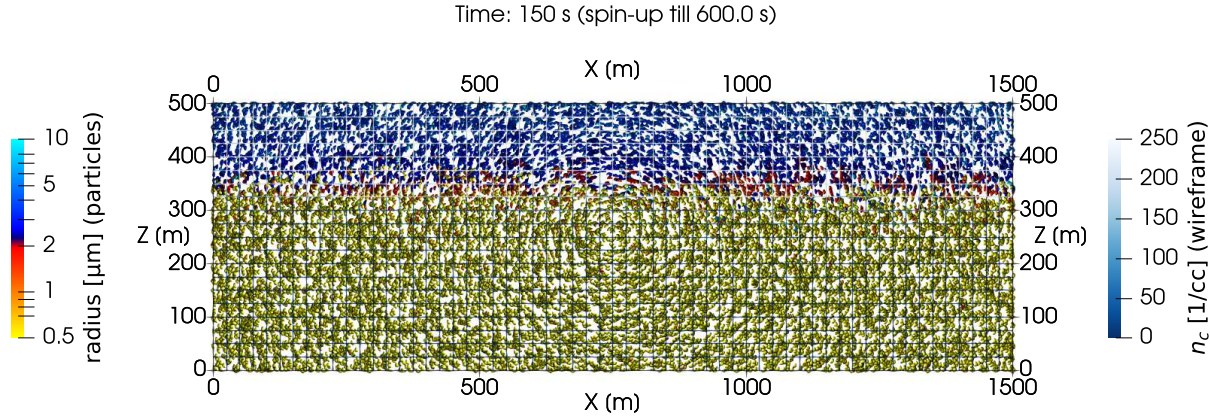


16+16 super-particles/cell for INP-rich + INP-free particles

$N_{\text{aer}} = 300/cc$ (two-mode lognormal) $N_{\text{INP}} = 150/L$ (lognormal, $D_g = 0.74$ μm, $\sigma_g = 2.55$)

spin-up = freezing off; subsequently frozen particles act as tracers

immersion freezing in clouds (Arabas et al. 2025, JAMES)

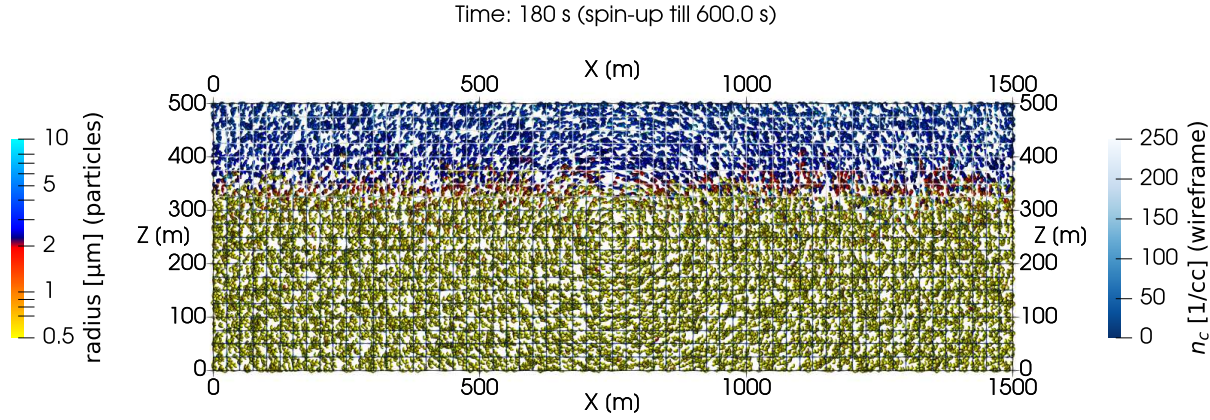


16+16 super-particles/cell for INP-rich + INP-free particles

$N_{\text{aer}} = 300/cc$ (two-mode lognormal) $N_{\text{INP}} = 150/L$ (lognormal, $D_g = 0.74$ μm, $\sigma_g = 2.55$)

spin-up = freezing off; subsequently frozen particles act as tracers

immersion freezing in clouds (Arabas et al. 2025, JAMES)

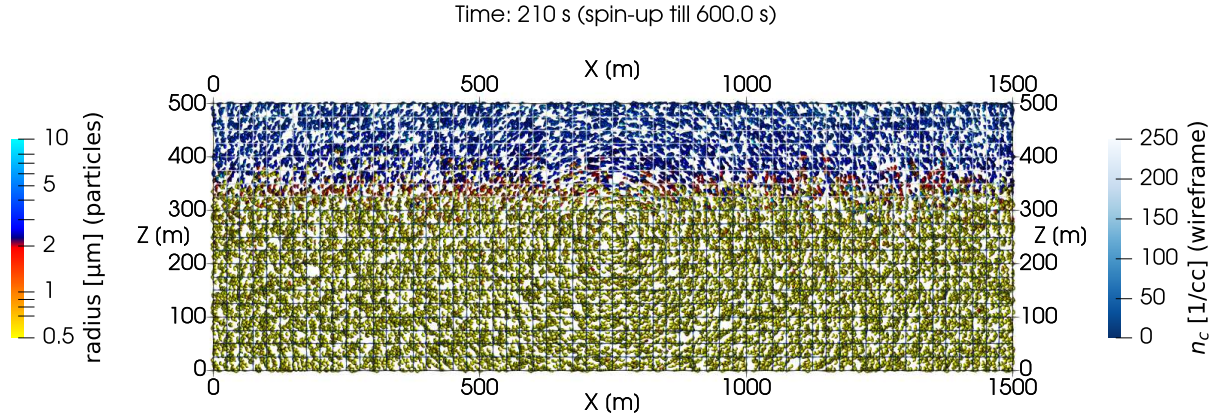


16+16 super-particles/cell for INP-rich + INP-free particles

$N_{\text{aer}} = 300/cc$ (two-mode lognormal) $N_{\text{INP}} = 150/L$ (lognormal, $D_g = 0.74$ μm, $\sigma_g = 2.55$)

spin-up = freezing off; subsequently frozen particles act as tracers

immersion freezing in clouds (Arabas et al. 2025, JAMES)

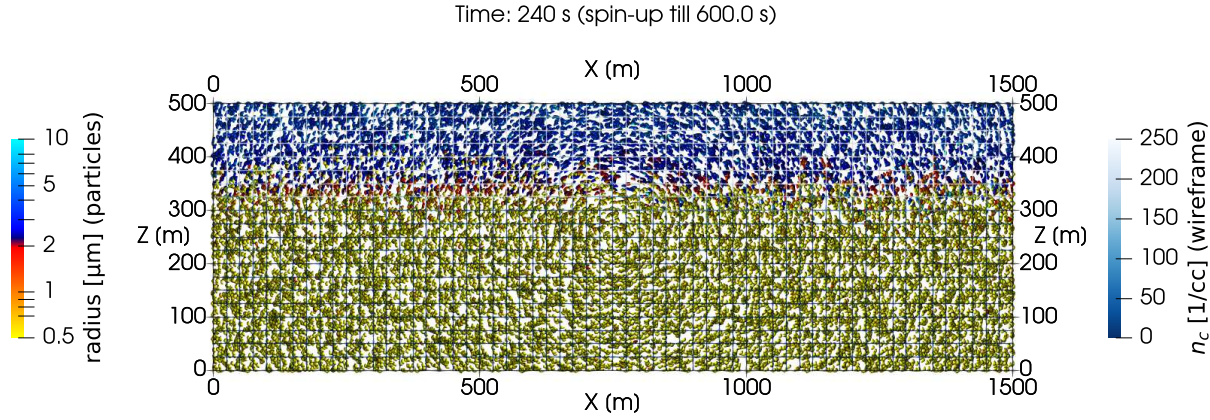


16+16 super-particles/cell for INP-rich + INP-free particles

$N_{\text{aer}} = 300/cc$ (two-mode lognormal) $N_{\text{INP}} = 150/L$ (lognormal, $D_g = 0.74$ μm, $\sigma_g = 2.55$)

spin-up = freezing off; subsequently frozen particles act as tracers

immersion freezing in clouds (Arabas et al. 2025, JAMES)

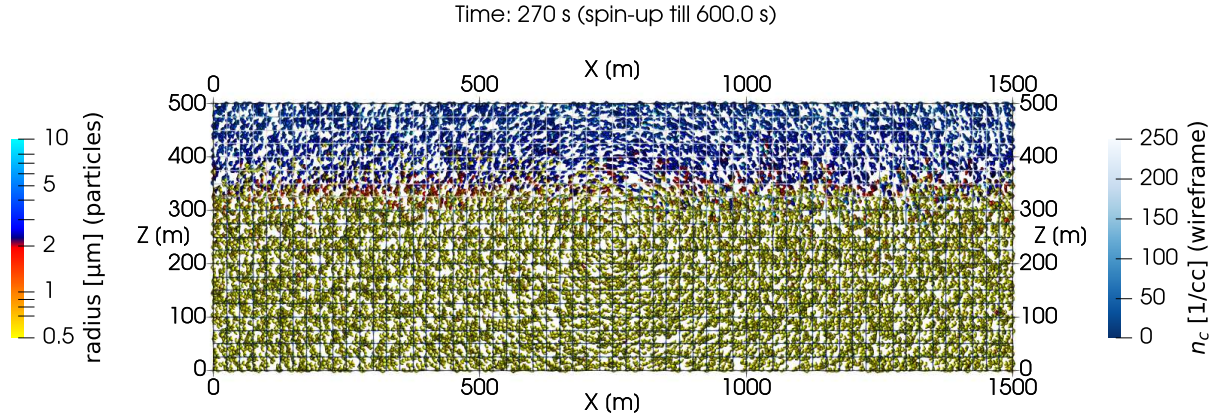


16+16 super-particles/cell for INP-rich + INP-free particles

$N_{\text{aer}} = 300/cc$ (two-mode lognormal) $N_{\text{INP}} = 150/L$ (lognormal, $D_g = 0.74$ μm, $\sigma_g = 2.55$)

spin-up = freezing off; subsequently frozen particles act as tracers

immersion freezing in clouds (Arabas et al. 2025, JAMES)

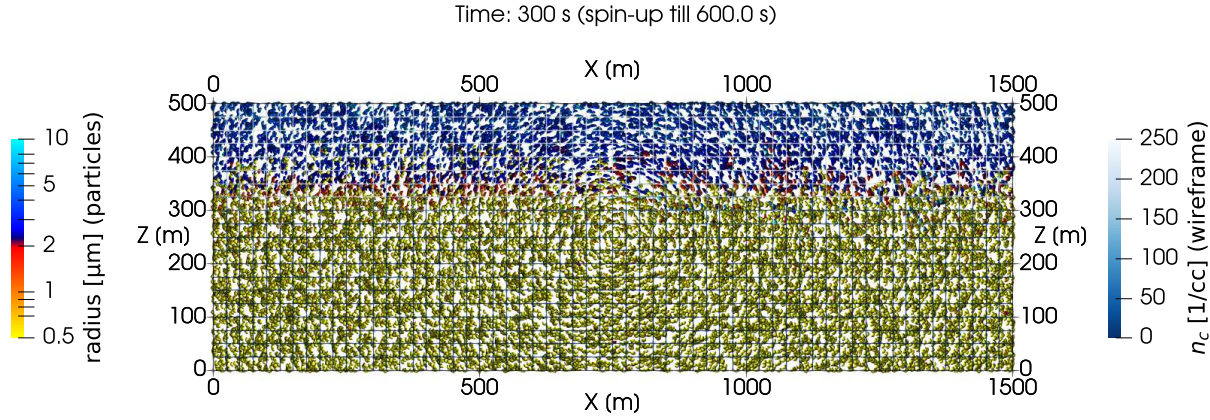


16+16 super-particles/cell for INP-rich + INP-free particles

$N_{\text{aer}} = 300/cc$ (two-mode lognormal) $N_{\text{INP}} = 150/L$ (lognormal, $D_g = 0.74$ μm, $\sigma_g = 2.55$)

spin-up = freezing off; subsequently frozen particles act as tracers

immersion freezing in clouds (Arabas et al. 2025, JAMES)

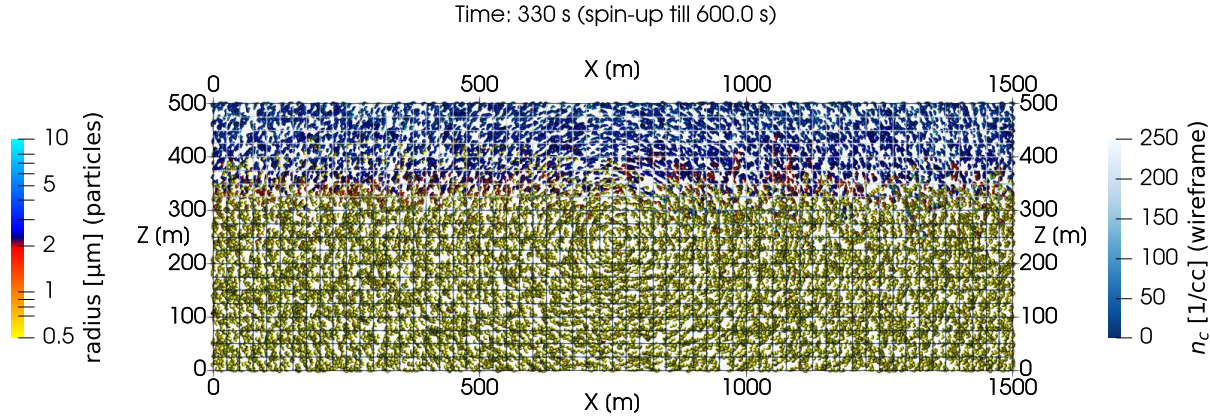


16+16 super-particles/cell for INP-rich + INP-free particles

$N_{\text{aer}} = 300/cc$ (two-mode lognormal) $N_{\text{INP}} = 150/L$ (lognormal, $D_g = 0.74$ μm, $\sigma_g = 2.55$)

spin-up = freezing off; subsequently frozen particles act as tracers

immersion freezing in clouds (Arabas et al. 2025, JAMES)

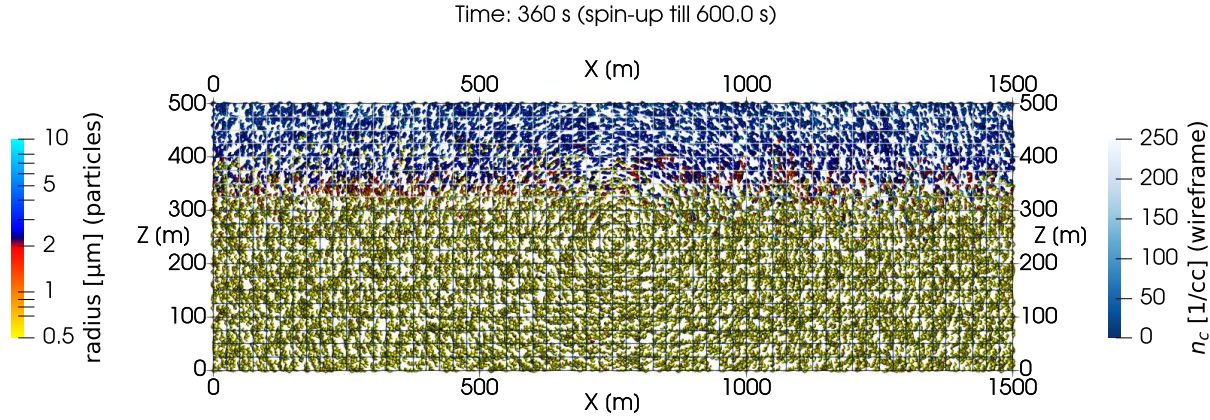


16+16 super-particles/cell for INP-rich + INP-free particles

$N_{\text{aer}} = 300/cc$ (two-mode lognormal) $N_{\text{INP}} = 150/L$ (lognormal, $D_g = 0.74$ μm, $\sigma_g = 2.55$)

spin-up = freezing off; subsequently frozen particles act as tracers

immersion freezing in clouds (Arabas et al. 2025, JAMES)

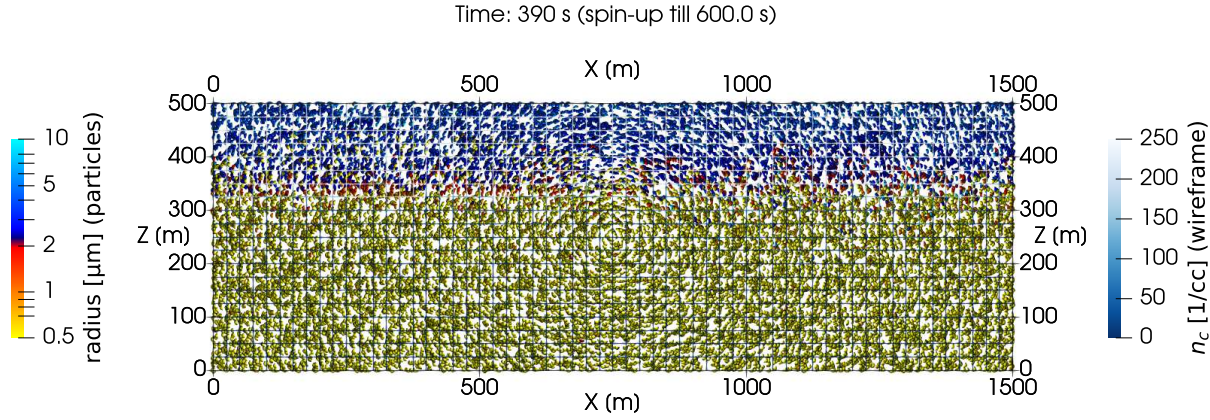


16+16 super-particles/cell for INP-rich + INP-free particles

$N_{\text{aer}} = 300/cc$ (two-mode lognormal) $N_{\text{INP}} = 150/L$ (lognormal, $D_g = 0.74$ μm, $\sigma_g = 2.55$)

spin-up = freezing off; subsequently frozen particles act as tracers

immersion freezing in clouds (Arabas et al. 2025, JAMES)

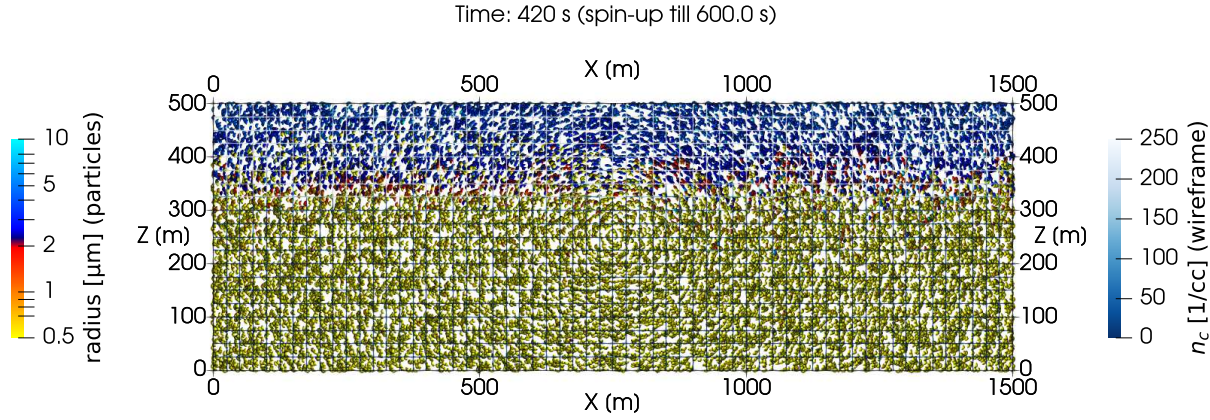


16+16 super-particles/cell for INP-rich + INP-free particles

$N_{\text{aer}} = 300/cc$ (two-mode lognormal) $N_{\text{INP}} = 150/L$ (lognormal, $D_g = 0.74$ μm, $\sigma_g = 2.55$)

spin-up = freezing off; subsequently frozen particles act as tracers

immersion freezing in clouds (Arabas et al. 2025, JAMES)

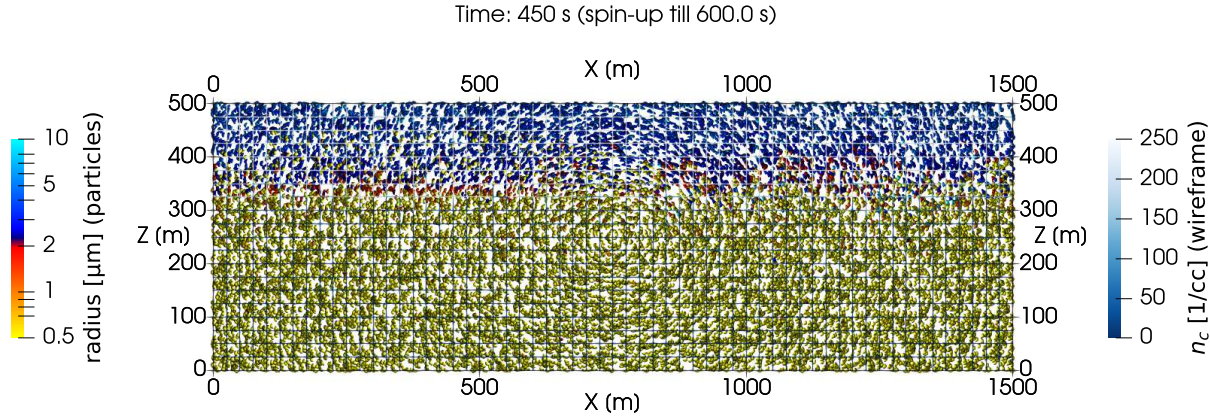


16+16 super-particles/cell for INP-rich + INP-free particles

$N_{\text{aer}} = 300/cc$ (two-mode lognormal) $N_{\text{INP}} = 150/L$ (lognormal, $D_g = 0.74$ μm, $\sigma_g = 2.55$)

spin-up = freezing off; subsequently frozen particles act as tracers

immersion freezing in clouds (Arabas et al. 2025, JAMES)

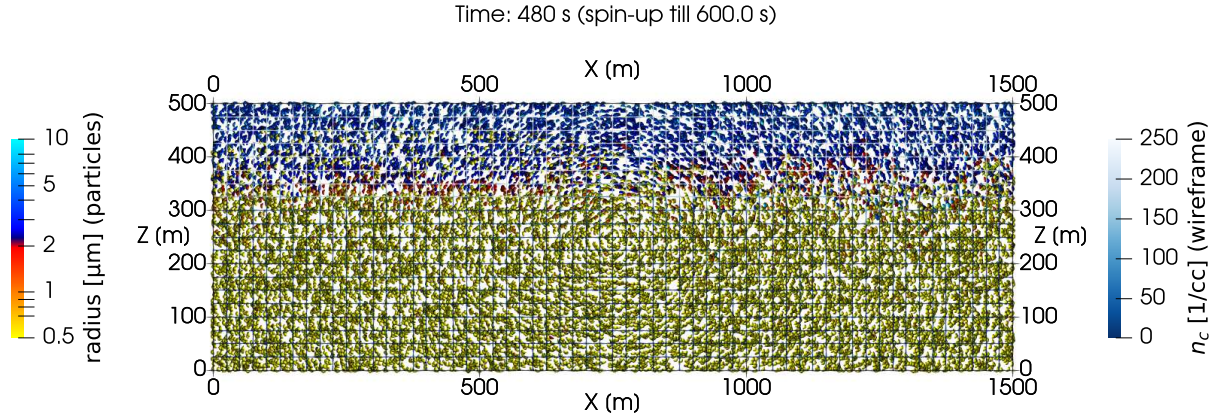


16+16 super-particles/cell for INP-rich + INP-free particles

$N_{\text{aer}} = 300/cc$ (two-mode lognormal) $N_{\text{INP}} = 150/L$ (lognormal, $D_g = 0.74$ μm, $\sigma_g = 2.55$)

spin-up = freezing off; subsequently frozen particles act as tracers

immersion freezing in clouds (Arabas et al. 2025, JAMES)

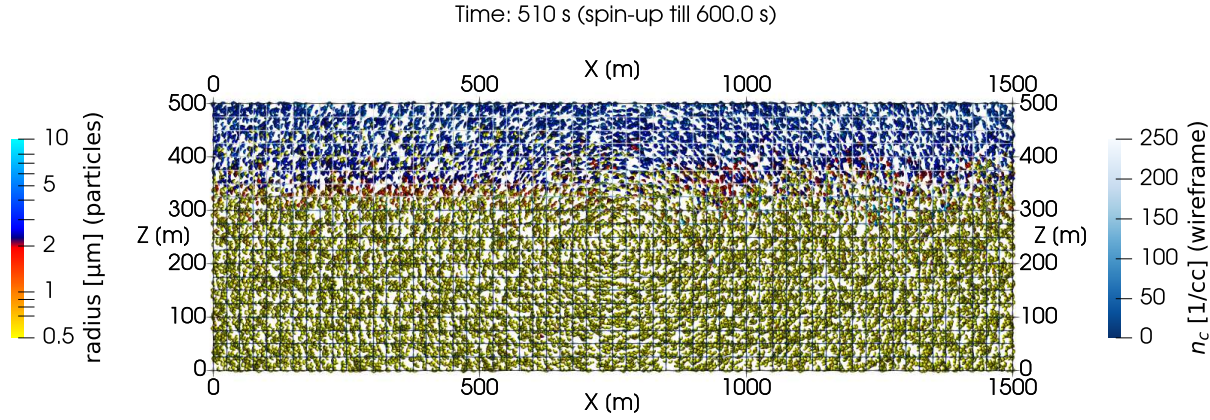


16+16 super-particles/cell for INP-rich + INP-free particles

$N_{\text{aer}} = 300/cc$ (two-mode lognormal) $N_{\text{INP}} = 150/L$ (lognormal, $D_g = 0.74$ μm, $\sigma_g = 2.55$)

spin-up = freezing off; subsequently frozen particles act as tracers

immersion freezing in clouds (Arabas et al. 2025, JAMES)

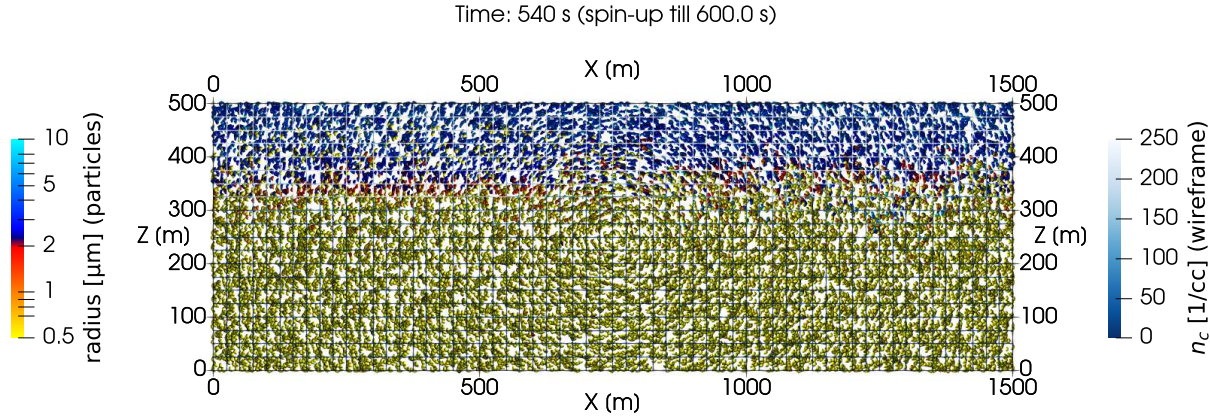


16+16 super-particles/cell for INP-rich + INP-free particles

$N_{\text{aer}} = 300/cc$ (two-mode lognormal) $N_{\text{INP}} = 150/L$ (lognormal, $D_g = 0.74$ μm, $\sigma_g = 2.55$)

spin-up = freezing off; subsequently frozen particles act as tracers

immersion freezing in clouds (Arabas et al. 2025, JAMES)

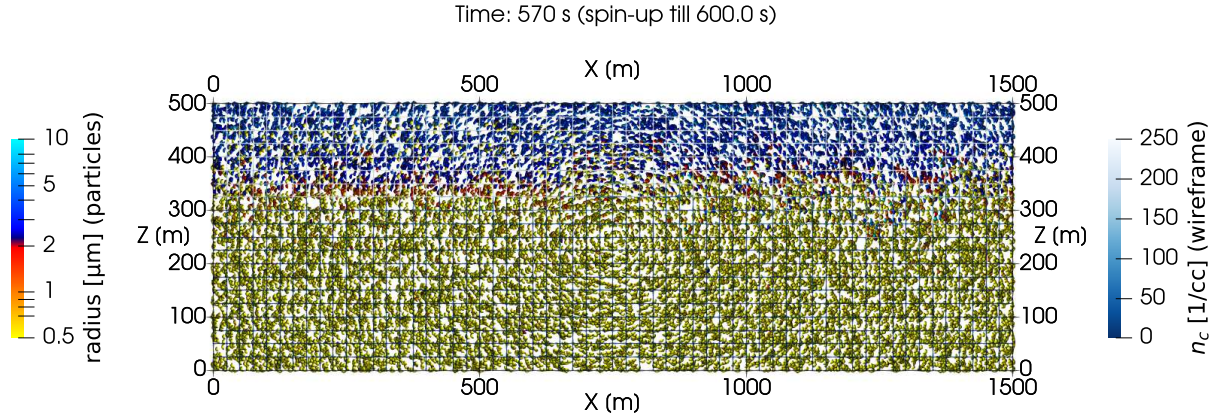


16+16 super-particles/cell for INP-rich + INP-free particles

$N_{\text{aer}} = 300/cc$ (two-mode lognormal) $N_{\text{INP}} = 150/L$ (lognormal, $D_g = 0.74$ μm, $\sigma_g = 2.55$)

spin-up = freezing off; subsequently frozen particles act as tracers

immersion freezing in clouds (Arabas et al. 2025, JAMES)

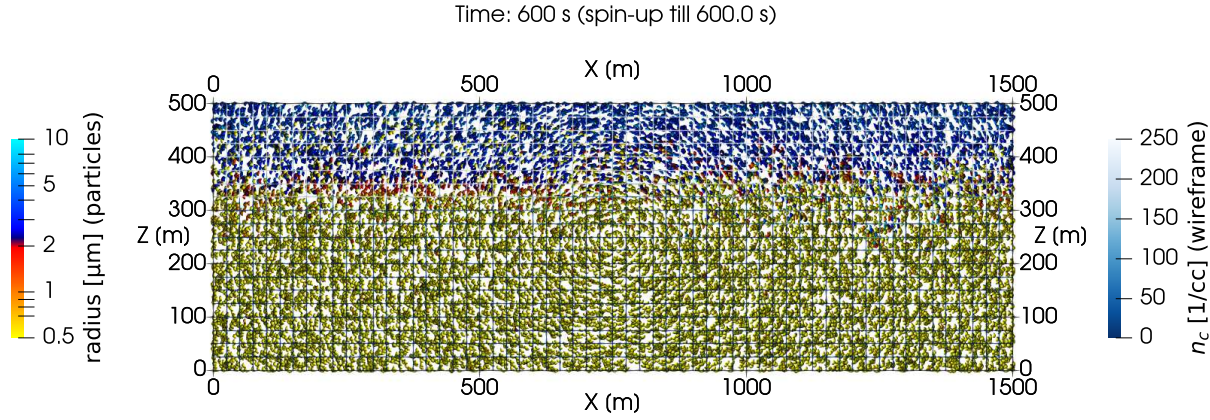


16+16 super-particles/cell for INP-rich + INP-free particles

$N_{\text{aer}} = 300/cc$ (two-mode lognormal) $N_{\text{INP}} = 150/L$ (lognormal, $D_g = 0.74$ μm, $\sigma_g = 2.55$)

spin-up = freezing off; subsequently frozen particles act as tracers

immersion freezing in clouds (Arabas et al. 2025, JAMES)

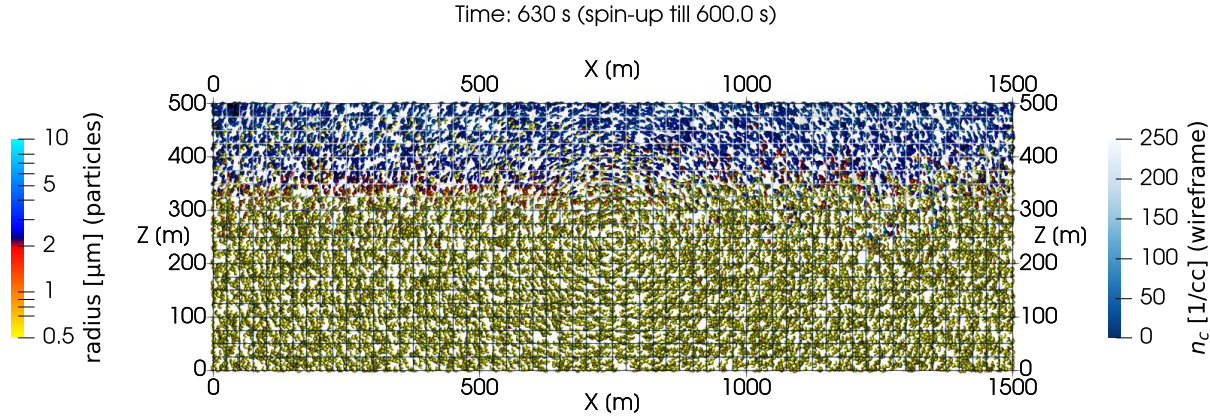


16+16 super-particles/cell for INP-rich + INP-free particles

$N_{\text{aer}} = 300/cc$ (two-mode lognormal) $N_{\text{INP}} = 150/L$ (lognormal, $D_g = 0.74$ μm, $\sigma_g = 2.55$)

spin-up = freezing off; subsequently frozen particles act as tracers

immersion freezing in clouds (Arabas et al. 2025, JAMES)

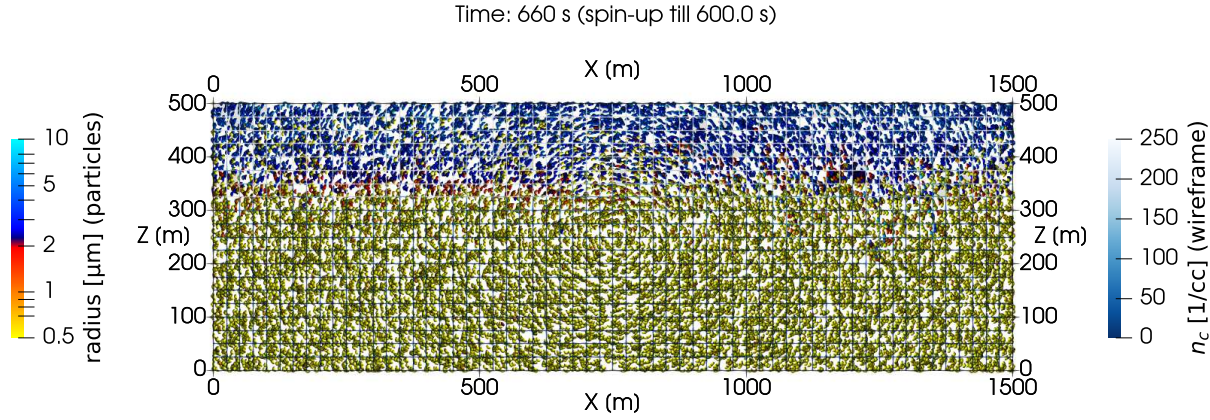


16+16 super-particles/cell for INP-rich + INP-free particles

$N_{\text{aer}} = 300/cc$ (two-mode lognormal) $N_{\text{INP}} = 150/L$ (lognormal, $D_g = 0.74$ μm, $\sigma_g = 2.55$)

spin-up = freezing off; subsequently frozen particles act as tracers

immersion freezing in clouds (Arabas et al. 2025, JAMES)

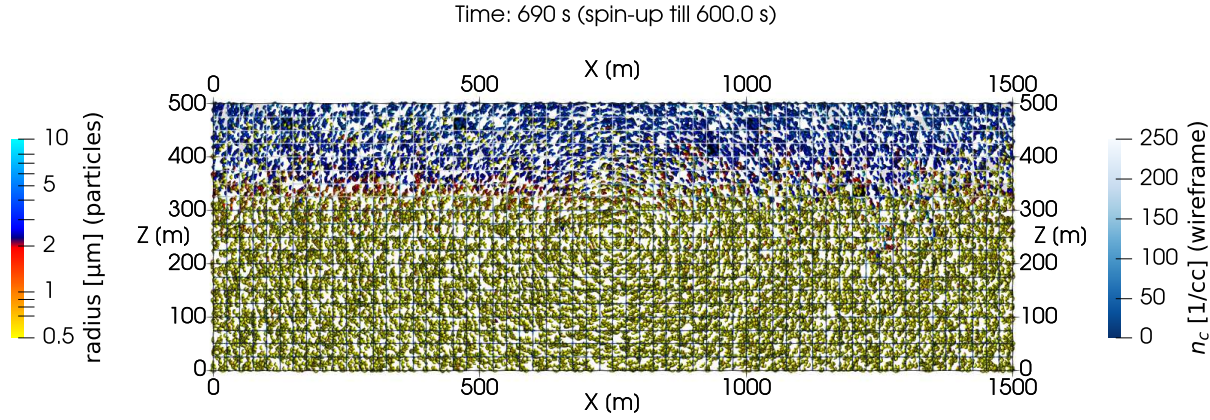


16+16 super-particles/cell for INP-rich + INP-free particles

$N_{\text{aer}} = 300/cc$ (two-mode lognormal) $N_{\text{INP}} = 150/L$ (lognormal, $D_g = 0.74$ μm, $\sigma_g = 2.55$)

spin-up = freezing off; subsequently frozen particles act as tracers

immersion freezing in clouds (Arabas et al. 2025, JAMES)

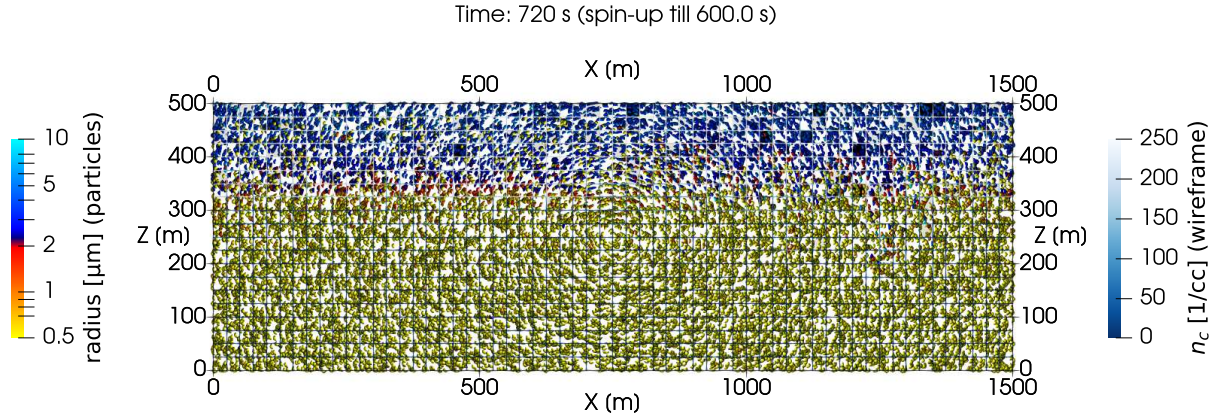


16+16 super-particles/cell for INP-rich + INP-free particles

$N_{\text{aer}} = 300/cc$ (two-mode lognormal) $N_{\text{INP}} = 150/L$ (lognormal, $D_g = 0.74$ μm, $\sigma_g = 2.55$)

spin-up = freezing off; subsequently frozen particles act as tracers

immersion freezing in clouds (Arabas et al. 2025, JAMES)

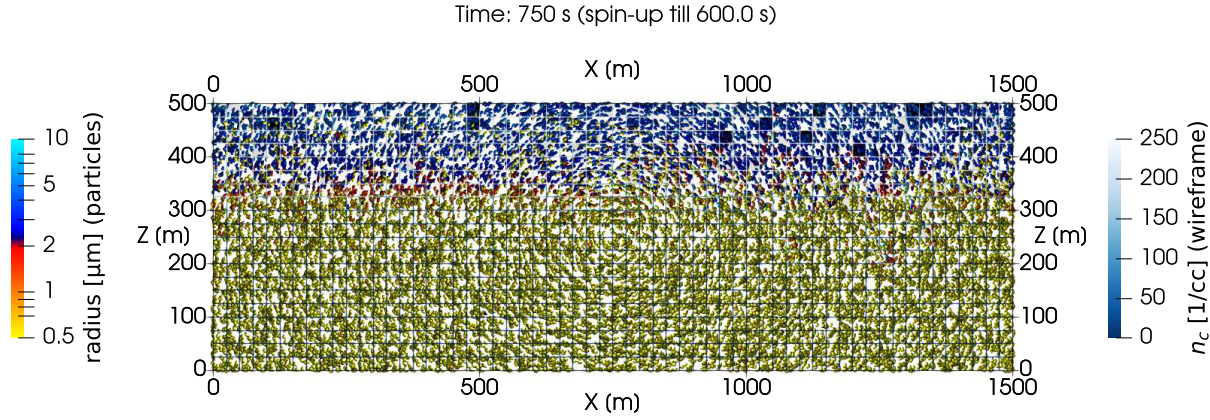


16+16 super-particles/cell for INP-rich + INP-free particles

$N_{\text{aer}} = 300/cc$ (two-mode lognormal) $N_{\text{INP}} = 150/L$ (lognormal, $D_g = 0.74$ μm, $\sigma_g = 2.55$)

spin-up = freezing off; subsequently frozen particles act as tracers

immersion freezing in clouds (Arabas et al. 2025, JAMES)

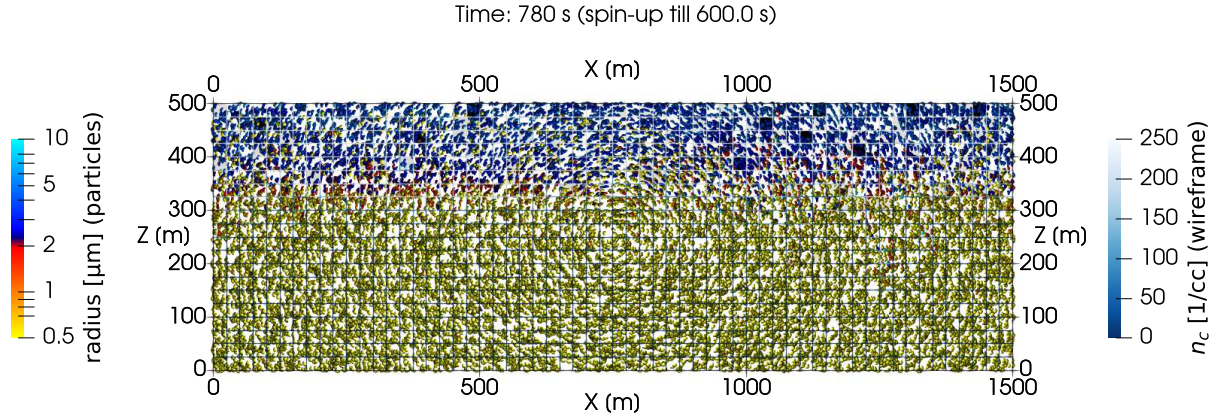


16+16 super-particles/cell for INP-rich + INP-free particles

$N_{\text{aer}} = 300/cc$ (two-mode lognormal) $N_{\text{INP}} = 150/L$ (lognormal, $D_g = 0.74$ μm, $\sigma_g = 2.55$)

spin-up = freezing off; subsequently frozen particles act as tracers

immersion freezing in clouds (Arabas et al. 2025, JAMES)

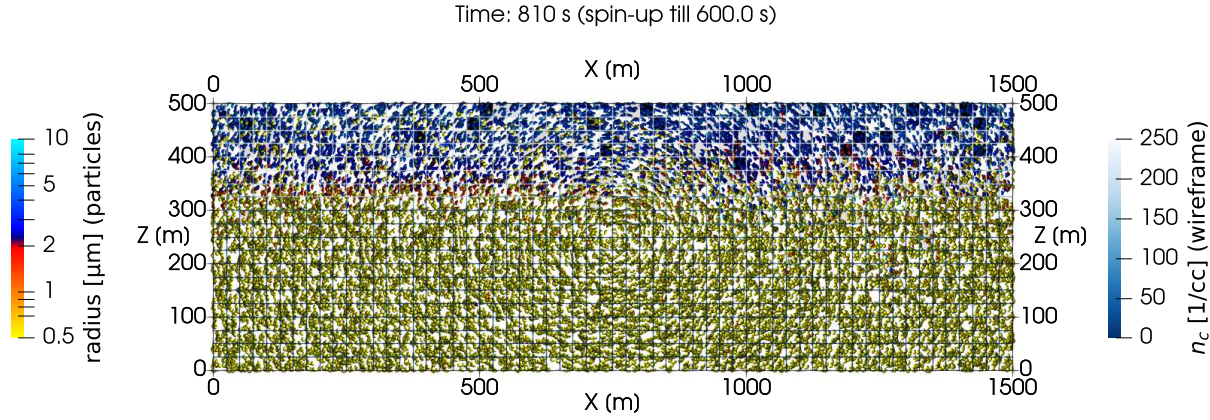


16+16 super-particles/cell for INP-rich + INP-free particles

$N_{\text{aer}} = 300/cc$ (two-mode lognormal) $N_{\text{INP}} = 150/L$ (lognormal, $D_g = 0.74$ μm, $\sigma_g = 2.55$)

spin-up = freezing off; subsequently frozen particles act as tracers

immersion freezing in clouds (Arabas et al. 2025, JAMES)

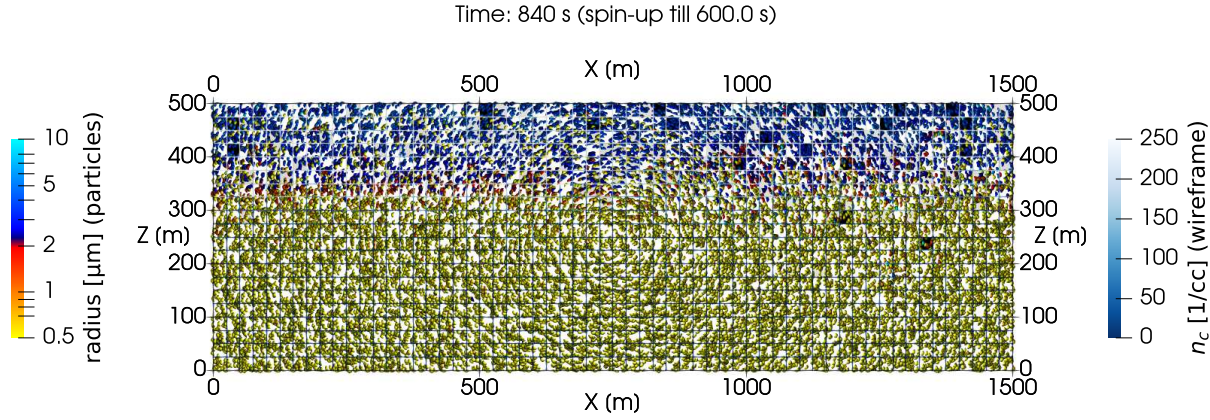


16+16 super-particles/cell for INP-rich + INP-free particles

$N_{\text{aer}} = 300/cc$ (two-mode lognormal) $N_{\text{INP}} = 150/L$ (lognormal, $D_g = 0.74$ μm, $\sigma_g = 2.55$)

spin-up = freezing off; subsequently frozen particles act as tracers

immersion freezing in clouds (Arabas et al. 2025, JAMES)

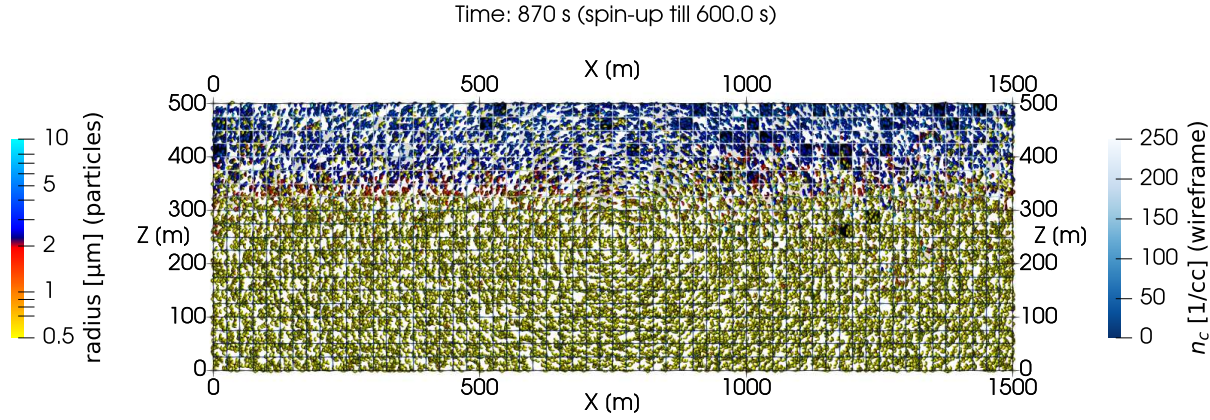


16+16 super-particles/cell for INP-rich + INP-free particles

$N_{\text{aer}} = 300/cc$ (two-mode lognormal) $N_{\text{INP}} = 150/L$ (lognormal, $D_g = 0.74$ μm, $\sigma_g = 2.55$)

spin-up = freezing off; subsequently frozen particles act as tracers

immersion freezing in clouds (Arabas et al. 2025, JAMES)

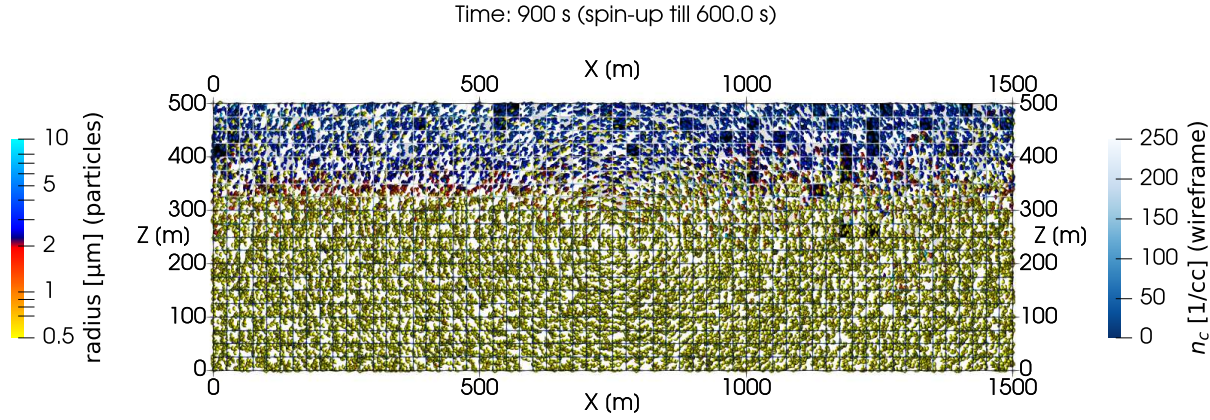


16+16 super-particles/cell for INP-rich + INP-free particles

$N_{\text{aer}} = 300/cc$ (two-mode lognormal) $N_{\text{INP}} = 150/L$ (lognormal, $D_g = 0.74$ μm, $\sigma_g = 2.55$)

spin-up = freezing off; subsequently frozen particles act as tracers

immersion freezing in clouds (Arabas et al. 2025, JAMES)

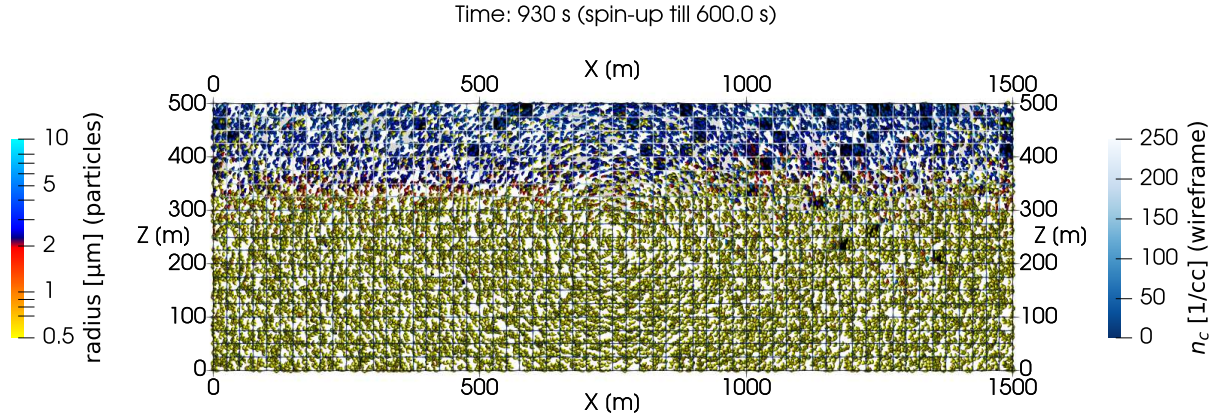


16+16 super-particles/cell for INP-rich + INP-free particles

$N_{\text{aer}} = 300/cc$ (two-mode lognormal) $N_{\text{INP}} = 150/L$ (lognormal, $D_g = 0.74$ μm, $\sigma_g = 2.55$)

spin-up = freezing off; subsequently frozen particles act as tracers

immersion freezing in clouds (Arabas et al. 2025, JAMES)

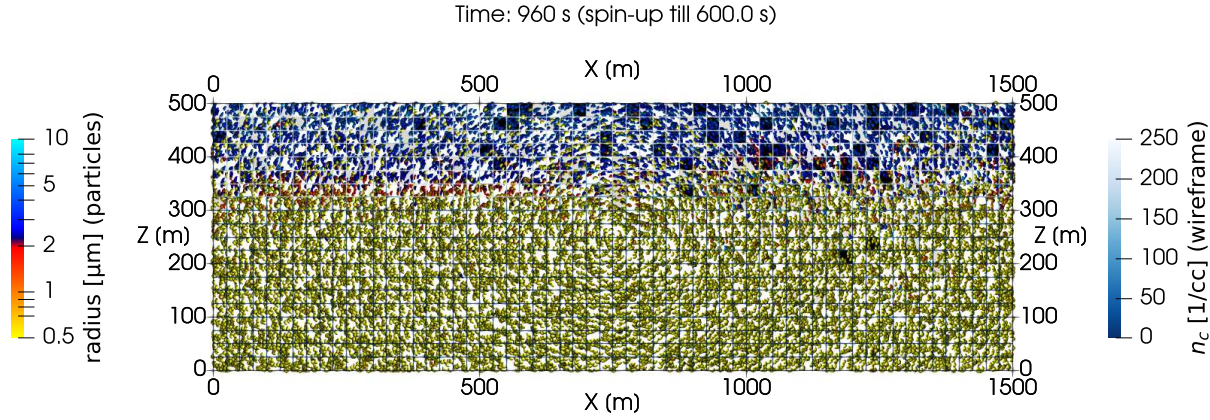


16+16 super-particles/cell for INP-rich + INP-free particles

$N_{\text{aer}} = 300/cc$ (two-mode lognormal) $N_{\text{INP}} = 150/L$ (lognormal, $D_g = 0.74$ μm, $\sigma_g = 2.55$)

spin-up = freezing off; subsequently frozen particles act as tracers

immersion freezing in clouds (Arabas et al. 2025, JAMES)

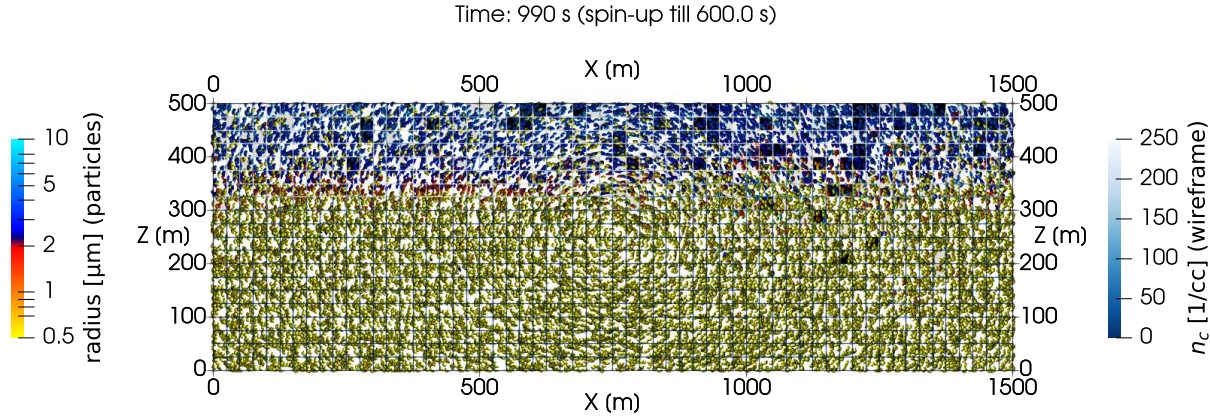


16+16 super-particles/cell for INP-rich + INP-free particles

$N_{\text{aer}} = 300/cc$ (two-mode lognormal) $N_{\text{INP}} = 150/L$ (lognormal, $D_g = 0.74$ μm, $\sigma_g = 2.55$)

spin-up = freezing off; subsequently frozen particles act as tracers

immersion freezing in clouds (Arabas et al. 2025, JAMES)

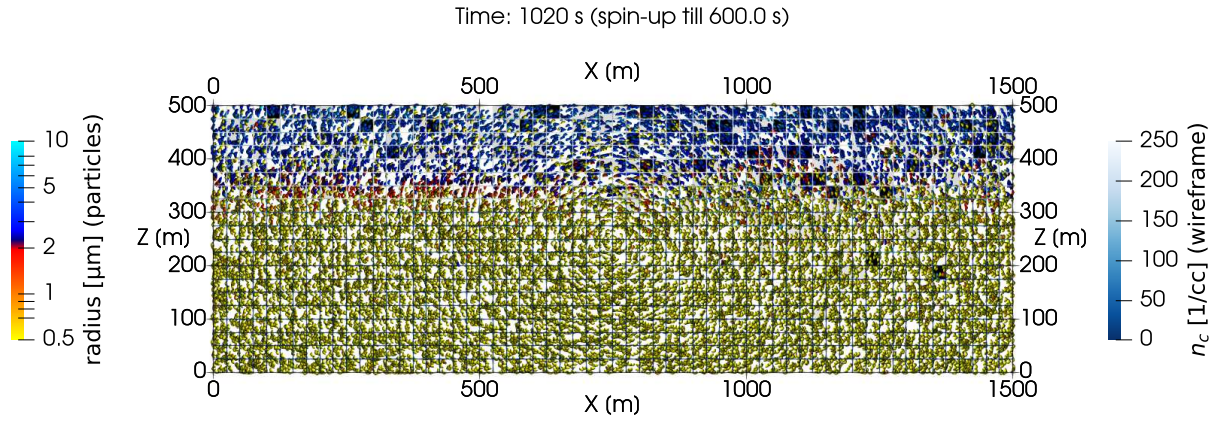


16+16 super-particles/cell for INP-rich + INP-free particles

$N_{\text{aer}} = 300/cc$ (two-mode lognormal) $N_{\text{INP}} = 150/L$ (lognormal, $D_g = 0.74$ μm, $\sigma_g = 2.55$)

spin-up = freezing off; subsequently frozen particles act as tracers

immersion freezing in clouds (Arabas et al. 2025, JAMES)

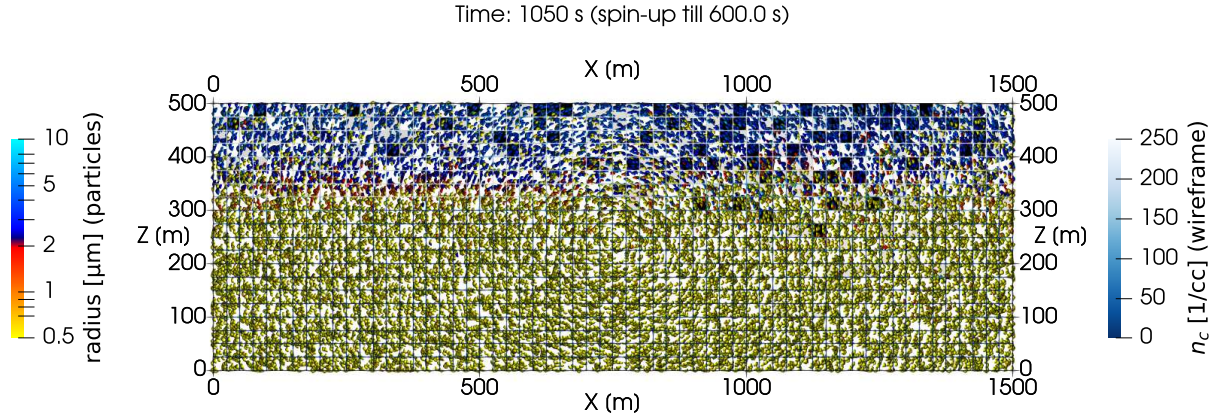


16+16 super-particles/cell for INP-rich + INP-free particles

$N_{\text{aer}} = 300/cc$ (two-mode lognormal) $N_{\text{INP}} = 150/L$ (lognormal, $D_g = 0.74$ μm, $\sigma_g = 2.55$)

spin-up = freezing off; subsequently frozen particles act as tracers

immersion freezing in clouds (Arabas et al. 2025, JAMES)

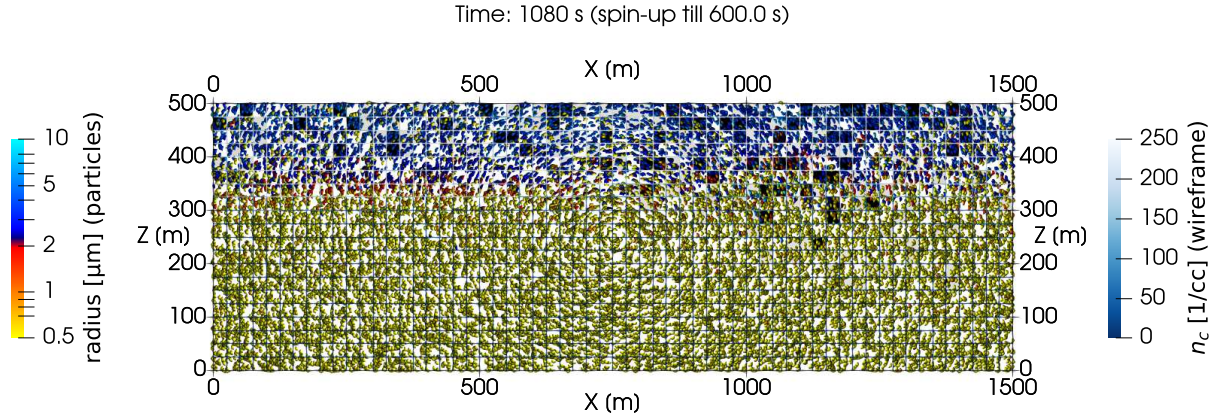


16+16 super-particles/cell for INP-rich + INP-free particles

$N_{\text{aer}} = 300/cc$ (two-mode lognormal) $N_{\text{INP}} = 150/L$ (lognormal, $D_g = 0.74$ μm, $\sigma_g = 2.55$)

spin-up = freezing off; subsequently frozen particles act as tracers

immersion freezing in clouds (Arabas et al. 2025, JAMES)

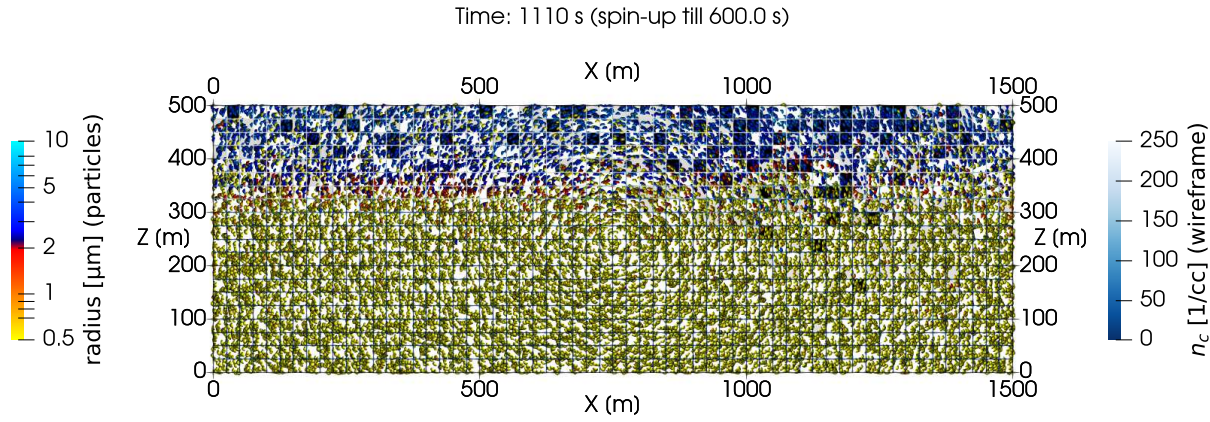


16+16 super-particles/cell for INP-rich + INP-free particles

$N_{\text{aer}} = 300/cc$ (two-mode lognormal) $N_{\text{INP}} = 150/L$ (lognormal, $D_g = 0.74$ μm, $\sigma_g = 2.55$)

spin-up = freezing off; subsequently frozen particles act as tracers

immersion freezing in clouds (Arabas et al. 2025, JAMES)

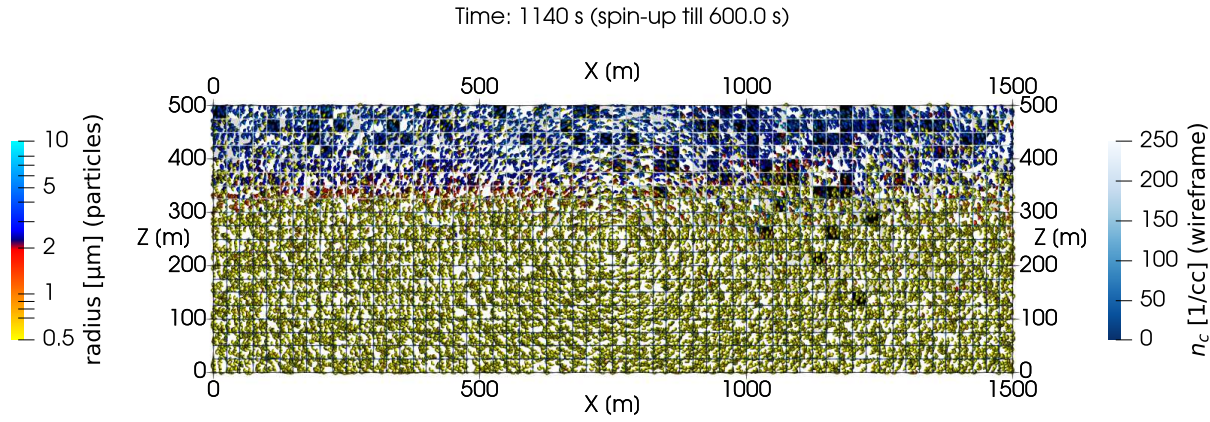


16+16 super-particles/cell for INP-rich + INP-free particles

$N_{\text{aer}} = 300/cc$ (two-mode lognormal) $N_{\text{INP}} = 150/L$ (lognormal, $D_g = 0.74$ μm, $\sigma_g = 2.55$)

spin-up = freezing off; subsequently frozen particles act as tracers

immersion freezing in clouds (Arabas et al. 2025, JAMES)

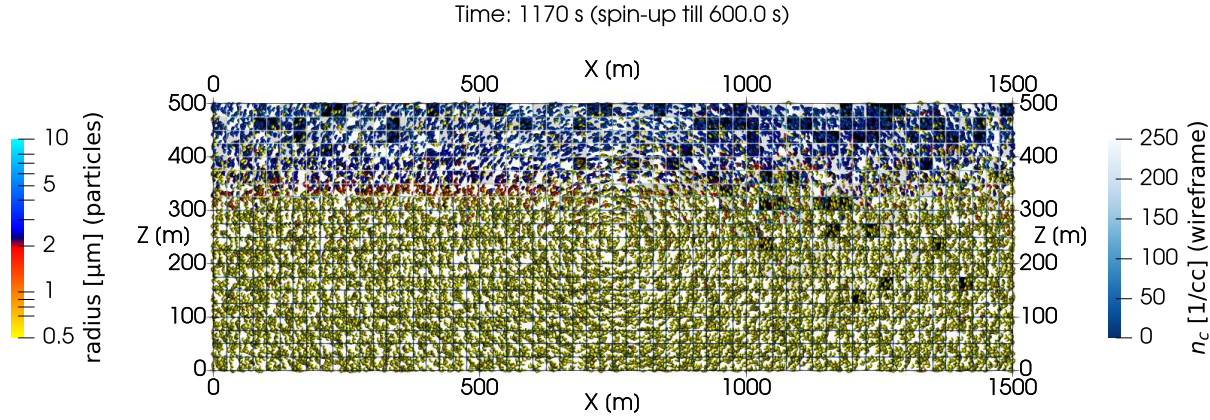


16+16 super-particles/cell for INP-rich + INP-free particles

$N_{\text{aer}} = 300/cc$ (two-mode lognormal) $N_{\text{INP}} = 150/L$ (lognormal, $D_g = 0.74$ μm, $\sigma_g = 2.55$)

spin-up = freezing off; subsequently frozen particles act as tracers

immersion freezing in clouds (Arabas et al. 2025, JAMES)

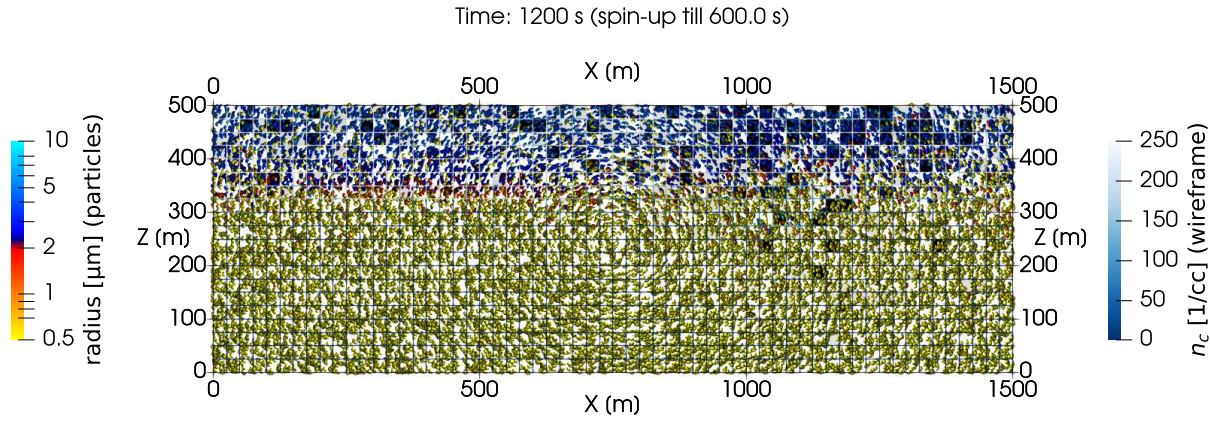


16+16 super-particles/cell for INP-rich + INP-free particles

$N_{\text{aer}} = 300/cc$ (two-mode lognormal) $N_{\text{INP}} = 150/L$ (lognormal, $D_g = 0.74$ μm, $\sigma_g = 2.55$)

spin-up = freezing off; subsequently frozen particles act as tracers

immersion freezing in clouds (Arabas et al. 2025, JAMES)



16+16 super-particles/cell for INP-rich + INP-free particles

$N_{\text{aer}} = 300/cc$ (two-mode lognormal) $N_{\text{INP}} = 150/L$ (lognormal, $D_g = 0.74$ μm, $\sigma_g = 2.55$)

spin-up = freezing off; subsequently frozen particles act as tracers

MPDATA for Asian option pricing using 2D PDE (Magnuszewski et al. 2025)

arXiv > q-fin > arXiv:2505.24435

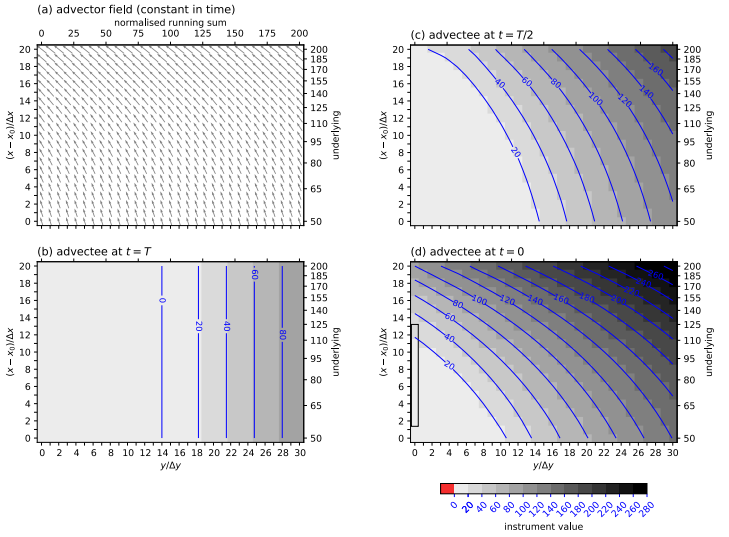
Quantitative Finance > Computational Finance

[Submitted on 30 May 2025]

Path-dependent option pricing with two-dimensional PDE using MPDATA

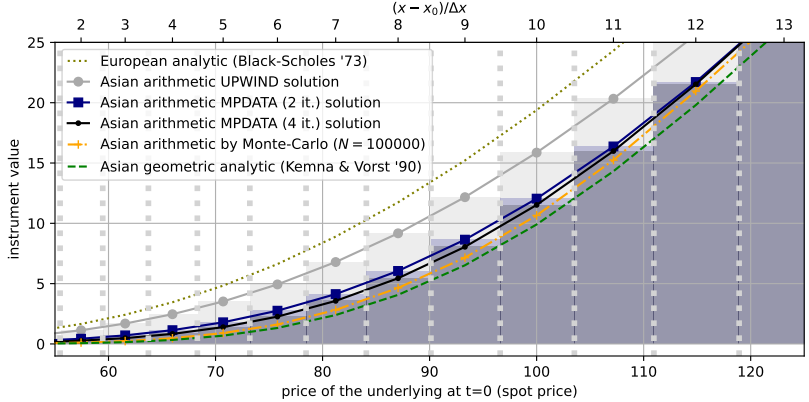
Paweł Magnuszewski, Sylwester Arabas

In this paper, we discuss a simple yet robust PDE method for evaluating path-dependent Asian-style options using the non-oscillatory forward-in-time second-order MPDATA finite-difference scheme. The valuation methodology involves casting the Black-Merton-Scholes equation as a transport problem by first transforming it into a homogeneous advection-diffusion PDE via variable substitution, and then expressing the diffusion term as an advective flux using the pseudo-velocity technique.



MPDATA for Asian option pricing using 2D PDE (Magnuszewski et al. 2025)

$$\frac{\partial f}{\partial t} + rS\frac{\partial f}{\partial S} + \frac{\sigma^2}{2}S^2\frac{\partial^2 f}{\partial S^2} + v\frac{\partial f}{\partial A} - rf = 0$$



arXiv:2505.24435

- **PyMPDATA performance vs. C++**

PyMPDATA v1.0: Bartman et al. 2022 (JOSS)

The Journal of Open Source Software

DOI: 10.21105/joss.03896

Software

- Review
- Repository
- Archive

Editor: Arfon Smith

Reviewers:

- @Chiil
- @wdeconinck

Submitted: 25 October 2021
Published: 05 September 2022

PyMPDATA v1: Numba-accelerated implementation of MPDATA with examples in Python, Julia and Matlab

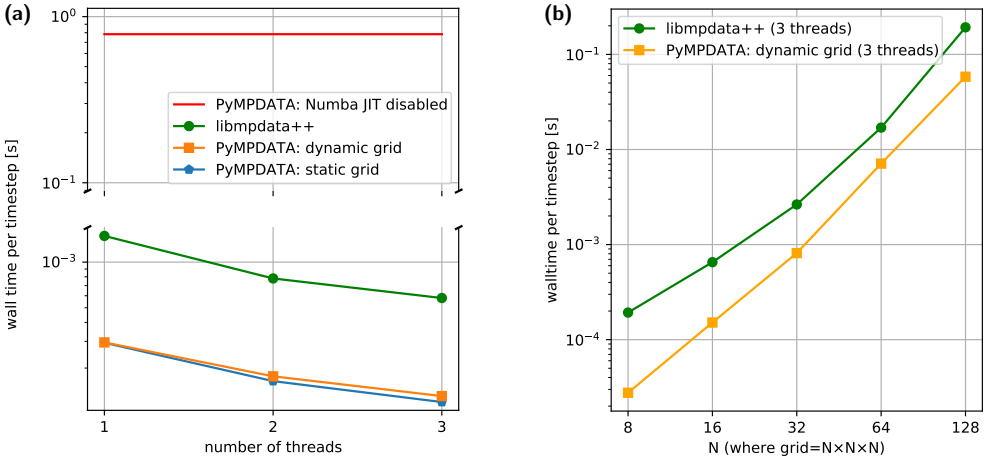
Piotr Bartman [1], **Jakub Banaśkiewicz**[1], **Szymon Drenda**[1], **Maciej Manna**[1], **Michael A. Olesik** [1], **Paweł Rozwoda**[1], **Michał Sadowski** [1], **and Sylwester Arabas** [1,2]

1 Jagiellonian University, Kraków, Poland **2** University of Illinois at Urbana-Champaign, IL, USA

Statement of need

Convection-diffusion problems arise across a wide range of pure and applied research, in particular in geosciences, aerospace engineering, and financial modelling (for an overview of applications, see, e.g., section 1.1 in Morton (1996)). One of the key challenges in numerical solutions of problems involving advective transport is sign preservation of the advected field (for an overview of this and other aspects of numerical solutions to advection problems, see, e.g., Røed (2019)). The Multidimensional Positive Definite Advection Transport Algorithm (MPDATA) is a robust, explicit-in-time, and sign-preserving solver introduced in Smolarkiewicz (1983) and Smolarkiewicz (1984). MPDATA has been subsequently developed into a family of numerical schemes with numerous variants and solution procedures addressing a diverse set of problems in geophysical fluid dynamics and beyond. For reviews of MPDATA applications and variants, see, e.g., Smolarkiewicz & Margolin (1998) and Smolarkiewicz (2006).

Numba JIT & multi-threading: PyMPDATA vs. libmpdata++ performance



Bartman et al. 2022 (JOSS, doi:10.21105/joss.03896)

- **MPI, HPC & distributed-memory parallelism**

introducing Numba-MPI (now a dependency of py-pde)

ScienceDirect

SoftwareX

Volume 28, December 2024, 101897

Original software publication

Numba-MPI v1.0: Enabling MPI communication within Numba/LLVM JIT-compiled Python code

Kacper Derlatka [a 1], Maciej Manna [a 2], Oleksii Bulenok [a 3], David Zwicker [b], Sylwester Arabas [c]

[a] Faculty of Mathematics and Computer Science, Jagiellonian University in Kraków, Poland

[b] Max Planck Institute for Dynamics and Self-Organization, Göttingen, Germany

[c] Faculty of Physics and Applied Computer Science, AGH University of Krakow, Poland

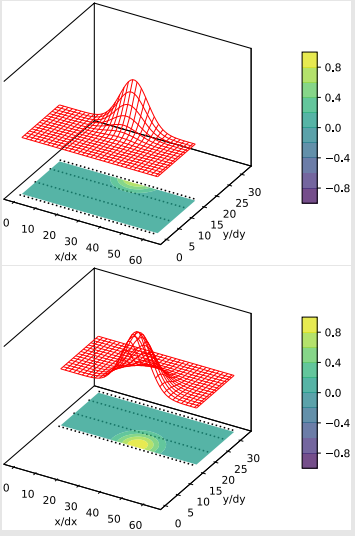
https://doi.org/10.1016/j.softx.2024.101897

Abstract

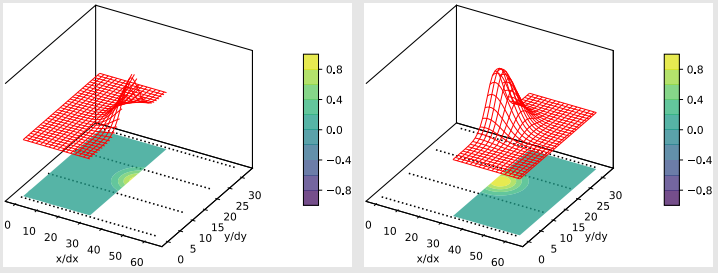
The numba-mpi package offers access to the Message Passing Interface (MPI) routines from Python code that uses the Numba just-in-time (JIT) compiler. As a result, high-performance and multi-threaded Python code may utilize MPI communication facilities without leaving the JIT-compiled code blocks, which is not possible with the mpi4py package, a higher-level Python interface to MPI. For debugging or code-coverage analysis purposes, numba-mpi retains full functionality of the code even if the JIT compilation is disabled.

PyMPDATA-MPI: customisable hybrid threading + MPI parallelisation

threading dim = MPI dim

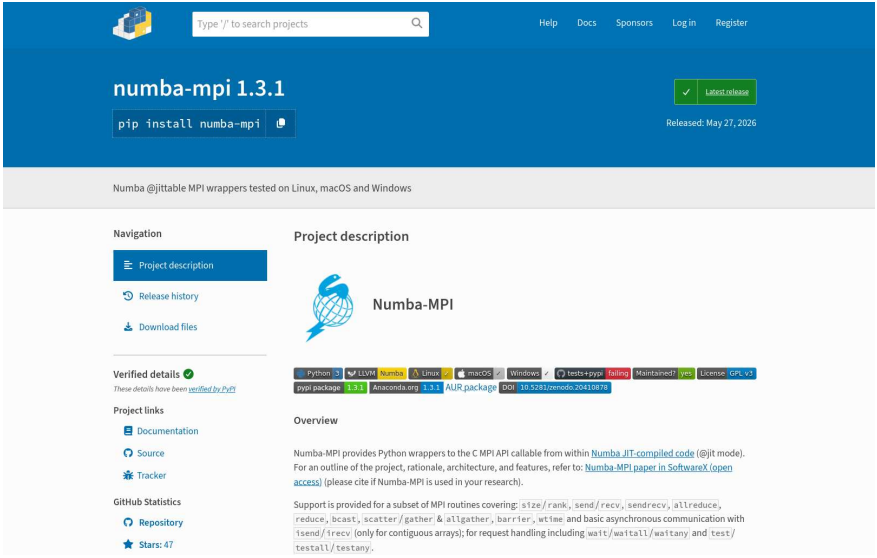


threading dimension ≠ MPI dimension



Derlatka et al. 2024 (SoftwareX, doi:10.1016/j.softx.2024.101897)

Numba-MPI


Type '/' to search projects
Help
Docs
Sponsors
Log in
Register
numba-mpi 1.3.1
pip install numba-mpi
Latest release
Released: May 27, 2026
Numba @jittable MPI wrappers tested on Linux, macOS and Windows
Navigation
Project description
Release history
Download files
Verified details
These details have been verified by PyPI
Project links
Documentation
Source
Tracker
GitHub Statistics
Repository
Stars: 47
Project description
Numba-MPI
Python
LLVM
Numba
Linux
macOS
Windows
tests+pypi
failing
Maintained?
yes
License
GPL v3
pypi package
1.3.1
Anaconda.org
1.3.1
AUR package
DOI
10.5281/zenodo.20410878
Overview
Numba-MPI provides Python wrappers to the C MPI API callable from within Numba JIT-compiled code (@jit mode). For an outline of the project, rationale, architecture, and features, refer to: Numba-MPI paper in SoftwareX (open access) (please cite if Numba-MPI is used in your research).
Support is provided for a subset of MPI routines covering: size/rank, send/recv, sendrecv, allreduce, reduce, bcast, scatter/gather & allgather, barrier, wtime and basic asynchronous communication with isend/irecv (only for contiguous arrays); for request handling including wait/waitall/waitany and test/testall/testany.

pypi.org/p/PyMPDATA

pypi.org/p/Numba-MPI

Thanks to all (>20 students) code contributors!

Thank you for your attention!

sylwester.arabas@agh.edu.pl