Imperial College London

Massachusetts Institute of Technology

Introducing Students to Open-Source Research Codes

Solving PDEs in Python

Pavan Inguva, Vijesh Bhute, Pierre Walker, Thomas Cheng

Imperial College London

Acknowledgements

Dr Vijesh Bhute
Teaching Fellow,
Imperial College

Mr Pierre Walker
Graduate Student,
Caltech

Mr Thomas Cheng
Undergraduate,
Imperial College

Imperial College London

Criteria to Evaluate Computational Tools

1. Availability: Are they free/low cost and available to users readily?
2. Ease of use: Is there a steep learning curve / are resources available?
3. Scalability: Can these tools be used for both small tasks on a laptop and large tasks on a HPC cluster?
4. Learning Objectives: Is the course focused on theory or application?

Imperial College London

PDE Solver Landscape

Commercial Codes

- Often used in teaching and professional settings
- Extensive resources available
- Very expensive

Open-Source (C++ etc.)

- Well-established in professional/research uses
- Limited resources available
- Steep learning curve
- Free

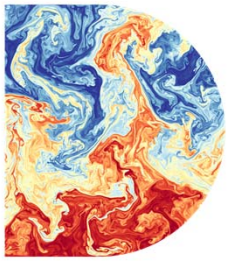
DIY Codes

- Typically simple finite difference codes used for teaching purposes
- Often scales poorly
- Bespoke codes that are fast and scale are very inaccessible

Imperial College London

Examples of PDE Solvers in Python

1. FEniCS: Finite Element Method (FEM)
2. FiPy: Finite Volume Method (FVM)
3. Dedalus: Spectral Methods
4. Pystencils: Finite Difference Method (FDM)



Imperial College London

Open-Source PDE Solvers in Python

- ✓ Very available due to their open-source nature
- ✓ Easy to use Python front-ends
- ✓ Often coupled to a fast backend written in C++ / compiled languages e.g., PETSc
- ✓ Many discretization methods (FVM, FEM, FDM, Spectral Methods).
- ✓ Adequate documentation and access to developers

Imperial College London

Lesson Objective

- To introduce 2nd/3rd year students to PDE solver codes
  - Provide students the foundation to use such codes in their own research/work
  - Complement the teaching of advanced engineering math / transport
  - Introduce basic concepts in numerical methods
  - Establish some familiarity with coding

Imperial College London

Coding Environment

- Anaconda was selected as the coding environment for multiple reasons
  - Convenient coding environment well suited for Windows
  - Comes with the Spyder IDE → Low barriers to entry for students
  - Good introduction to reproducible coding environments and containerization → Inculcate good practices

Imperial College London

Lesson Structure

- 4 exercises of increasing complexity are developed with progression as the guiding philosophy
- FiPy was used as the main solver code as it is very intuitive
- Extensive notes were provided for each exercise outlining the basic numerical methods concepts and mathematics (e.g. analytical solutions where available)
- Students were advised to treat the provided code as boiler plate
- Some exercises are further broken down into parts that explore specific aspects

Imperial College London

Exercises

Exercise 1: Laplace Equation

$$\nabla^2 \phi = 0$$

$$\phi(0, y) = \phi(1, y) = \phi(x, 0) = 0, \phi(x, 1) = 1$$

Objectives:

- Set up and run a simulation
- Visualize the results
- Benchmark the results to the analytical solution

Exercise 2: Diffusion Equation

$$\frac{\partial c}{\partial t} = D\nabla^2 c$$

$$c(x = 0, t) = 1, c(x = 1, t) = 0, c(x, t = 0) = 0$$

Objectives:

- Explore simple timestepping schemes
- Appreciate the features of the transient solution
- Introduce source and sink terms

Imperial College London

Exercises

Exercise 3: Advection-Diffusion

$$\frac{\partial c}{\partial \tilde{t}} + \frac{\partial c}{\partial \tilde{x}} = \frac{1}{\mathrm{Pe}} \frac{\partial^2 c}{\partial \tilde{x}^2}$$

Objectives:

- Appreciate the mechanism of advective transport
- Appreciate the issues related to the numerical solution
- Implement coupled problems

Exercise 4: Cahn-Hilliard Equation

$$\tilde{\mu}_{AB} = \frac{\partial g}{\partial a} - \tilde{\kappa}\tilde{\nabla}^2 a$$

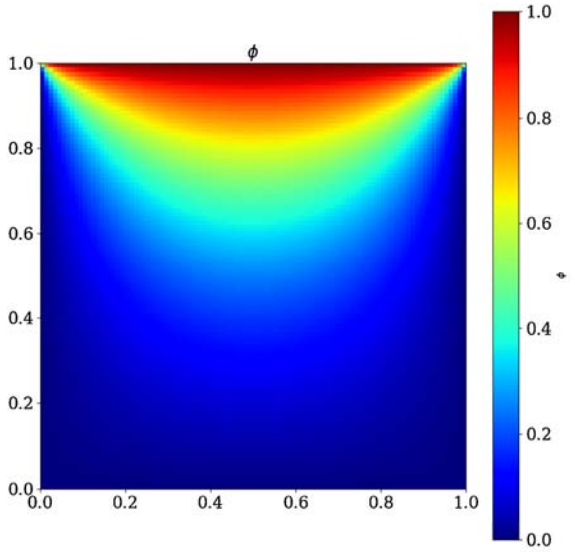
$$\frac{\partial a}{\partial \tilde{t}} = \tilde{\nabla} \cdot (a(1-a)\tilde{\nabla}\tilde{\mu}_{AB})$$

Objectives:

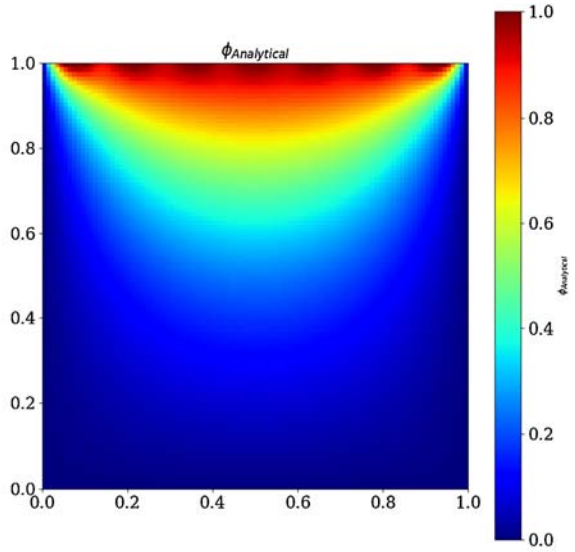
- Explore an advanced problem
- Advanced postprocessing in ParaView

Imperial College London

Exemplar Results



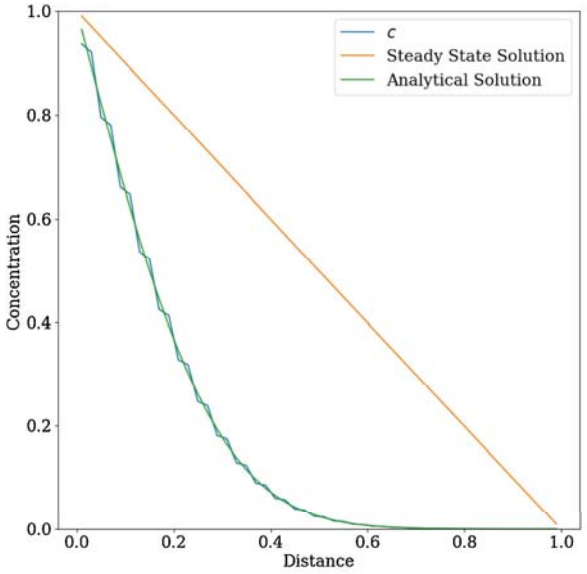
(a) Numerical solution



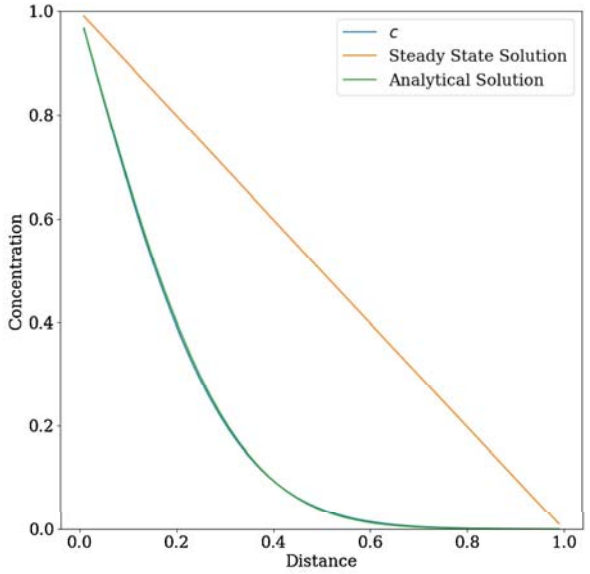
(b) Analytical solution truncated to the 7th term

Imperial College London

Exemplar Results



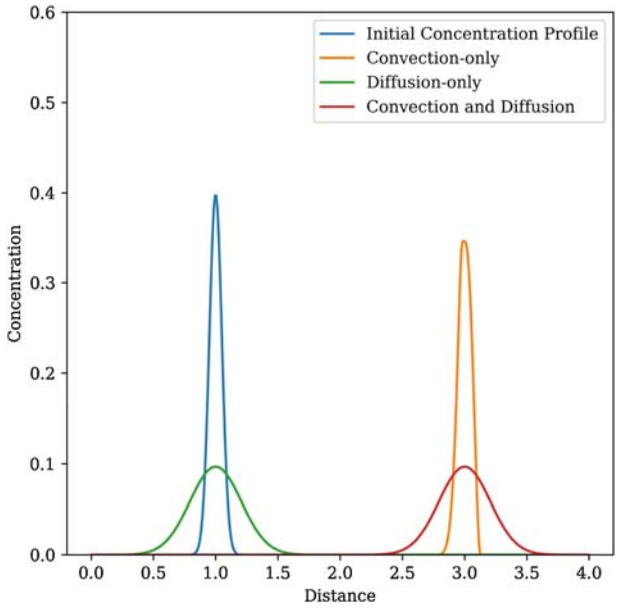
(a) Explicit forward Euler time-stepping with $\Delta t = 0.0001998$

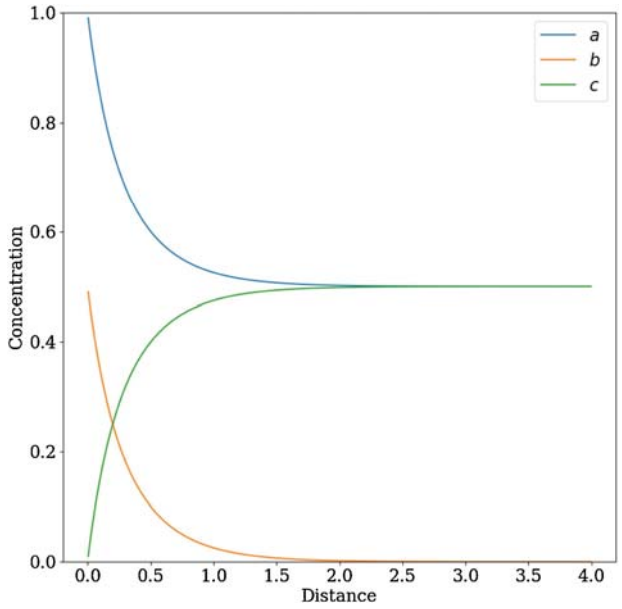


(b) Implicit backward Euler time-stepping with $\Delta t = 0.0018$

Imperial College London

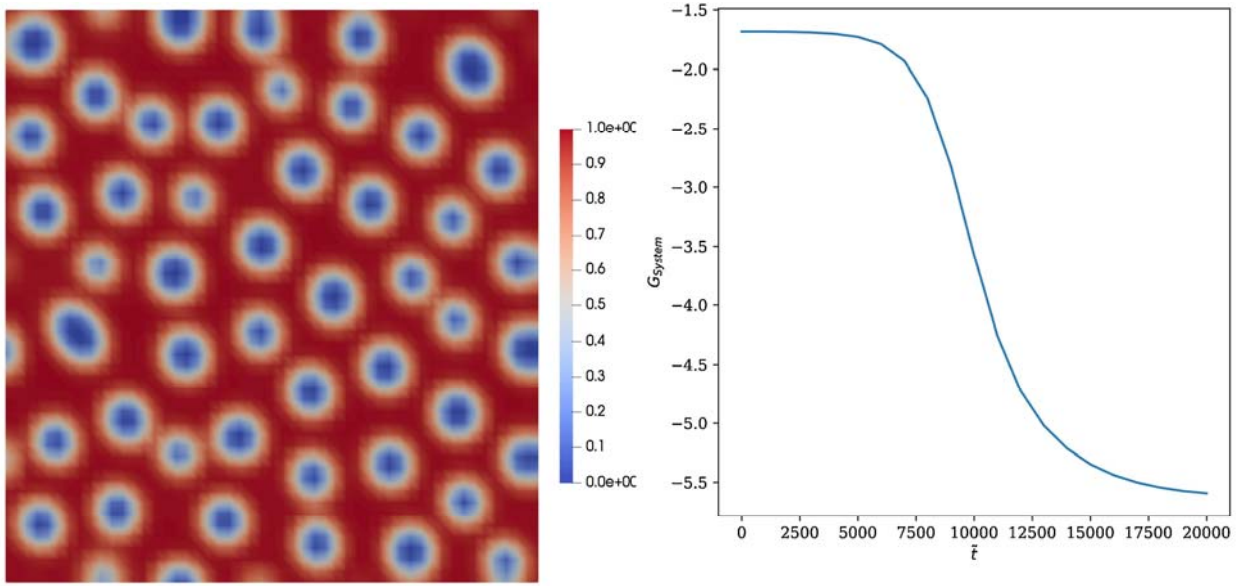
Exemplar Results





Imperial College London

Exemplar Results



(a) Polymer blend morphology @ $\tilde{t}$ = 20000. The volume fraction colour bar is included for reference.

(b) Evolution of G_{System} over time, demonstrating how the demixing process decreases the total Gibbs energy.

Imperial College London

Student Perception and Feedback

Students valued the visualizations of various mathematical concepts made possible:

> "*I felt the course explained PDEs better than we got taught it, because you can see it as it was happening and see how one thing affects the other. The graphs gave a more visual approach than when it was taught*..."

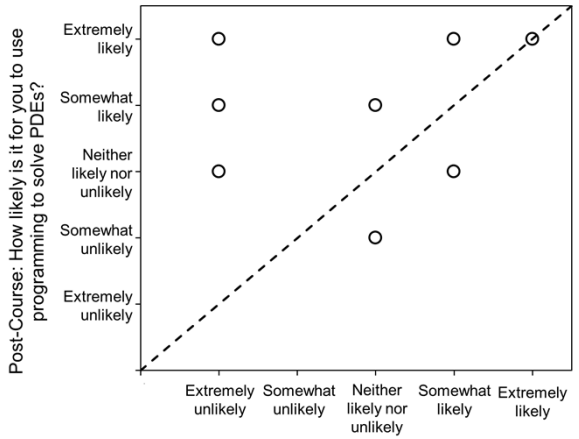
Students also indicated they would like to know more about the mechanics of FiPy and Python:

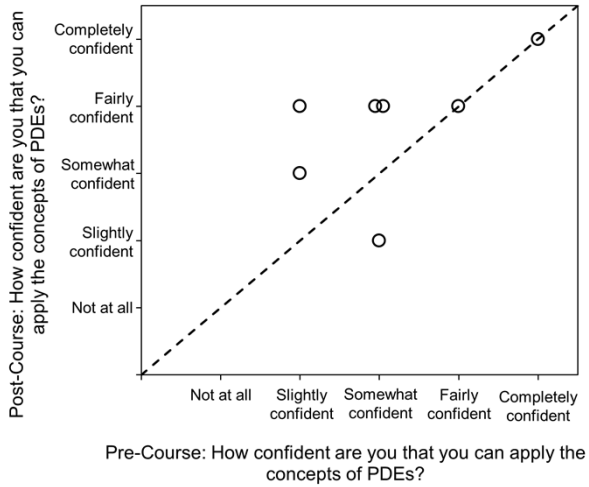
> "*I was more thinking in the future. For example, if I want to use this myself to solve some problems. Then it would then be nice to also be introduced to how this whole thing is structured.*"

Imperial College London

Student Perception and Feedback

The short course has also helped to improve students' self efficacy in programming and PDEs





Imperial College London

Conclusions

- A short course successfully introducing students to PDE solvers was developed.
- Student feedback indicated they had an improved perception of Python and increased self-efficacy in advanced engineering math.
- Educators can employ these tools to enhance their teaching
- The course material is available at https://github.com/pavaninguva/pde-Solver-Course