Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie

AGH University of Krakow

Comparison of PyMPDATA solution against py-pde

Jakub Kot
Piotr Karaś
Norbert Klockiewisz
Kacper Majchrzak

Introduction & Motivation

AGH

www.agh.edu.pl

What's the project?

- Numerically solve a 2D diffusion equation using two Python libraries:
  - `PyMPDATA`
  - `py-pde`
- Problem from `py-pde` gallery: diffusion on a Cartesian grid.
- Develop a Jupyter notebook for side-by-side simulations.
- Compare solutions, performance, and ease of use.

Why this comparison?

- Understand strengths/weaknesses of different Python PDE solvers.
- Evaluate suitability for specific problem types.
- Assess factors:
  - Implementation complexity
  - Computational speed
  - Accuracy

Introducing py-pde

Overview:

- Python package for solving partial differential equations (PDEs).
- Focus: ease of use for exploring PDE behavior (time-evolution).
- Method: method of lines with finite-difference approximations.

Key Features:

- Various grid types: Cartesian, polar, spherical, cylindrical.
- Handles non-linear PDEs, complex boundary conditions.
- Direct specification of evolution equations.
- Core computations JIT-compiled with Numba.

Example Problem Context:

- Solves $\partial_t c = D\nabla^2 c$, where D is diffusivity.
- Demonstrates: `CartesianGrid`, `ScalarField`, `DiffusionPDE`.

Potential Limitations: Best for simple geometries, fixed discretization.

Introducing PyMPDATA

Overview:

- High-performance Python implementation of MPDATA (Multi-dimensional Positive Definite Advection Transport Algorithm).
- For generalized convection-diffusion PDEs.

Key Features:

- Numba-accelerated for performance.
- 1D, 2D, 3D structured meshes, coordinate transformations.
- Modular: `Options`, `BoundaryCondition`, `Stepper`, `Solver`.
- `PyMPDATA-MPI` for distributed memory parallelism.

Relevance to Diffusion:

- MPDATA handles diffusion terms within its generalized convection-diffusion framework.

The Chosen Problem: 2D Diffusion

AGH

Equation:

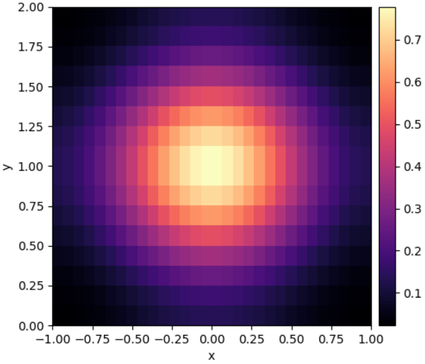
$$\partial_t c = D\nabla^2 c$$

- c: scalar field (e.g., concentration)
- D: diffusivity constant

`py-pde` **Example Setup (to be replicated):**

- Grid: `CartesianGrid([[-1, 1], [-0.5, 0.5]], [30, 16])`
- Initial Condition: Scalar field with value 1 at center.
- PDE: `DiffusionPDE(diffusivity=0.1)`
- Simulation: `t_range=1`, `dt=0.01`

Visualization from `py-pde`:



-
- Expected evolution: a peak diffusing outwards.

Methodology for Comparison

Implementation in Jupyter Notebook:

- `py-pde`: Re-implement the gallery example.
- `PyMPDATA`:
  - Set up equivalent grid and initial scalar field.
  - Configure `Solver` for diffusion (e.g., zero advection velocity).
  - Define matching boundary conditions.

Simulation Parameters (Identical):

- Grid dimensions, resolution, initial conditions, diffusivity, total time, time step (or equivalent stability-controlled).

Comparison Metrics:

- **Visual:** Plot final states, time-evolution snapshots.
- **Quantitative (if feasible):** L_2 norm of difference, mass conservation.
- **Performance:** Wall-clock time.
- **Qualitative:** Ease of setup, code verbosity, API clarity.

Expected Outcomes & Discussion Points

- **Solution Agreement:** How closely do numerical solutions match?
- **Performance:**
  - How does `PyMPDATA`'s performance (Numba, advection-focused design) compare to `py-pde` (Numba, general) on pure diffusion?
  - Is MPDATA's machinery an overhead for pure diffusion?
- **Usability:**
  - Intuition in setting up a simple diffusion problem?
  - Impact of `PyMPDATA`'s geophysical focus vs. `py-pde`'s general nature.
- **Potential Challenges:**
  - Ensuring equivalent boundary conditions.
  - Mapping general PDE to `PyMPDATA`'s specific structure for diffusion.

Conclusion & Next Steps

Summary:

- Goal: A fair comparison of two powerful Python PDE solvers on a specific diffusion problem.

Future Work / Next Steps:

- Implementation in PyMPDATA.
- Prepare test case scenario.

Thank You! Questions?