

PyMPDATA: A Just-in-Time Compiled Implementation of MPDATA

Sylwester Arabas

AGH University of Krakow

June 26 2025 (M3ODEL Lunch Talk @uni-mainz.de)





- zfs.agh.edu.pl

AGH Wissenschaftlich-Technische Universität

Die AGH Wissenschaftlich-Technische Universität^[4] (polnisch: Akademia Górnictwa-Hutnicza im. Stanisława Staszica w Krakowie = Akademie für Bergbau und Hüttenwesen „Stanisław Staszic“ zu Krakau) – kurz AGH – ist die größte technische Universität in der Stadt Krakau mit 23.570 Studenten und 2.154 wissenschaftlichen Mitarbeitern.^[2] Sie hat den Ruf einer der besten Universitäten in Polen.^[5]

Die AGH Wissenschaftlich-Technische Universität wurde 1919 gegründet und das Hauptgebäude wurde in den Jahren 1923–1935 in Czarna Wieś errichtet. Rektor der Universität ist seit Mai 2020 Jerzy Lis.^{[6][7]}

Fakultäten

Die AGH Universität gliedert sich in 16 Fakultäten^[7]:

- Fakultät für Bauingenieurwesen und Ressourcenmanagement (*Wydział Inżynierii Lądowej i Gospodarki Zasobami*)
- Fakultät für Metalltechnik und Industrieinformatik (*Wydział Inżynierii Metali i Informatyki Przemysłowej*)
- Fakultät für Elektrotechnik, Automatik, Informatik und Biomedizintechnik (*Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej*)
- Fakultät für Informatik, Elektronik und Telekommunikation (*Wydział Informatyki, Elektroniki i Telekomunikacji*)
- Fakultät für Maschinenbau und Robotik (*Wydział Inżynierii Mechanicznej i Robotyki*)
- Fakultät für Geologie, Geophysik und Umweltschutz (*Wydział Geologii, Geofizyki i Ochrony Środowiska*)
- Fakultät für montane Geodäsie und Umwelt ingenieurwesen (*Wydział Geodezji Górniczej i Inżynierii Środowiska*)
- Fakultät für Material- und Keramikwissenschaften (*Wydział Inżynierii Materiałowej i Ceramiki*)
- Fakultät für Gießereitechnik (*Wydział Odlewnictwa*)
- Fakultät für Nichteisenmetalle (*Wydział Metali Nieżelaznych*)
- Fakultät für Bohr-, Öl- und Gaswesen (*Wydział Wiertnictwa, Nafty i Gazu*)
- Fakultät für Management (*Wydział Zarządzania*)
- Fakultät für Energie und Kraftstoffe (*Wydział Energetyki i Paliw*)
- Fakultät für Physik und Angewandte Informatik (*Wydział Fizyki i Informatyki Stosowanej*)
- Fakultät für angewandte Mathematik (*Wydział Matematyki Stosowanej*)
- Fakultät für Geisteswissenschaften (*Wydział Humanistyczny*)

AGH Wissenschaftlich-Technische Universität

Akademia Górnictwa-Hutnicza im. Stanisława Staszica w Krakowie



Motto

„Labore crea, labore et scientiae servis“

Gründung

1919

Trägerschaft

staatlich

Ort

Krakau, Polen

Rектор

Jerzy Lis

Studierende

18,074^[1] (12/2023)

Mitarbeiter

4,167^[2] (31. Dezember 2017)

davon

157^[2] (31. Dezember 2017)

Professoren

IAU^[3]

Netzwerke

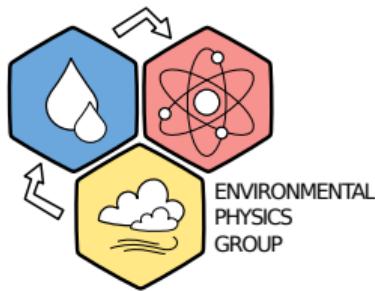
www.agh.edu.pl (https://www.agh.edu.pl)

Website

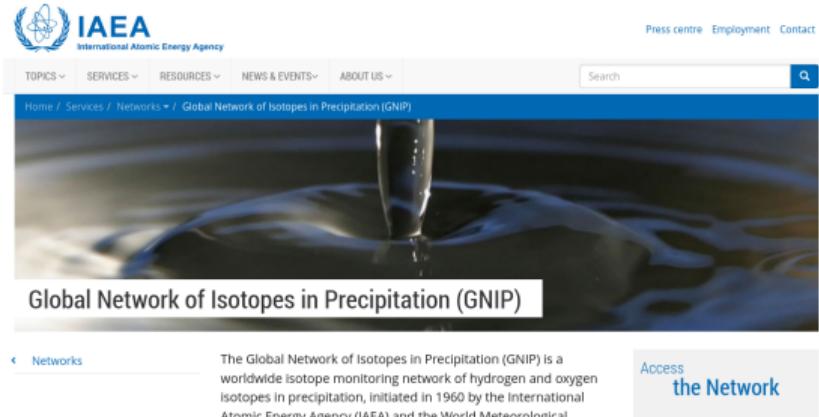
www.agh.edu.pl

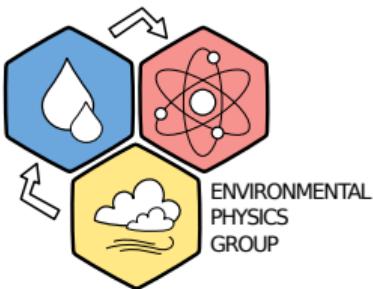


Hauptgebäude der AGH

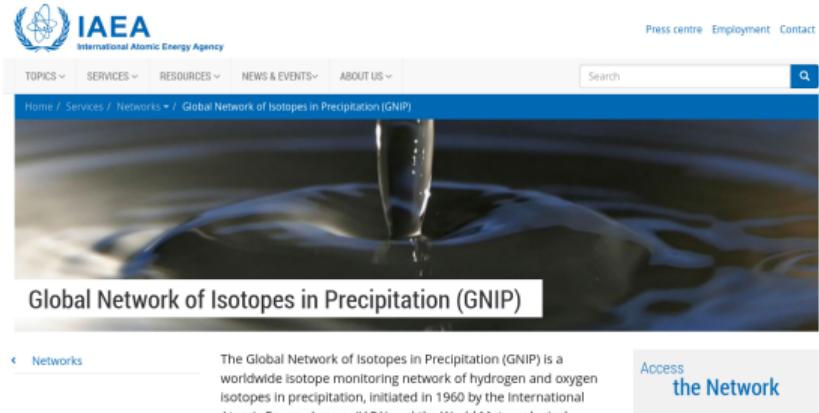


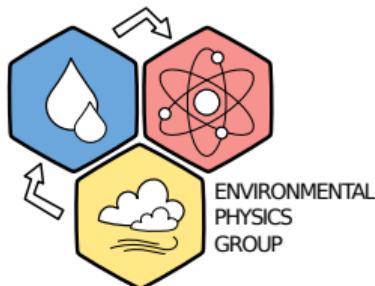
- zfs.agh.edu.pl
- IAEA/GNIP site in Kraków

A screenshot of the IAEA's Global Network of Isotopes in Precipitation (GNIP) website. At the top right are links for 'Press centre', 'Employment', and 'Contact'. Below that is a navigation bar with 'TOPICS', 'SERVICES', 'RESOURCES', 'NEWS & EVENTS', and 'ABOUT US'. A search bar is on the far right. The main content area features a large image of a water droplet falling into water. A banner at the top says 'Home / Services / Networks / Global Network of Isotopes in Precipitation (GNIP)'. Below it, a section titled 'Networks' contains text about the GNIP and a button labeled 'Access the Network'.



- zfs.agh.edu.pl
- IAEA/GNIP site in Kraków
- 50-year precip isotopic data record

A screenshot of the IAEA's Global Network of Isotopes in Precipitation (GNIP) website. At the top, the IAEA logo and name are displayed, along with links for Press centre, Employment, and Contact. A navigation bar below includes TOPICS, SERVICES, RESOURCES, NEWS & EVENTS, and ABOUT US. A search bar is also present. The main content area features a large image of a water droplet falling into water, with the text 'Global Network of Isotopes in Precipitation (GNIP)' overlaid. Below this, a section titled 'Networks' provides information about the GNIP, stating it is a worldwide isotope monitoring network of hydrogen and oxygen isotopes in precipitation, initiated in 1960 by the International Atomic Energy Agency (IAEA) and the World Meteorological Organization (WMO), and operates in cooperation with numerous partner institutions in Member States. A button labeled 'Access the Network' is visible on the right.



- zfs.agh.edu.pl
- IAEA/GNIP site in Kraków
- 50-year precip isotopic data record
- high-altitude lab (clouds in-situ)
@Kasprowy Wierch (6500 ft AMSL)



photo: naukaoklimacie.pl

plan of the talk

- MPDATA scheme and its implementations
- PyMPDATA: pure-Python just-in-time compiled MPDATA
- PyMPDATA "examples" and documentation
- PyMPDATA performance vs. C++
- MPI, HPC & distributed-memory parallelisation?
- PyMPDATA in teaching (i.e., implemented by students!)

MPDATA key concepts: UPWIND discretisation

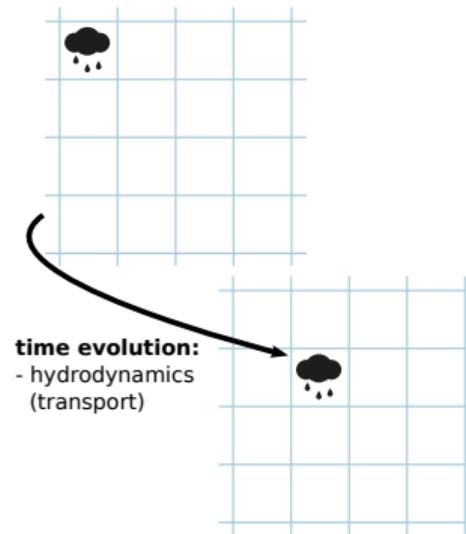
- advection equation / scalar conservation law:

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

$\psi(\mathbf{x}, t)$: advected scalar field (adveecee),

$\mathbf{v} = \{u, \dots\} = G\dot{\mathbf{x}}$: flow velocity vector field (advector),

$G(\mathbf{x})$: fluid density, Jacobian of coordinate transformation, or their product



MPDATA key concepts: UPWIND discretisation

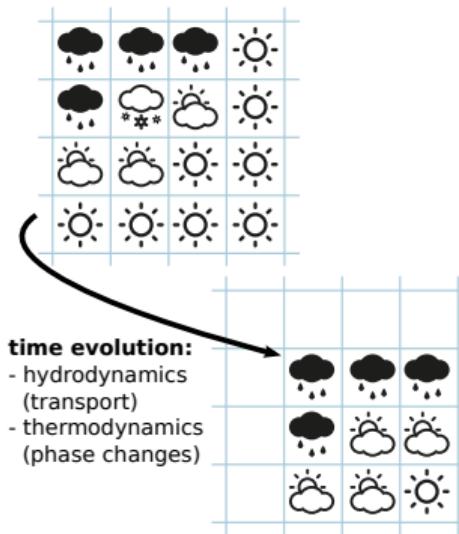
- advection equation / scalar conservation law:

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

$\psi(\mathbf{x}, t)$: advected scalar field (adveecee),

$\mathbf{v} = \{u, \dots\} = G\dot{\mathbf{x}}$: 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),

$\mathbf{v} = \{u, \dots\} = G\dot{\mathbf{x}}$: flow velocity vector field (advector),

$G(\mathbf{x})$: fluid density, Jacobian of coordinate transformation, or their product

- homogeneous problem in 1D and with $G = 1$:

$$\partial_t \psi + \partial_x(u\psi) = 0$$

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),

$\mathbf{v} = \{u, \dots\} = G\dot{\mathbf{x}}$: flow velocity vector field (advector),

$G(\mathbf{x})$: fluid density, Jacobian of coordinate transformation, or their product

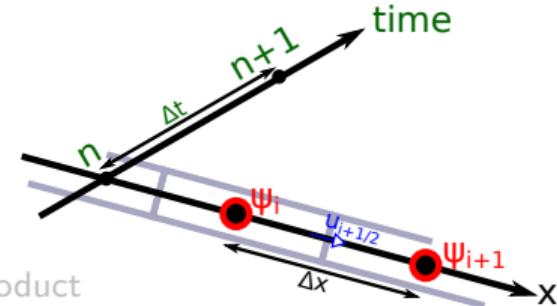
- homogeneous problem in 1D and with $G = 1$:

$$\partial_t \psi + \partial_x(u\psi) = 0$$

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

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

$$f(\psi_l, \psi_r, u) = \underbrace{\frac{u + |u|}{2}}_{\text{positive part}} \psi_l + \underbrace{\frac{u - |u|}{2}}_{\text{negative part}} \psi_r$$



MPDATA key concepts: Courant number & UPWIND stability criterion

- introducing non-dimensional Courant number $C = u \frac{\Delta t}{\Delta x}$:

$$\psi_i^{n+1} = \psi_i^n - [f(\psi_i^n, \psi_{i+1}^n, C_{i+1/2}^n) - f(\psi_{i-1}^n, \psi_i^n, C_{i-1/2}^n)]$$

yields a conservative and sing-preserving “UPWIND” hello-world scheme stable for $|C| \leq 1$.

```
1 def f(psi_l, psi_r, C):
2     return .5 * (C + abs(C)) * psi_l + \
3             .5 * (C - abs(C)) * psi_r
4 def step(psi: np.ndarray, i: slice, C: np.ndarray):
5     psi[i] = psi[i] - (
6         f(psi[i], psi[i + one], C[i + hlf]) -
7         f(psi[i - one], psi[i], C[i - hlf]))
8
9 def upwind(nt: int, C: np.ndarray, psi: np.ndarray):
10    i = slice(1, len(psi) - 1)
11    for _ in range(nt):
12        step(psi, i, C)
```

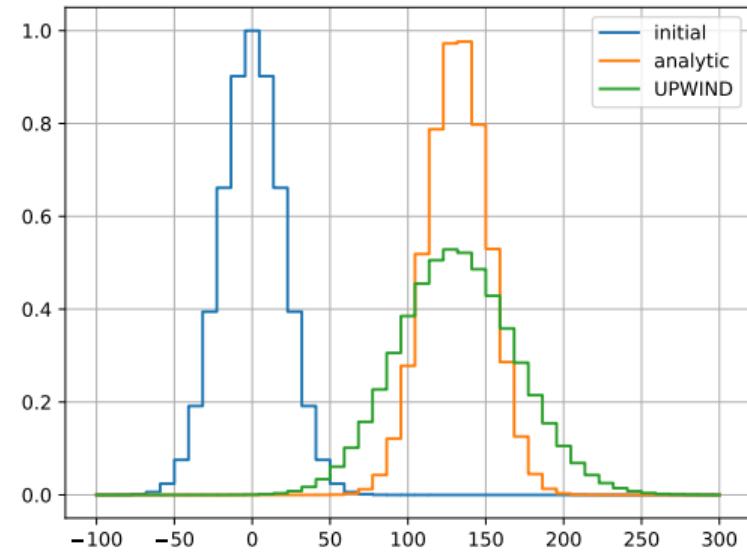
MPDATA key concepts: Courant number & UPWIND stability criterion

- introducing non-dimensional Courant number $C = u \frac{\Delta t}{\Delta x}$:

$$\psi_i^{n+1} = \psi_i^n - [f(\psi_i^n, \psi_{i+1}^n, C_{i+1/2}^n) - f(\psi_{i-1}^n, \psi_i^n, C_{i-1/2}^n)]$$

yields a conservative and sing-preserving “UPWIND” hello-world scheme stable for $|C| \leq 1$.

```
1 def f(psi_l, psi_r, C):
2     return .5 * (C + abs(C)) * psi_l + \
3             .5 * (C - abs(C)) * psi_r
4 def step(psi: np.ndarray, i: slice, C: np.ndarray):
5     psi[i] = psi[i] - (
6         f(psi[i       ], psi[i + one], C[i + hlf]) -
7         f(psi[i - one], psi[i       ], C[i - hlf]))
8
9 def upwind(nt: int, C: np.ndarray, psi: np.ndarray):
10    i = slice(1, len(psi) - 1)
11    for _ in range(nt):
12        step(psi, i, C)
```



MPDATA key concepts: numerical diffusion & modified-equation analysis

- UPWIND incurs **numerical diffusion**, quantifiable using Taylor expansion (const. C for simplicity):

$$\psi_i^{n+1} = \psi_i^n + \partial_t \psi|_i^n (+\Delta t) + \frac{1}{2} \partial_t^2 \psi|_i^n (+\Delta t)^2 + O(\Delta t^3)$$

$$\psi_{i+1}^n = \psi_i^n + \partial_x \psi|_i^n (+\Delta x) + \frac{1}{2} \partial_x^2 \psi|_i^n (+\Delta x)^2 + O(\Delta x^3)$$

$$\psi_{i-1}^n = \psi_i^n + \partial_x \psi|_i^n (-\Delta x) + \frac{1}{2} \partial_x^2 \psi|_i^n (-\Delta x)^2 + O(\Delta x^3)$$

$$\rightsquigarrow \begin{aligned} \psi_i^{n+1} = \psi_i^n - \left[\frac{C + |C|}{2} (\psi_i^n - \psi_{i-1}^n) \right. \\ \left. + \frac{C - |C|}{2} (\psi_{i+1}^n - \psi_i^n) \right] \end{aligned}$$

MPDATA key concepts: numerical diffusion & modified-equation analysis

- UPWIND incurs **numerical diffusion**, quantifiable using Taylor expansion (const. C for simplicity):

$$\psi_i^{n+1} = \psi_i^n + \partial_t \psi|_i^n (+\Delta t) + \frac{1}{2} \partial_t^2 \psi|_i^n (+\Delta t)^2 + O(\Delta t^3)$$

$$\psi_{i+1}^n = \psi_i^n + \partial_x \psi|_i^n (+\Delta x) + \frac{1}{2} \partial_x^2 \psi|_i^n (+\Delta x)^2 + O(\Delta x^3)$$

$$\psi_{i-1}^n = \psi_i^n + \partial_x \psi|_i^n (-\Delta x) + \frac{1}{2} \partial_x^2 \psi|_i^n (-\Delta x)^2 + O(\Delta x^3)$$

$$\rightsquigarrow \begin{aligned} \psi_i^{n+1} = \psi_i^n - \left[\frac{C + |C|}{2} (\psi_i^n - \psi_{i-1}^n) \right. \\ \left. + \frac{C - |C|}{2} (\psi_{i+1}^n - \psi_i^n) \right] \end{aligned}$$

- which substituted to the UPWIND formulæ yields (up to second-order terms):

$$\partial_t \psi|_i^n \Delta t + \underbrace{\partial_t^2 \psi|_i^n}_{u^2 \partial_x^2 \psi} \frac{\Delta t^2}{2} = -C \Delta x \partial_x \psi|_i^n + \frac{|C|}{2} \Delta x^2 \partial_x^2 \psi|_i^n$$

MPDATA key concepts: numerical diffusion & modified-equation analysis

- UPWIND incurs **numerical diffusion**, quantifiable using Taylor expansion (const. C for simplicity):

$$\psi_i^{n+1} = \psi_i^n + \partial_t \psi|_i^n (+\Delta t) + \frac{1}{2} \partial_t^2 \psi|_i^n (+\Delta t)^2 + O(\Delta t^3)$$

$$\psi_{i+1}^n = \psi_i^n + \partial_x \psi|_i^n (+\Delta x) + \frac{1}{2} \partial_x^2 \psi|_i^n (+\Delta x)^2 + O(\Delta x^3)$$

$$\psi_{i-1}^n = \psi_i^n + \partial_x \psi|_i^n (-\Delta x) + \frac{1}{2} \partial_x^2 \psi|_i^n (-\Delta x)^2 + O(\Delta x^3)$$

$$\begin{aligned} \psi_i^{n+1} = \psi_i^n - \left[\frac{C + |C|}{2} (\psi_i^n - \psi_{i-1}^n) \right. \\ \left. + \frac{C - |C|}{2} (\psi_{i+1}^n - \psi_i^n) \right] \end{aligned}$$

- which substituted to the UPWIND formulæ yields (up to second-order terms):

$$\partial_t \psi|_i^n \Delta t + \underbrace{\partial_t^2 \psi|_i^n}_{u^2 \partial_x^2 \psi} \frac{\Delta t^2}{2} = -C \Delta x \partial_x \psi|_i^n + \frac{|C|}{2} \Delta x^2 \partial_x^2 \psi|_i^n$$

where $\partial_t^2 \psi$ can be replaced with spatial derivative using a time derivative of the advection eq.:

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

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

MPDATA key concepts: antidiiffusive pseudo-velocities

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

$$\partial_t \psi = k \partial_x^2 \psi + \dots \quad \leadsto \quad \partial_t \psi + \partial_x \left(k \underbrace{\frac{\partial_x \psi}{\psi}}_{\text{pseudo-velocity}} \psi \right) = \dots$$

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

MPDATA key concepts: antidiiffusive pseudo-velocities

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

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

(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)

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

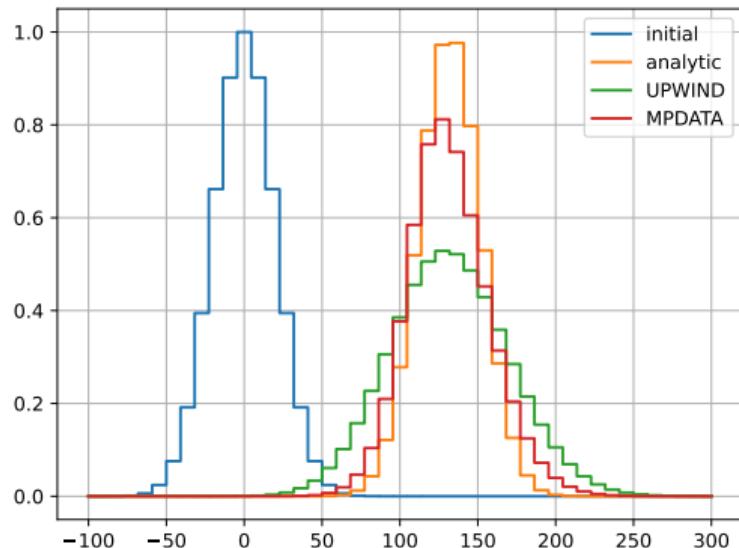
(Smolarkiewicz 1983 MWR, 1984 JCP: doi:10.1016/0021-9991(84)90121-9)

MPDATA hello-world (1D, single iteration) implementation

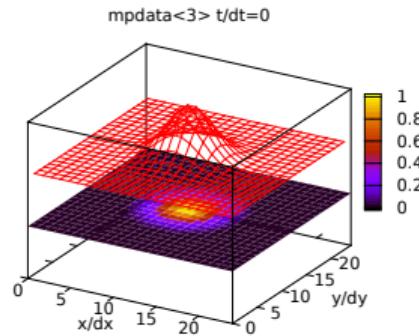
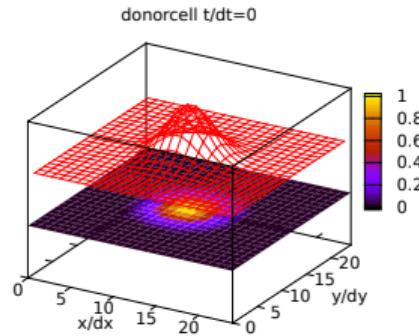
```
1 def C_corr(C: np.ndarray, i: slice, psi: np.ndarray):
2     return (abs(C[i - hlf]) - C[i - hlf] ** 2) * (
3         psi[i] - psi[i - one]
4     ) / (
5         psi[i - one] + psi[i]
6     )
7
7 def mpdata(nt: int, C: np.ndarray, psi: np.ndarray):
8     i = slice(1, len(psi) - 1)
9     i_ext = slice(1, len(psi))
10    for _ in range(nt):
11        upwind(psi, i, C)
12        upwind(psi, i, C_corr(C, i_ext, psi))
```

MPDATA hello-world (1D, single iteration) implementation

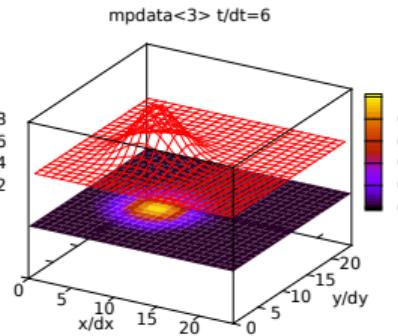
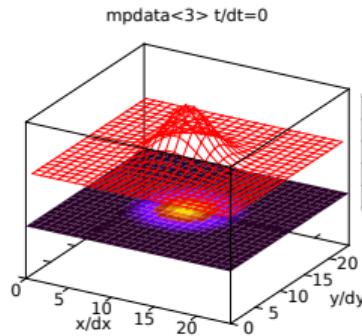
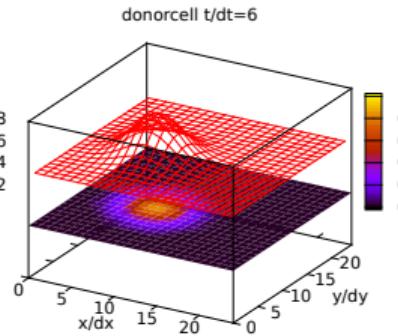
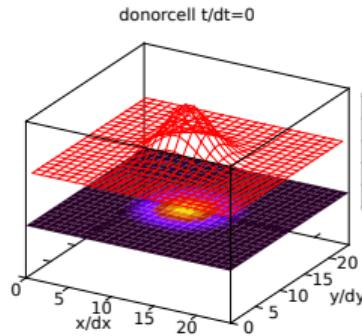
```
1 def C_corr(C: np.ndarray, i: slice, psi: np.ndarray):
2
3     return (abs(C[i - hlf]) - C[i - hlf] ** 2) * (
4
5         psi[i] - psi[i - one]
6
7     ) / (
8
9         psi[i - one] + psi[i]
10
11 )
12
13 def mpdata(nt: int, C: np.ndarray, psi: np.ndarray):
14
15     i = slice(1, len(psi) - 1)
16
17     i_ext = slice(1, len(psi))
18
19     for _ in range(nt):
20
21         upwind(psi, i, C)
22
23         upwind(psi, i, C_corr(C, i_ext, psi))
```



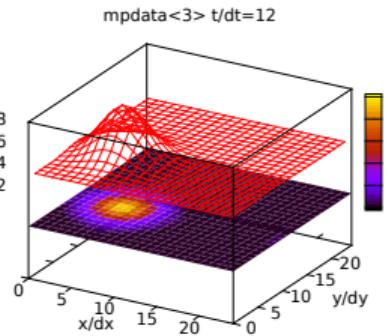
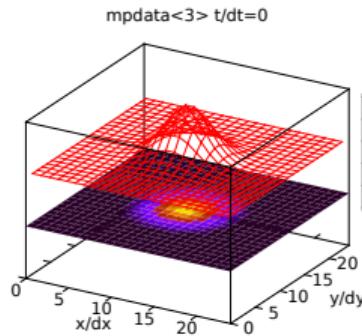
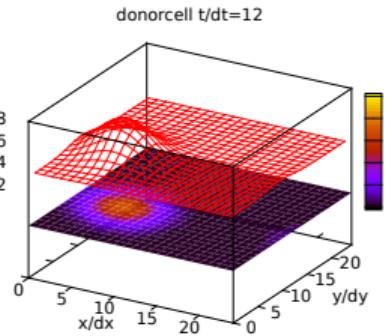
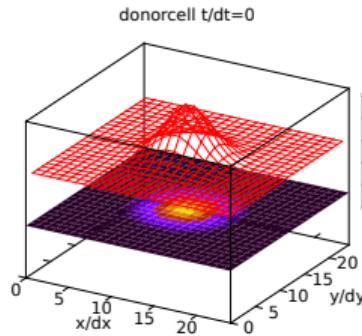
MPDATA: "M" for multi-dimensional (2D is not a sum of two 1D MPDATAs)



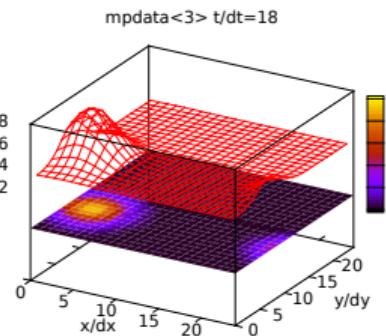
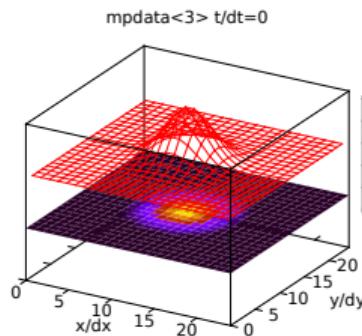
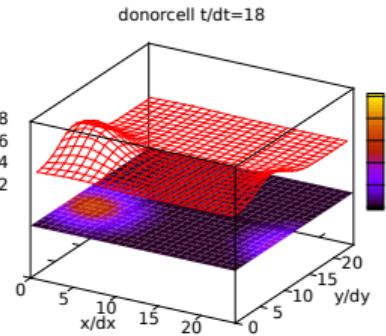
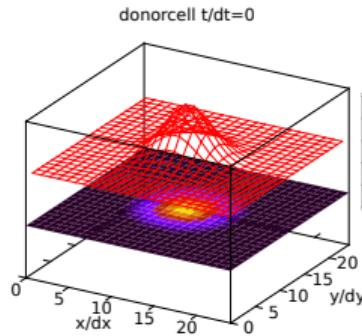
MPDATA: "M" for multi-dimensional (2D is not a sum of two 1D MPDATAs)



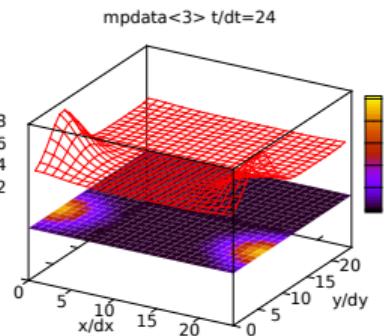
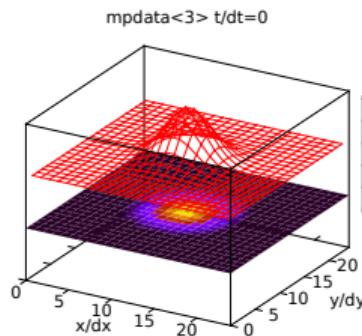
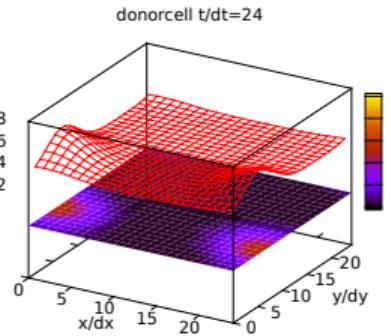
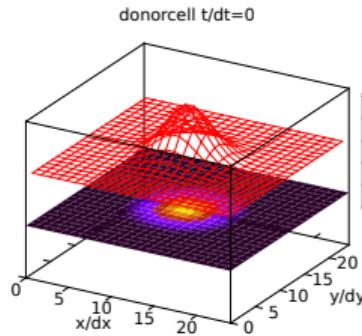
MPDATA: "M" for multi-dimensional (2D is not a sum of two 1D MPDATAs)



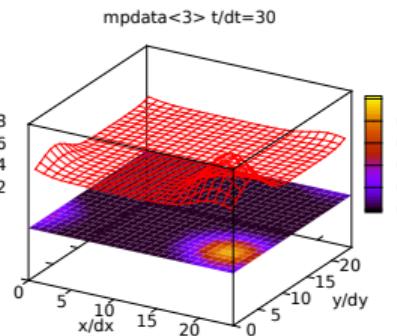
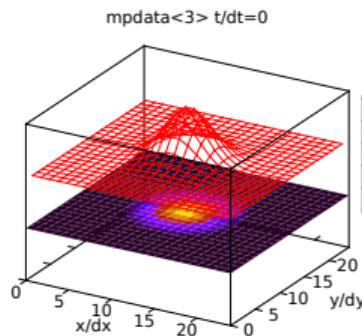
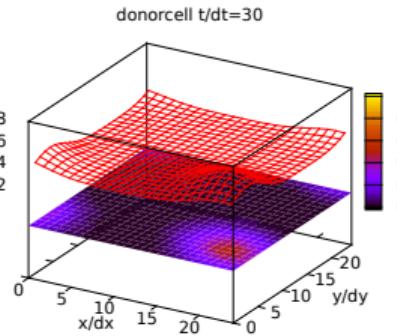
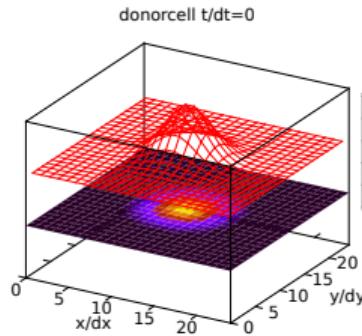
MPDATA: "M" for multi-dimensional (2D is not a sum of two 1D MPDATAs)



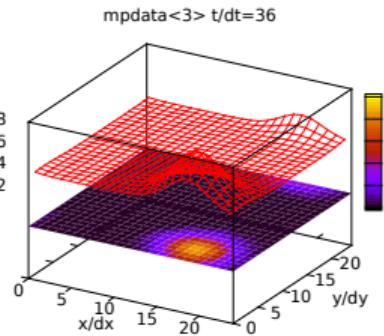
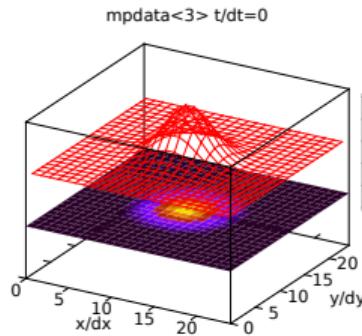
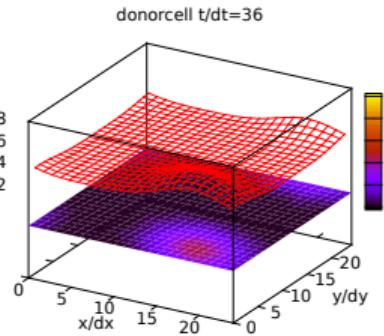
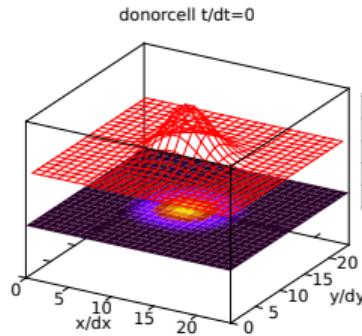
MPDATA: "M" for multi-dimensional (2D is not a sum of two 1D MPDATAs)



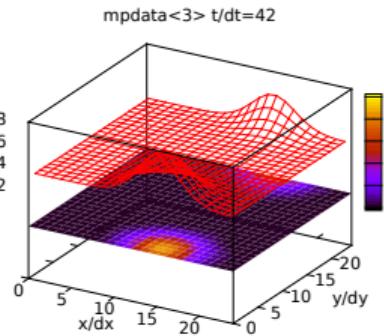
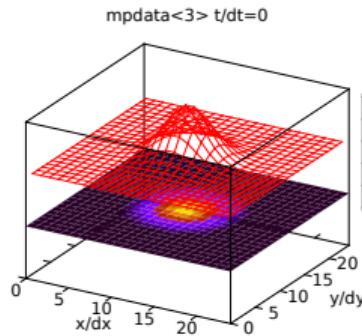
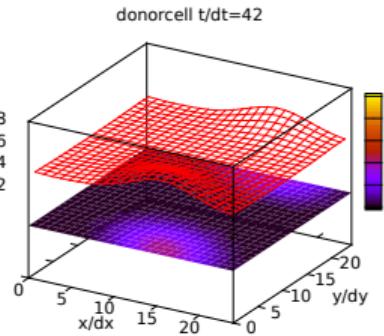
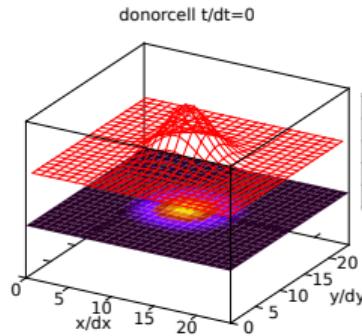
MPDATA: "M" for multi-dimensional (2D is not a sum of two 1D MPDATAs)



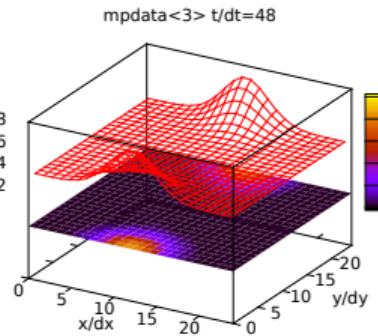
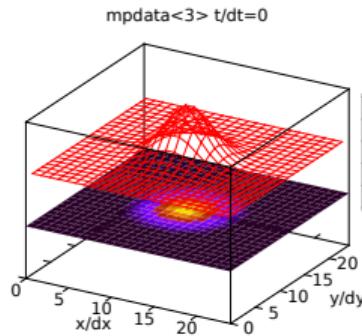
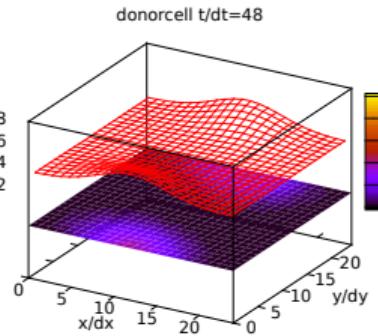
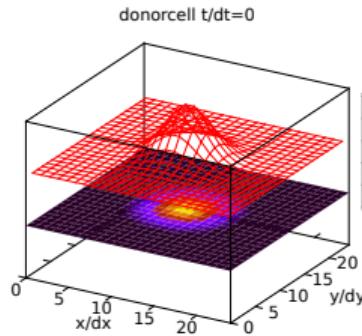
MPDATA: "M" for multi-dimensional (2D is not a sum of two 1D MPDATAs)



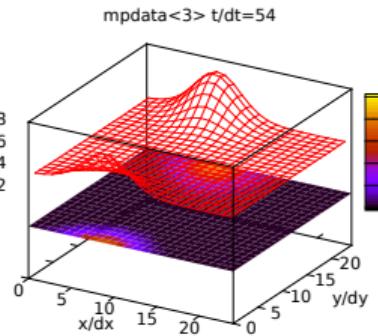
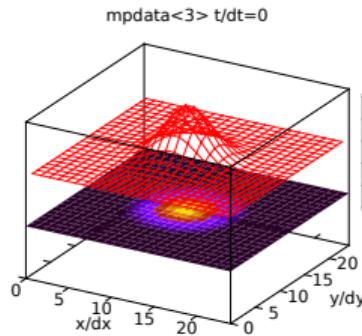
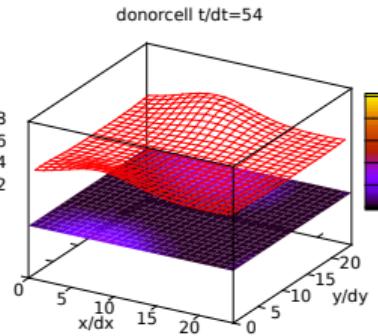
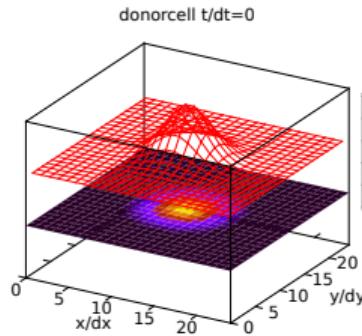
MPDATA: "M" for multi-dimensional (2D is not a sum of two 1D MPDATAs)



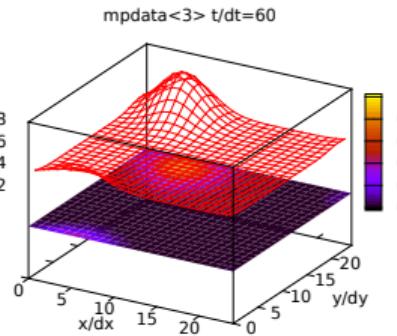
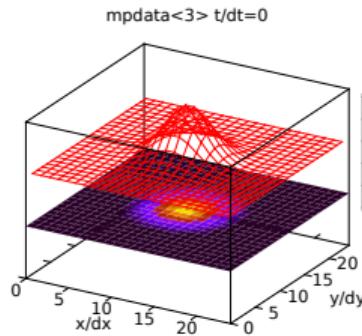
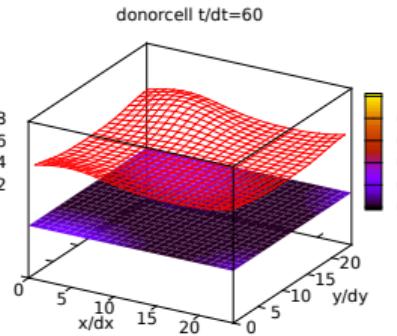
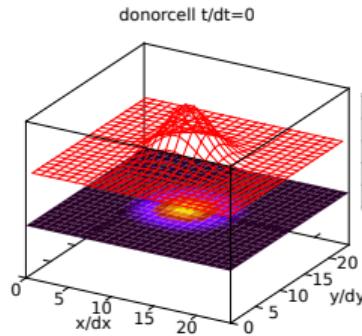
MPDATA: "M" for multi-dimensional (2D is not a sum of two 1D MPDATAs)



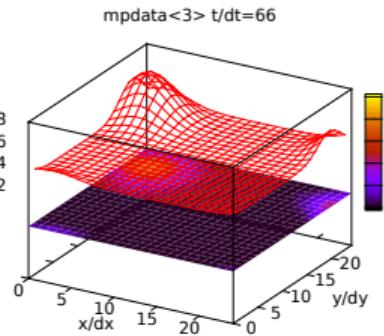
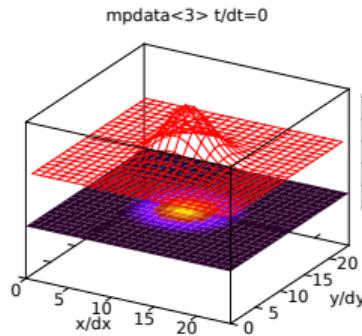
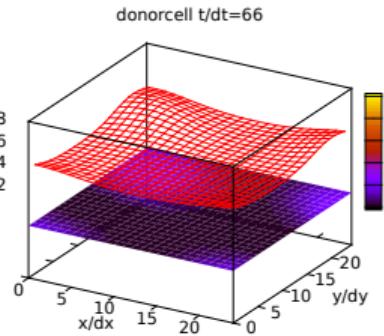
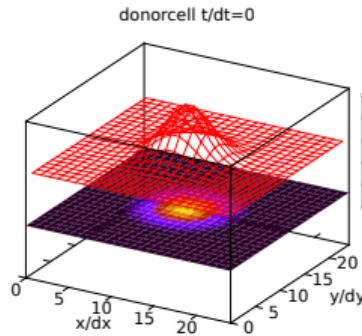
MPDATA: "M" for multi-dimensional (2D is not a sum of two 1D MPDATAs)



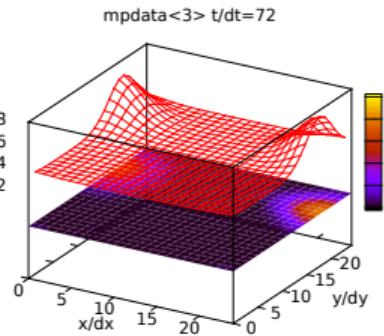
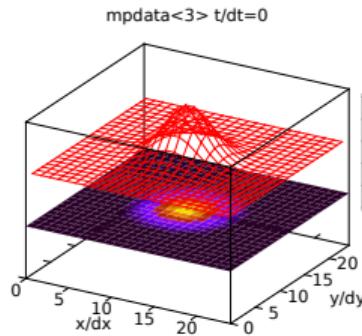
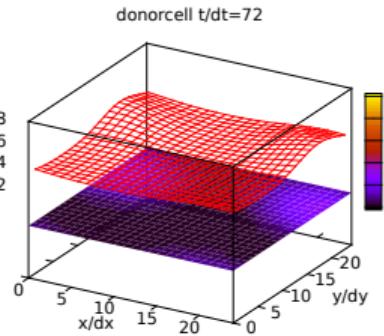
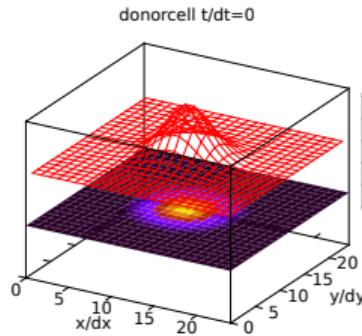
MPDATA: "M" for multi-dimensional (2D is not a sum of two 1D MPDATAs)



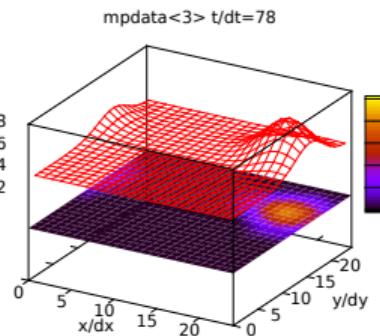
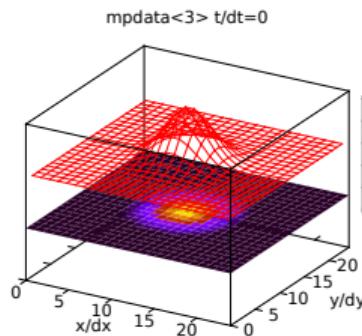
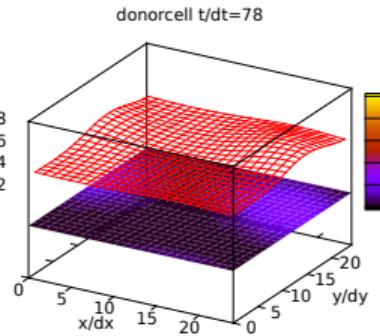
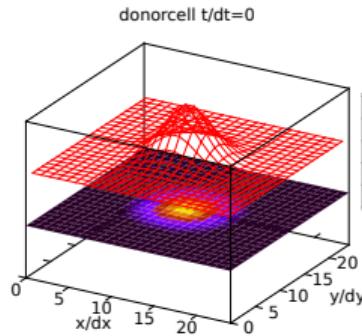
MPDATA: "M" for multi-dimensional (2D is not a sum of two 1D MPDATAs)



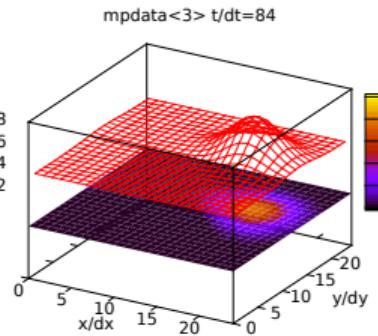
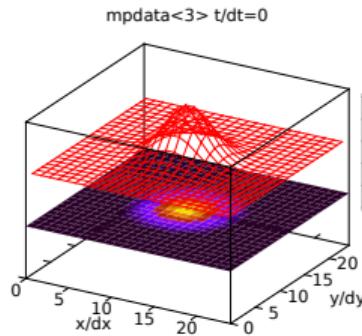
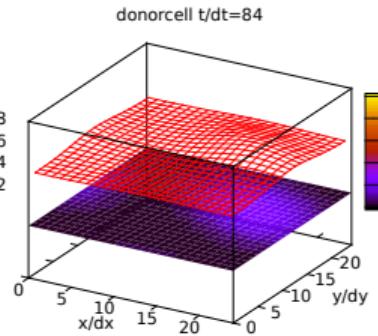
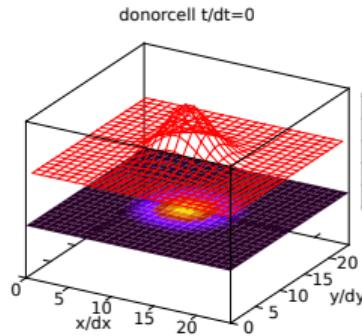
MPDATA: "M" for multi-dimensional (2D is not a sum of two 1D MPDATAs)



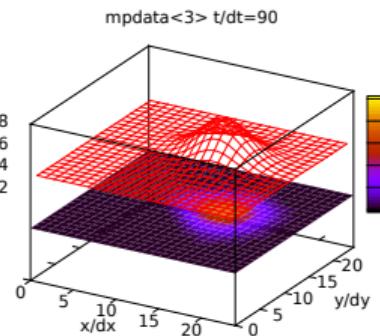
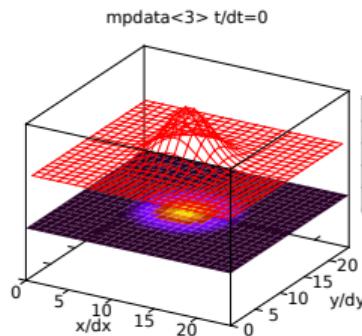
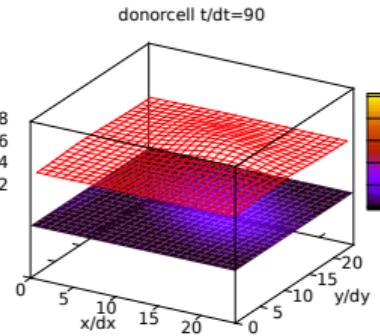
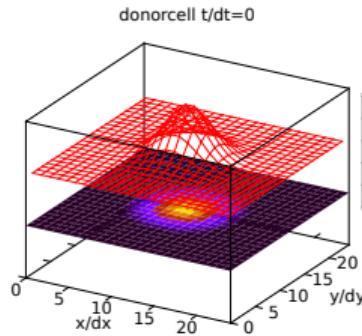
MPDATA: "M" for multi-dimensional (2D is not a sum of two 1D MPDATAs)



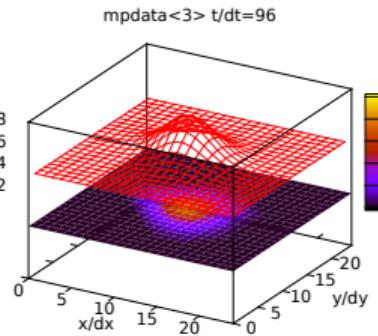
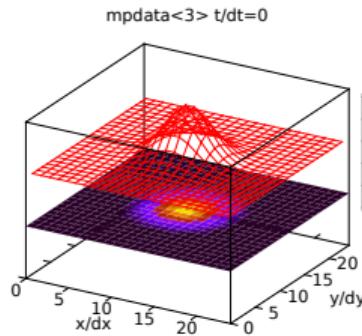
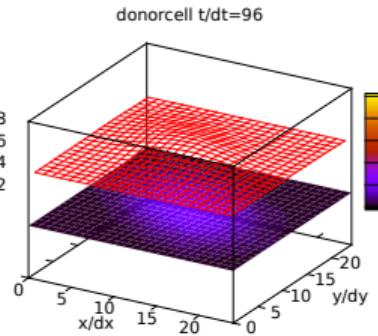
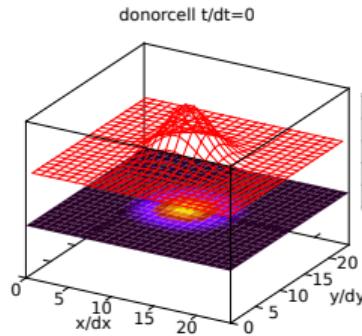
MPDATA: "M" for multi-dimensional (2D is not a sum of two 1D MPDATAs)



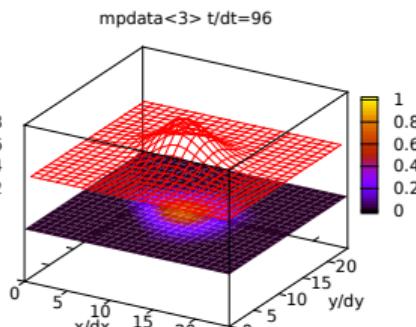
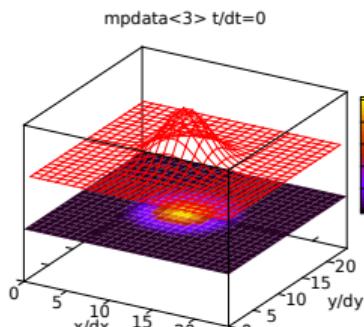
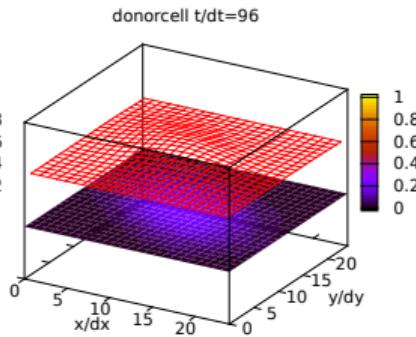
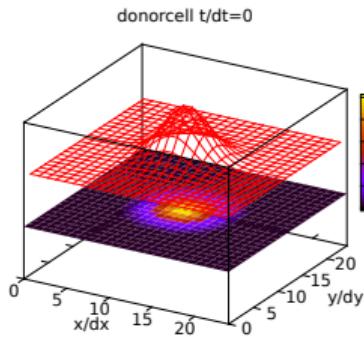
MPDATA: "M" for multi-dimensional (2D is not a sum of two 1D MPDATAs)



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



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



$$\begin{aligned}
 & \sum_{d=0}^1 \psi_{[i,j]+\pi_{1,0}^d} \equiv \psi_{[i+1,j]} + \psi_{[i,j+1]} \\
 & C'^{[d]}_{[i,j]+\pi_{1/2,0}^d} = \left| C^{[d]}_{[i,j]+\pi_{1/2,0}^d} \right| \cdot \left[1 - \left| C^{[d]}_{[i,j]+\pi_{1/2,0}^d} \right| \right] \cdot A^{[d]}_{[i,j]}(\psi) \\
 & \quad - \sum_{q=0, q \neq d}^N C^{[d]}_{[i,j]+\pi_{1/2,0}^d} \cdot \overline{C}^{[q]}_{[i,j]+\pi_{1/2,0}^d} \cdot B^{[d]}_{[i,j]}(\psi) \\
 & \overline{C}^{[q]}_{[i,j]+\pi_{1/2,0}^d} = \frac{1}{4} \cdot \left(C^{[q]}_{[i,j]+\pi_{1,1/2}^d} + C^{[q]}_{[i,j]+\pi_{0,1/2}^d} + \right. \\
 & \quad \left. C^{[q]}_{[i,j]+\pi_{1,-1/2}^d} + C^{[q]}_{[i,j]+\pi_{0,-1/2}^d} \right) \\
 & A^{[d]}_{[i,j]} = \frac{\psi_{[i,j]+\pi_{1,0}^d} - \psi_{[i,j]}}{\psi_{[i,j]+\pi_{1,0}^d} + \psi_{[i,j]}} \\
 & B^{[d]}_{[i,j]} = \frac{1}{2} \frac{\psi_{[i,j]+\pi_{1,1}^d} + \psi_{[i,j]+\pi_{0,1}^d} - \psi_{[i,j]+\pi_{1,-1}^d} - \psi_{[i,j]+\pi_{0,-1}^d}}{\psi_{[i,j]+\pi_{1,1}^d} + \psi_{[i,j]+\pi_{0,1}^d} + \psi_{[i,j]+\pi_{1,-1}^d} + \psi_{[i,j]+\pi_{0,-1}^d}}
 \end{aligned}$$

8

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
- recursive anti-diffusive velocities: Beeson & 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 (Numerical Weather Prediction):

- COSMO
- ECMWF IFS

open-source:

integrated into CFD packages:

MPDATA implementations

known closed-source (Numerical Weather Prediction):

- COSMO
- ECMWF IFS

open-source:

integrated into CFD packages:

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

MPDATA implementations

known closed-source (Numerical Weather Prediction):

- COSMO
- ECMWF IFS

open-source:

integrated into CFD packages:

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

MPDATA implementations

known closed-source (Numerical Weather Prediction):

- COSMO
- ECMWF IFS

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 (Numerical Weather Prediction):

- COSMO
- ECMWF IFS

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	igfw/bE_SDs	NCAR

MPDATA implementations

known closed-source (Numerical Weather Prediction):

- COSMO
- ECMWF IFS

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	igfw/bE_SDs	NCAR
apc-llc/mpdata	C/CUDA	3D	apc-llc	RAS (?)

MPDATA implementations

known closed-source (Numerical Weather Prediction):

- COSMO
- ECMWF IFS

open-source:

integrated into CFD packages:

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

reusable:

libmpdata++	C++/Blitz++	1,2,3D	Q /igfw	IGF FUW
-------------	-------------	--------	-------------------------	---------

MPDATA implementations

known closed-source (Numerical Weather Prediction):

- COSMO
- ECMWF IFS

open-source:

integrated into CFD packages:

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

reusable:

libmpdata++	C++/Blitz++	1,2,3D	Q /igfw	IGF FUW
PyMPDATA	Python/Numba	1,2,3D	Q /open-atmos	UJ, AGH

plan of the talk

- MPDATA scheme and its implementations
- PyMPDATA: pure-Python just-in-time compiled MPDATA
- PyMPDATA "examples" and documentation
- PyMPDATA performance vs. C++
- MPI, HPC & distributed-memory parallelisation?
- PyMPDATA in teaching (i.e., implemented by students!)

PyMPDATA: design goals, tech stack, features

- Numba JIT \rightsquigarrow pure-Python code with compiled-language performance
(plus OpenMP-like multi-threading, but no profiling tools)



PyMPDATA: 100% Python codebase

```
@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
    )
```

PyMPDATA: design goals, tech stack, features

- Numba JIT \rightsquigarrow pure-Python code with compiled-language performance (plus OpenMP-like multi-threading, but no profiling tools)
- 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, but no profiling tools)
- 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, but no profiling tools)
- 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, but no profiling tools)
- 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, but no profiling tools)
- 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, but no profiling tools)
- 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, but no profiling tools)
- 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

plan of the talk

- MPDATA scheme and its implementations
- PyMPDATA: pure-Python just-in-time compiled MPDATA
- PyMPDATA "examples" and documentation
- PyMPDATA performance vs. C++
- MPI, HPC & distributed-memory parallelisation?
- PyMPDATA in teaching (i.e., implemented by students!)

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



Documentation

PyMPDATA

What is PyMPDATA?

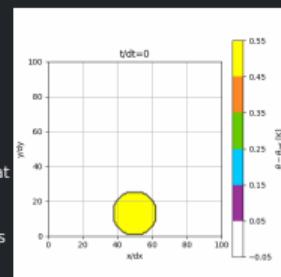
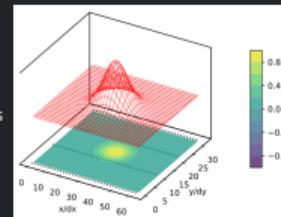
PyMPDATA is a **Numba-accelerated** multi-threaded Pythonic implementation of the **MPDATA algorithm** of Smolarkiewicz et al. used in geophysical fluid dynamics and beyond for **numerically solving generalised convection-diffusion PDEs**. PyMPDATA supports integration in 1D, 2D and 3D structured meshes 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).



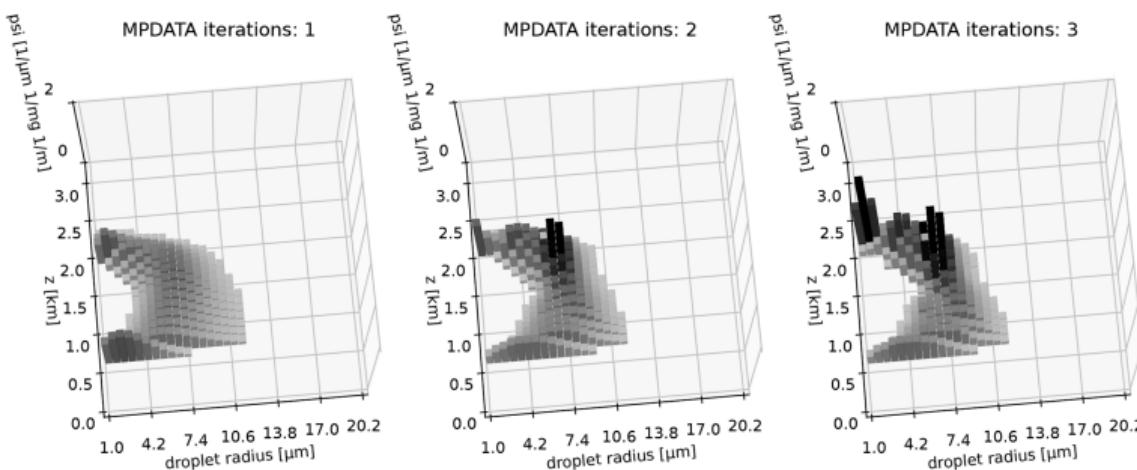
MPDATA for condensational growth in bin μ -physics (Olesik et al. 2022)

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



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

Michael A. Olesik¹, Jakub Banaśkiewicz², Piotr Bartman², Manuel Baumgartner^{3,4}, Simon Unterstrasser⁵, and Sylwester Arabas^{6,2}



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

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

arXiv > q-fin > arXiv:2505.24435

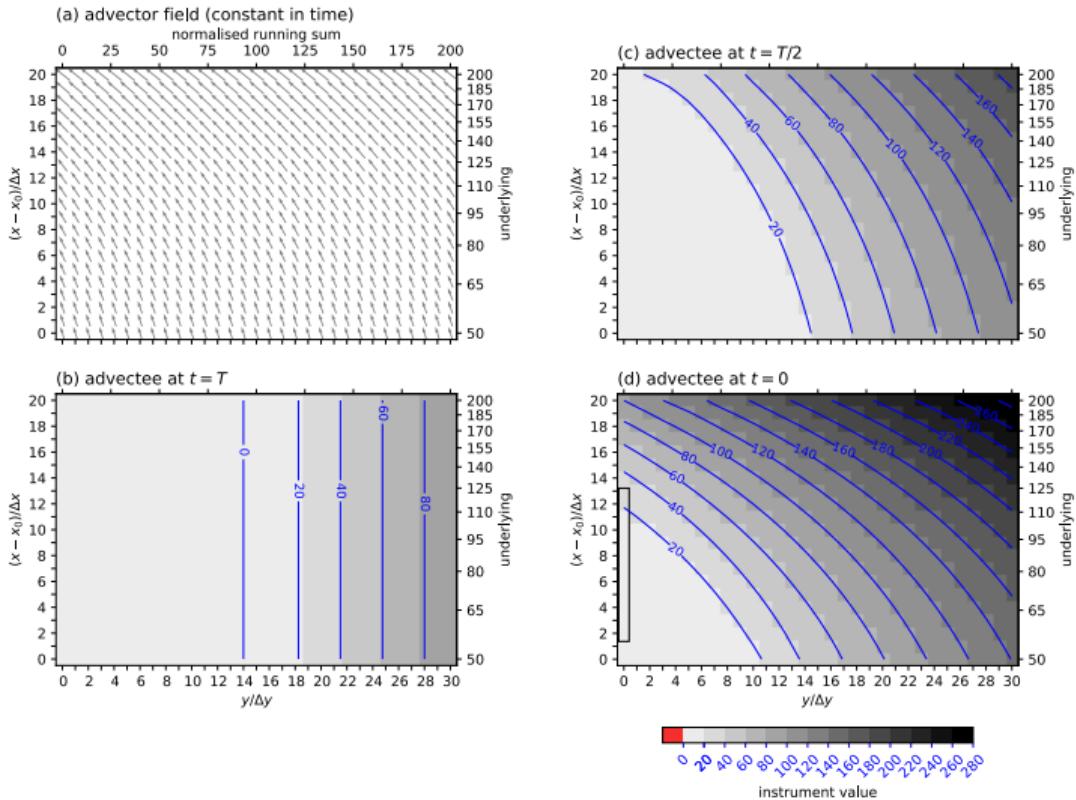
Quantitative Finance > Computational Finance

[Submitted on 30 May 2025]

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

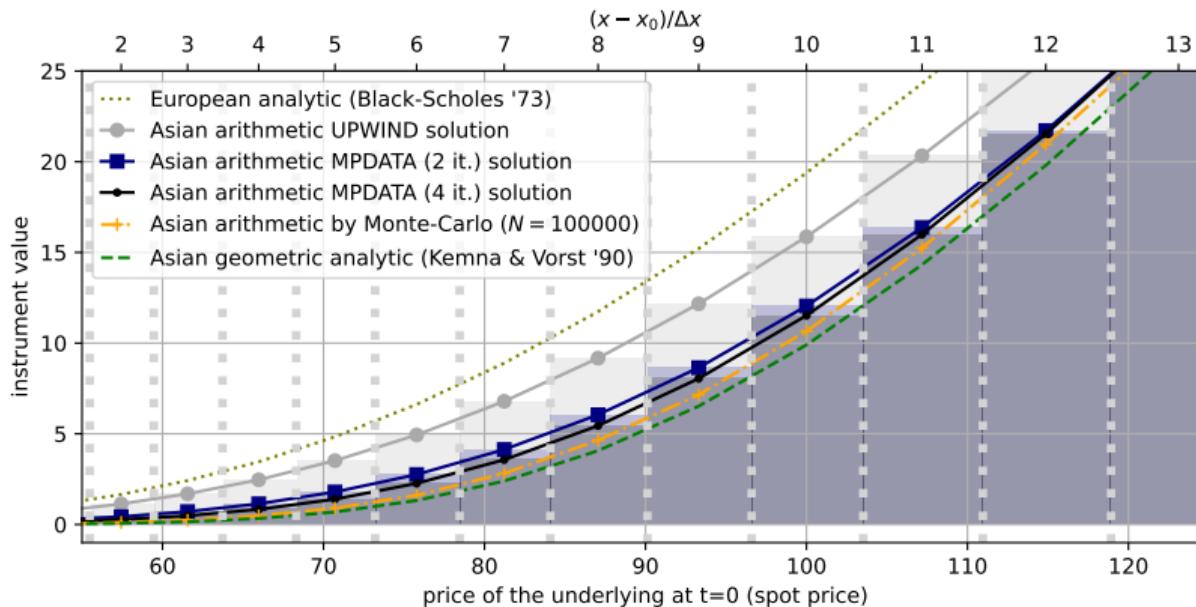
Pawel Magnuszewski, Sylwester Arabas

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)

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

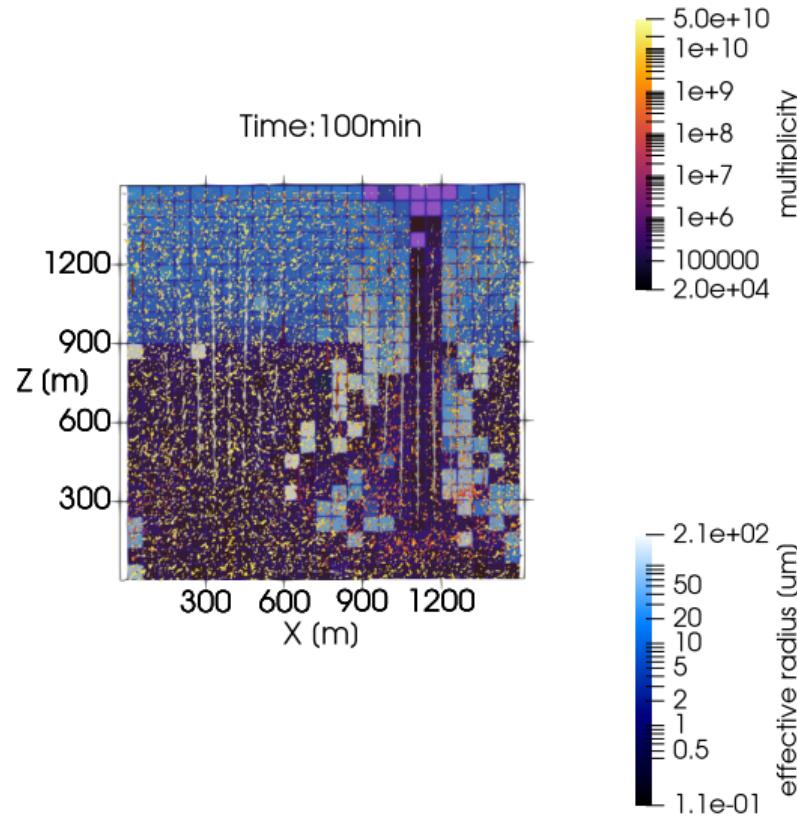


Eulerian transport for PySDM examples (the original reason for PyMPDATA dev)

<https://pypi.org/p/PySDM>



PySDM



plan of the talk

- MPDATA scheme and its implementations
- PyMPDATA: pure-Python just-in-time compiled MPDATA
- PyMPDATA "examples" and documentation
- **PyMPDATA performance vs. C++**
- MPI, HPC & distributed-memory parallelisation?
- PyMPDATA in teaching (i.e., implemented by students!)

Geosci. Model Dev., 8, 1005–1032, 2015
www.geosci-model-dev.net/8/1005/2015/
doi:10.5194/gmd-8-1005-2015



libmpdata++ 1.0: a library of parallel MPDATA solvers for systems of generalised transport equations

A. Jaruga¹, S. Arabas¹, D. Jarecka^{1,2}, H. Pawlowska¹, P. K. Smolarkiewicz³, and M. Waruszewski¹

¹Institute of Geophysics, Faculty of Physics, University of Warsaw, Warsaw, Poland

²National Center for Atmospheric Research, Boulder, CO, USA

³European Centre for Medium-Range Weather Forecasts, Reading, UK

Correspondence to: A. Jaruga (ajaruga@igf.fuw.edu.pl) and H. Pawlowska (hanna.pawlowska@igf.fuw.edu.pl)



DOI: [10.21105/joss.03896](https://doi.org/10.21105/joss.03896)

Software

- [Review](#)
- [Repository](#)
- [Archive](#)

Editor: [Afon Smith](#)

Reviewers:

- [@Chiil](#)
- [@wdeconinck](#)

Submitted: 25 October 2021

Published: 05 September 2022

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

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

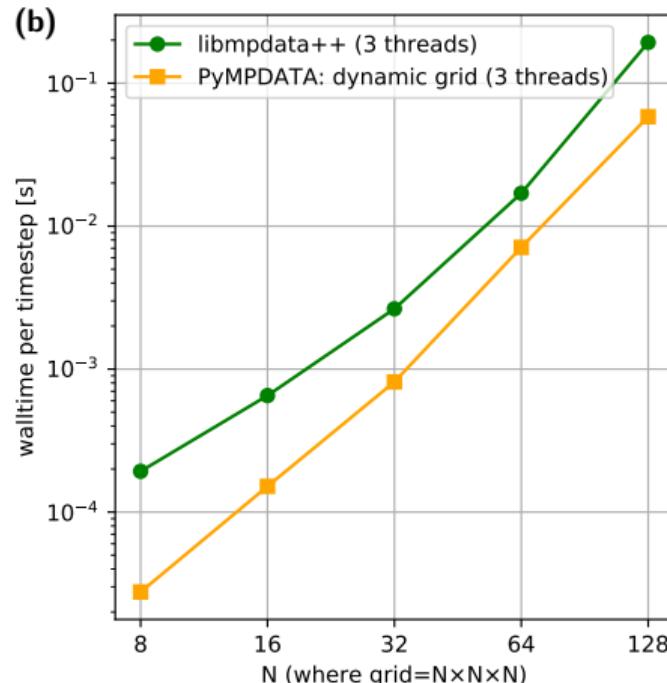
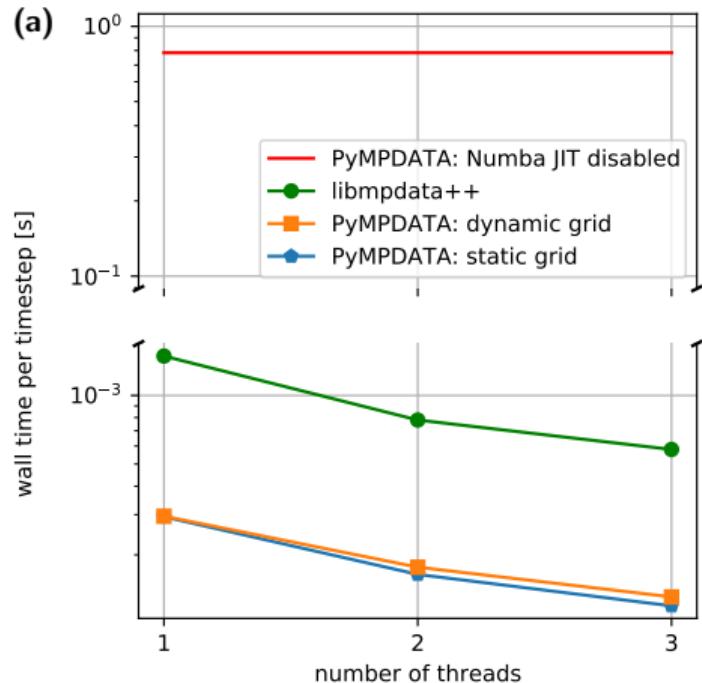
Piotr Bartman ¹, Jakub Banaśkiewicz¹, Szymon Drenda¹, Maciej Manna¹, Michael A. Olesik ¹, Paweł Rozwoda¹, Michał Sadowski ¹, and Sylwester Arabas ^{1,2}

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



plan of the talk

- MPDATA scheme and its implementations
- PyMPDATA: pure-Python just-in-time compiled MPDATA
- PyMPDATA "examples" and documentation
- PyMPDATA performance vs. C++
- MPI, HPC & distributed-memory parallelisation?
- PyMPDATA in teaching (i.e., implemented by students!)

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> 

Under a Creative Commons license 

● Open access

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.

The screenshot shows a project page for "pympdata-mpi 0.1.1". At the top, there's a search bar labeled "Search projects" and navigation links for "Help", "Docs", "Sponsors", "Log in", and "Register". Below the header, the project name "pympdata-mpi 0.1.1" is displayed, along with a green button containing a checkmark and the text "Latest version". A release date of "Released: Apr 4, 2025" is also shown. On the left, there's a button with the command "pip install pympdata-mpi" and a small icon. The main content area contains the text "PyMPDATA + numba-mpi coupler sandbox".

Navigation

Project description

Release history

Download files

Verified details

These details have been [verified by PyPI](#)

Maintainers



Sfonxu

Project description

PyMPDATA-MPI

Python 3 LLVM Numba Linux macOS Maintained? yes

PL Funding by NCN License GPL v3 Copyright Jagiellonian University DOI 10.5281/zenodo.10866521

pull requests 7 open pull requests 131 closed

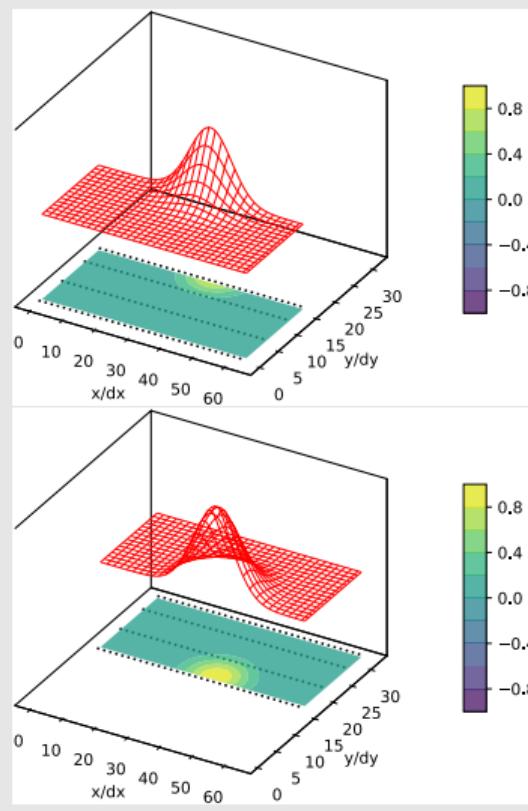
issues 14 open issues 36 closed

tests+pypi no status pypi package 0.1.1 docs pdoc.dev codecov 72%

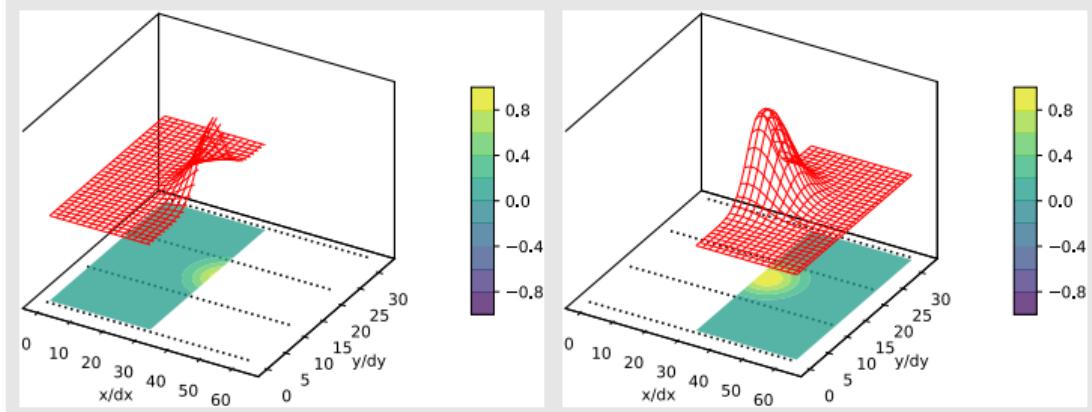
PyMPDATA-MPI constitutes a [PyMPDATA](#) + [numba-mpi](#) coupler enabling numerical solutions of transport equations with the MPDATA numerical scheme in a hybrid parallelisation model with both multi-threading and MPI distributed memory communication. PyMPDATA-MPI adapts to API of PyMPDATA offering domain decomposition logic.

PyMPDATA-MPI: customisable hybrid threading + MPI parallelisation

threading dim = MPI dim



threading dimension \neq MPI dimension



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

plan of the talk

- MPDATA scheme and its implementations
- PyMPDATA: pure-Python just-in-time compiled MPDATA
- PyMPDATA "examples" and documentation
- PyMPDATA performance vs. C++
- MPI, HPC & distributed-memory parallelisation?
- PyMPDATA in teaching (i.e., implemented by students!)

PyMPDATA documentation (incl. Matlab, Julia and Rust interoperability)

PyMPDATA_examples

Search...

Submodules

- Bjerksund_and_Stensland_1993
- Black_Scholes_1973
- analysis_figures_2_and_3
- analysis_table_1
- colors
- options
- setup1_european_corridor
- setup2_amERICAN_put
- simulation

built with 

PyMPDATA_examples

.Arabas_and_Farhat_2020

This example implements simulations presented in the [Arabas and Farhat 2020](#) study on pricing of European and American options using MPDATA.

Each notebook in this directory corresponds to a figure or a table in the paper.

fig_1.ipynb: [render on GitHub](#) [launch binder](#) [Open in Colab](#)

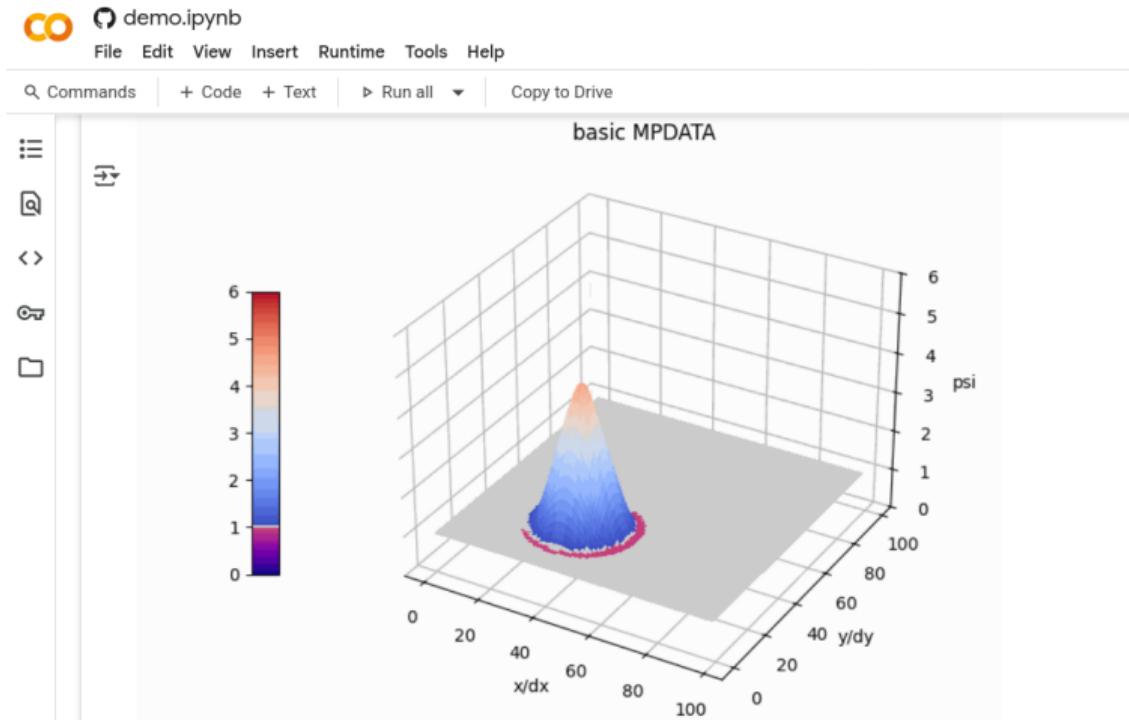
fig_2.ipynb: [render on GitHub](#) [launch binder](#) [Open in Colab](#)

fig_3.ipynb: [render on GitHub](#) [launch binder](#) [Open in Colab](#)

tab_1.ipynb: [render on GitHub](#) [launch binder](#) [Open in Colab](#)

[▶ View Source](#)

PyMPDATA documentation (incl. Matlab, Julia and Rust interoperability)



PyMPDATA documentation (incl. Matlab, Julia and Rust interoperability)

Bibliography with code cross-references

The list below summarises all literature references included in PyMPDATA codebase and includes links to both the referenced papers, as well as to the referring PyMPDATA source files.

1. Anderson & Fattahi 1974: *"A Comparison of Numerical Solutions of the Advective Equation"*
 - examples/PyMPDATA_examples/Molenkamp_test_as_in_Jaruga_et_al_2015_Fig_12/demo.ipynb
 - examples/PyMPDATA_examples/wikipedia_example/demo.ipynb
2. Arabas & Farhat 2020 (J. Comput. Appl. Math. 373): *"Derivative pricing as a transport problem: MPDATA solutions to Black-Scholes-type equations"*
 - examples/PyMPDATA_examples/Arabas_and_Farhat_2020/_init__.py
 - examples/PyMPDATA_examples/Arabas_and_Farhat_2020/fig_1.ipynb
 - examples/PyMPDATA_examples/Arabas_and_Farhat_2020/fig_2.ipynb
 - examples/PyMPDATA_examples/Arabas_and_Farhat_2020/fig_3.ipynb
 - examples/PyMPDATA_examples/Arabas_and_Farhat_2020/tab_1.ipynb
3. Arabas et al. 2014 (Sci. Prog. 22): *"Formula Translation in Blitz++, NumPy and Modern Fortran: A Case Study of the Language Choice Tradeoffs"*
 - docs/markdown/pympdata_landing.md
4. Barraquand & Pudet 1996 (Math. Financ. 6): *"Pricing of American path-dependent contingent claims"*
 - examples/PyMPDATA_examples/Magnuszewski_et_al_2025/barraquand_data.py
5. Bartman et al. 2022 (J. Open Source Soft. 7): *"PyMPDATA v1: Numba-accelerated implementation of MPDATA with examples in Python, Julia and Matlab"*
 - examples/PyMPDATA_examples/Bartman_et_al_2022/_init__.py
 - examples/PyMPDATA_examples/Bartman_et_al_2022/fig_X.ipynb
6. Beeson & Margolin 1988 (Nuclear explosives code developer's conference, Boulder, CO, USA): *"DPDC (double-pass donor cell): A second-order monotone scheme for advection"*
 - PyMPDATA/options.py
 - examples/docs/pympdata_examples_landing.md
7. Capiński and Zastawniak 2012 (Cambridge University Press): *"Numerical Methods in Finance with C++"*
 - examples/PyMPDATA_examples/Magnuszewski_et_al_2025/monte_carlo.py
8. Jarecka et al. 2015 (J. Comp. Phys. 289): *"A spreading drop of shallow water"*
 - examples/PyMPDATA_examples/jarecka_et_al_2015/_init__.py
 - examples/PyMPDATA_examples/jarecka_et_al_2015/fig_6.ipynb
9. Jaruga et al. 2015 (Geosci. Model Dev. 8): *"libmpdata++ 1.0: a library of parallel MPDATA solvers for systems of generalised transport equations "*
 - docs/markdown/pympdata_landing.md
 - examples/PyMPDATA_examples/jaruga_et_al_2015/_init__.py
 - examples/PyMPDATA_examples/jaruga_et_al_2015/fig19.ipynb

PyMPDATA documentation (incl. Matlab, Julia and Rust interoperability)

The image shows a screenshot of the PyMPDATA API Documentation. On the left, there's a sidebar with navigation links like "PyMPDATA", "API Documentation", and "Source". Below that is the main content area. In the main content area, there's a code snippet for the `advance` method of the `Solver` class. The code is color-coded for syntax highlighting. To the right of the code, there's a "View Source" link. At the bottom of the main content area, there's a detailed description of the `advance` method.

```
def advance(
    self,
    n_steps: int,
    mu_coeff: Optional[tuple] = None,
    post_step=None,
    post_iter=None
):
    def advance(
        self,
        n_steps: int,
        mu_coeff: Union[tuple, None] = None,
        post_step=None,
        post_iter=None,
    ):
        """advances solution by `n_steps` steps, optionally accepts: a tuple of diffusion coefficients (one value per dimension) as well as `post_iter` and `post_step` callbacks expected to be numba.jitclass'es with a `call` method, for signature see `PostStepNull` and `PostIterNull`;
        returns CPU time per timestep (units as returned by `clock.clock()`)"""
        if mu_coeff is not None:
            assert self.__stepper.options.non_zero_mu_coeff
        else:
            mu_coeff = (0.0, 0.0, 0.0)
        if (
            self.__stepper.options.non_zero_mu_coeff
            and not self.__fields["g_factor"].meta[META_IS_NULL]
        ):
            raise NotImplementedError()
        post_step = post_step or PostStepNull()
        post_iter = post_iter or PostIterNull()

        return self.__stepper(
            n_steps=n_steps,
            mu_coeff=mu_coeff,
            post_step=post_step,
            post_iter=post_iter,
            fields=self.__fields,
        )

```

advances solution by `n_steps` steps, optionally accepts: a tuple of diffusion coefficients (one value per dimension) as well as `post_iter` and `post_step` callbacks expected to be numba.jitclass'es with a `call` method, for signature see `PostStepNull` and `PostIterNull`; returns CPU time per timestep (units as returned by `clock.clock()`)

PyMPDATA documentation (incl. Matlab, Julia and Rust interoperability)

Module Index

Contents

- Introduction
- Tutorial (in Python, Julia, Rust and Matlab)
 - Options class
 - Aarakawa-C grid layer and boundary conditions
 - Stepper
 - Solver
- Debugging
- Contributing, reporting issues, seeking support
- Other open-source MPDATA implementations:
- Other Python packages for solving hyperbolic transport equations

Submodules

- boundary_conditions
- impl
- options
- scalar_field

As an example, the code below shows how to instantiate a scalar and a vector field given a 2D constant-velocity problem, using a grid of 24x24 points, Courant numbers of -0.5 and -0.25 in "x" and "y" directions, respectively, with periodic boundary conditions and with an initial Gaussian signal in the scalar field (settings as in Fig. 5 in [Arabas et al. 2014](#)):

► Julia code (click to expand)

▼ Matlab code (click to expand)

```
ScalarField = py.importlib.import_module('PyMPDATA').ScalarField;
VectorField = py.importlib.import_module('PyMPDATA').VectorField;
Periodic = py.importlib.import_module('PyMPDATA.boundary_conditions').Periodic;

nx = int32(24);
ny = int32(24);

Cx = -.5;
Cy = -.25;

[xi, yi] = meshgrid(double(0:1:nx-1), double(0:1:ny-1));

halo = options.n_halo;
advectionee = ScalarField(pyargs(...
    'data', py.numpy.array(exp( ...
        -(xi*.5*double(nx)/2).^2 / (2*(double(nx)/10)^2) ...
        -(yi*.5*double(ny)/2).^2 / (2*(double(ny)/10)^2) ...
    )));
    'halo', halo, ...
    'boundary_conditions', py.tuple({Periodic(), Periodic()}));
));
advector = VectorField(pyargs(...
    'data', py.tuple({ ...
        Cx * py.numpy.ones(int32([nx*1 ny])), ...
        Cy * py.numpy.ones(int32([nx ny*1])) ...
    });
    'halo', halo, ...
    'boundary_conditions', py.tuple({Periodic(), Periodic()}));
));
```

► Rust code (click to expand)

▼ Python code (click to expand)

code contributors (CS, math & physics students):

Jakub Banaśkiewicz (UJ), **Piotr Bartman** (UJ), Kacper Derlatka (UJ, Pega), Szymon Drenda (UJ),
Adrian Jaśkowiec (AGH), Piotr Karaś (AGH), Norbert Klockiewicz (AGH), Michał Kowalczyk (AGH),
Kacper Majchrzak (AGH), Paweł Magnuszewski (AGH), Maciej Manna (UJ, Autodesk), Wojciech Neuman (AGH),
Michael Olesik (UJ), Arkadiusz Paterak (AGH), Paulina Pojda (AGH), Wiktor Prosowicz (AGH),
Weronika Romaniec (AGH), Paweł Rozwoda (UJ), Michał Sadowski (UJ), Jan Stryszewski (AGH),
Michał Szczygieł (AGH), Michał Wroński (AGH), Joanna Wójcicka (AGH), Antoni Zięciak (AGH),
Agnieszka Żaba (AGH), **new contributors very welcome!**

code contributors (CS, math & physics students):

Jakub Banaśkiewicz (UJ), **Piotr Bartman** (UJ), Kacper Derlatka (UJ, Pega), Szymon Drenda (UJ),
Adrian Jaśkowiec (AGH), Piotr Karaś (AGH), Norbert Klockiewicz (AGH), Michał Kowalczyk (AGH),
Kacper Majchrzak (AGH), Paweł Magnuszewski (AGH), Maciej Manna (UJ, Autodesk), Wojciech Neuman (AGH),
Michael Olesik (UJ), Arkadiusz Paterak (AGH), Paulina Pojda (AGH), Wiktor Prosowicz (AGH),
Weronika Romaniec (AGH), Paweł Rozwoda (UJ), Michał Sadowski (UJ), Jan Stryszewski (AGH),
Michał Szczęgieł (AGH), Michał Wroński (AGH), Joanna Wójcicka (AGH), Antoni Zięciak (AGH),
Agnieszka Żaba (AGH), **new contributors very welcome!**



code contributors (CS, math & physics students):

Jakub Banaśkiewicz (UJ), Piotr Bartman (UJ), Kacper Derlatka (UJ, Pega), Szymon Drenda (UJ),
Adrian Jaśkowiec (AGH), Piotr Karaś (AGH), Norbert Klockiewicz (AGH), Michał Kowalczyk (AGH),
Kacper Majchrzak (AGH), Paweł Magnuszewski (AGH), Maciej Manna (UJ, Autodesk), Wojciech Neuman (AGH),
Michael Olesik (UJ), Arkadiusz Paterak (AGH), Paulina Pojda (AGH), Wiktor Prosowicz (AGH),
Weronika Romaniec (AGH), Paweł Rozwoda (UJ), Michał Sadowski (UJ), Jan Stryszewski (AGH),
Michał Szczęgieł (AGH), Michał Wroński (AGH), Joanna Wójcicka (AGH), Antoni Zięciak (AGH),
Agnieszka Żaba (AGH), **new contributors very welcome!**



Foundation for
Polish Science



NATIONAL SCIENCE CENTRE
POLAND

6-month postdoc position at AGH available

code contributors (CS, math & physics students):

Jakub Banaśkiewicz (UJ), Piotr Bartman (UJ), Kacper Derlatka (UJ, Pega), Szymon Drenda (UJ),
Adrian Jaśkowiec (AGH), Piotr Karaś (AGH), Norbert Klockiewicz (AGH), Michał Kowalczyk (AGH),
Kacper Majchrzak (AGH), Paweł Magnuszewski (AGH), Maciej Manna (UJ, Autodesk), Wojciech Neuman (AGH),
Michael Olesik (UJ), Arkadiusz Paterak (AGH), Paulina Pojda (AGH), Wiktor Prosowicz (AGH),
Weronika Romaniec (AGH), Paweł Rozwoda (UJ), Michał Sadowski (UJ), Jan Stryszewski (AGH),
Michał Szczygieł (AGH), Michał Wroński (AGH), Joanna Wójcicka (AGH), Antoni Zięciak (AGH),
Agnieszka Żaba (AGH), **new contributors very welcome!**



Foundation for
Polish Science



NATIONAL SCIENCE CENTRE
POLAND

6-month postdoc position at AGH available
AMS Annual Meeting @Houston (25-29 Jan 2026) session



18th Symposium on Aerosol-Cloud-Climate Interactions

Third Symposium on Cloud Physics

Abstracts are due by **14 August 2025 at 5:00 PM ET**

[SUBMIT ABSTRACT](#)

Joint Sessions

- Advances in numerical modeling of aerosol-cloud interactions: moment-, bin- and particle-resolved methods and beyond

code contributors (CS, math & physics students):

Jakub Banaśkiewicz (UJ), Piotr Bartman (UJ), Kacper Derlatka (UJ, Pega), Szymon Drenda (UJ),
Adrian Jaśkowiec (AGH), Piotr Karaś (AGH), Norbert Klockiewicz (AGH), Michał Kowalczyk (AGH),
Kacper Majchrzak (AGH), Paweł Magnuszewski (AGH), Maciej Manna (UJ, Autodesk), Wojciech Neuman (AGH),
Michael Olesik (UJ), Arkadiusz Paterak (AGH), Paulina Pojda (AGH), Wiktor Prosowicz (AGH),
Weronika Romaniec (AGH), Paweł Rozwoda (UJ), Michał Sadowski (UJ), Jan Stryszewski (AGH),
Michał Szczęgieł (AGH), Michał Wroński (AGH), Joanna Wójcicka (AGH), Antoni Zięciak (AGH),
Agnieszka Żaba (AGH), **new contributors very welcome!**



Foundation for
Polish Science



NATIONAL SCIENCE CENTRE
POLAND

6-month postdoc position at AGH available
AMS Annual Meeting @Houston (25-29 Jan 2026) session

Thank you for your attention!

sylwester.arabas@agh.edu.pl

MPDATA Wikipedia article: contributions welcome!

<https://en.wikipedia.org/wiki/Draft:MPDATA>

Contents hide

(Top)

Description of the basic scheme in 1D

Minimal implementation and convergence analysis in Python

Algorithm variants and techniques used in concert with MPDATA

Algorithm steps (shallow-water system example)

Applications

Open-source implementations

References

Description of the basic scheme in 1D [edit]

MPDATA is inherently multi-dimensional, and primarily used in [computational fluid dynamics](#) where the advective velocities and problem geometries are variable in time. Still, the key idea underlying the MPDATA approach can be conveyed with a basic example of [solenoidal stationary flow](#) in one dimension^[11] (i.e., $\mathbf{v} = [u]$ constant in time and space), without coordinate transformation ($G = 1$), for the case of [homogeneous advection](#) ($R = 0$) of a nonnegative scalar field ($\psi \geq 0$), with the following flux form of the [advection equation](#):

$$\partial_t \psi + \partial_x(u\psi) = 0. \quad (2)$$

[Upwind discretisation](#) of the problem on a [regular staggered grid](#) with a time step Δt and a grid step Δx , with $n = t/\Delta t$, $i = x/\Delta x$, and the half-integer spatial indices corresponding to grid-cell walls:



can be formulated with:

$$\frac{\psi_i^{n+1} - \psi_i^n}{\Delta t} + \frac{\overbrace{f(\psi_i^n, \psi_{i+1}^n, u_{i+1/2}^n)}^{\text{right-hand wall flux}} - \overbrace{f(\psi_{i-1}^n, \psi_i^n, u_{i-1/2}^n)}^{\text{left-hand wall flux}}}{\Delta x} = 0 \quad (3)$$

with the [flux](#) function defined using [positive](#) and [negative parts](#) of $u_{i\pm 1/2}$ as:

$$f(\psi_l, \psi_r, u) = \underbrace{\frac{u + |u|}{2}}_{\text{positive part}} \psi_l + \underbrace{\frac{u - |u|}{2}}_{\text{negative part}} \psi_r. \quad (4)$$

Introducing the [non-dimensional Courant number](#) $C = u\Delta t/\Delta x$, the resultant [explicit-in-time](#) scheme (referred to as "upwind", "upstream" or "donor-cell"), for a constant C reads: