Imperial College London

16/09/2022

CP123: Solving PDEs with AI Libraries

Supervisors: Christopher Pain, Claire Heaney, Chen Boyang

Chukwume Ijeh

Problem Statement

Due to:

- Complexity and computational costs associated with writing and running computational fluid dynamics (CFD) codes, and
- The recent advent of AI computers

We want to:

- Simplify CFD codes using an AI library with the main aim of running such codes on AI computers

Main Aims and Objectives

- Solve CFD problems on unstructured meshes using PyTorch [1],
- Solve the 2D advection-diffusion equation on unstructured meshes, using a graph neural network and a multigrid solver.

Methodology

- Graph Neural Network (GNN)
  - Use of graph data [2]
- Multigrid Solver
  - Two orthogonal space-filling curves (SFCs) [3]
  - Efficient and versatile approach [4]

Scalar transport equation:

$$\frac{\partial T}{\partial t} + u\frac{\partial T}{\partial x} + v\frac{\partial T}{\partial y} + \sigma T + \kappa\nabla^2 T = s,$$

where:

- T is the scalar concentration field,
- t is the time in seconds (s),
- u and v are the velocities (ms^{-1}) in the x and y directions respectively,
- σ is the absorption term,
- κ is the diffusivity (m^2s^{-1}), and
- s is the source.

GNN - Architecture

- Production of the shape functions using the finite element (FE) discretization method [5].
- Tested using the following variables:
  i. uniform velocity in the *x* and *y* directions, $(u, v) = (1, 1) ms^{-1}$
  ii. diffusivity, $\kappa = 0.2 m^2 s^{-1}$
  iii. source, $s = 0$
  iv. absorption term $\sigma = 0$
  v. change in *x* and *y*, $d_x = d_y = 1$
  vi. time-step, $d_t = 0.1s$

$$T^{n+1} = M_L^{-1}(M_L - Ad_t) \times T^n$$

where:
T^{n+1} is the scalar field at next time-step
M_L is the lumped mass matrix
A is the 'stiffness' matrix
d_t is the change in time
T^n is the scalar field at current time-step

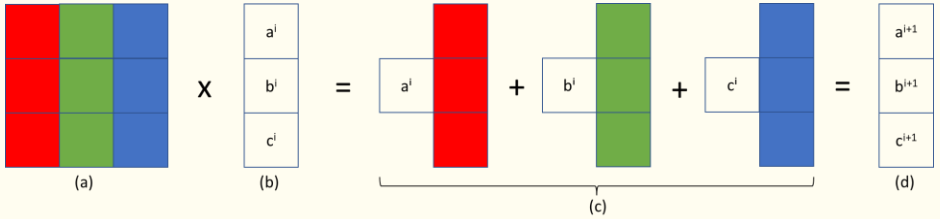
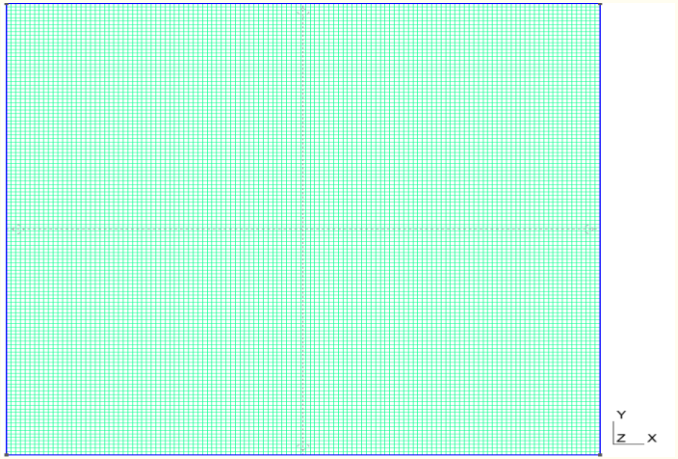


Fig 1: Simple architecture of the GNN with (b) being the scalar field at current timestep, (a) being the matrix of weights. The diagram shows a depiction of a matrix-vector multiplication where (d) is the scalar field at the next timestep

GNN- Results on a Structured Mesh



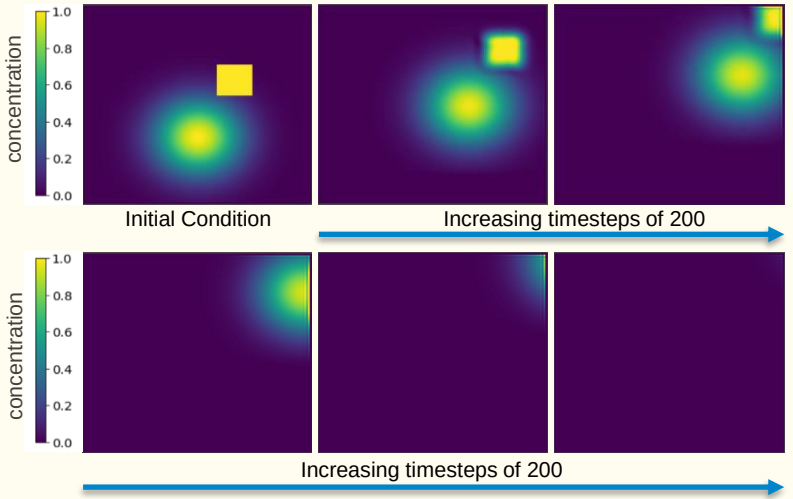


Fig 2: **Left:** 128 x 128 structured rectangular mesh (generated using GMSH [6]) used to solve the 2D advection-diffusion equation of a Gaussian and square distribution. **Right**: The system state from an initial condition (first image) to the system after 200 timesteps (second image), and another 200 timesteps (third image), and so on.

GNN- Results on an Unstructured Mesh

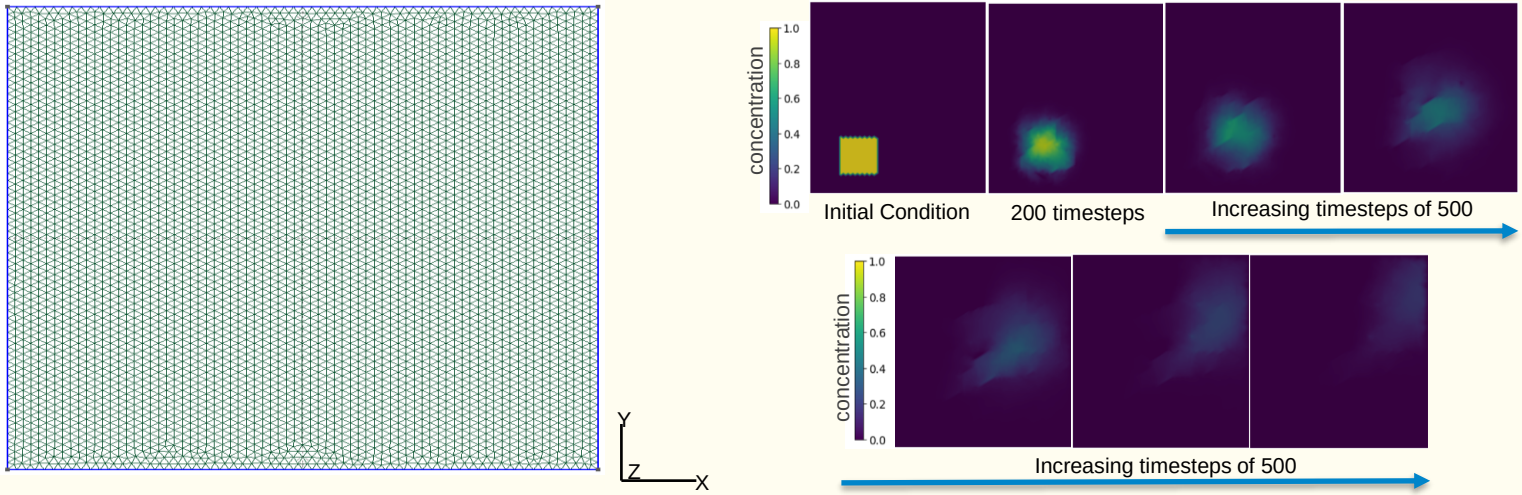


Fig 3: **Left**: 64 x 64 unstructured triangular mesh (generated using GMSH [6]) used to solve the 2D advection-diffusion equation of a rectangular distribution. **Right**: The system state from an initial condition (first image) to the system after 200 timesteps (second image), and then 500 timesteps (third image), till the last image.

Multigrid Solver - Architecture

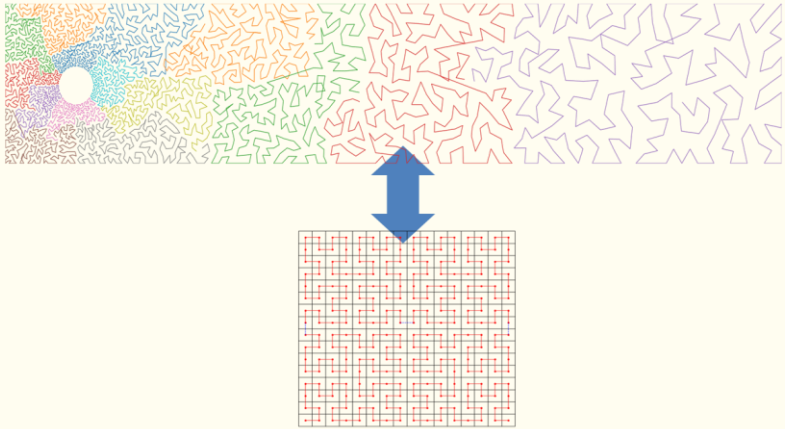


Fig 4: The use of space-filling curves (bottom) to transform 2D unstructured data for flow past a cylinder to 1D (a)

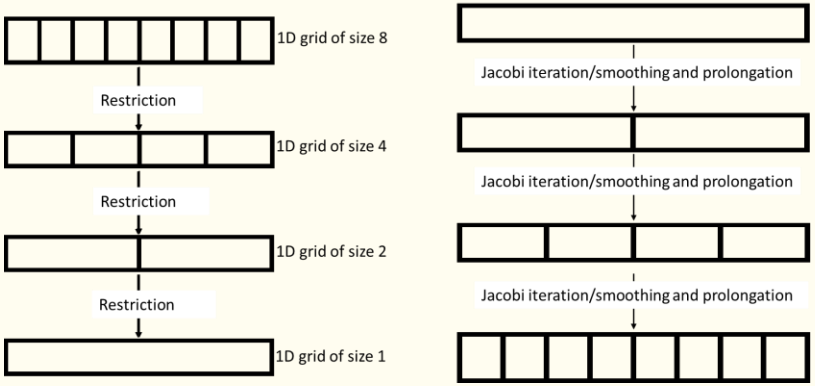


Fig 5: Simple depiction of the restriction/prolongation/smoothing done on each level of the multigrid

Multigrid Solver - Architecture

- FEM discretization to find *A*, M_L, and *b*.
- Tested using the following variables:
  i. Uniform velocity in the *x* and *y* directions, $(u, v) = (0, 0) ms^{-1}$,
  ii. Diffusivity, $\kappa = 0.2 m^2 s^{-1}$,
  iii. Source, $s = 0$, with a point source of *unity* at the middle of the domain,
  iv. absorption term $\sigma = 0$,
  v. Jacobi relaxation coefficient, $\alpha = 1/3$,
  vi. *relax_keep_off* = 0.5,
  vii. filter size, *nfilt_size_sfc=3.*

$$x^{(k+1)} = x^{(k)} - \alpha A D^{-1} x^{(k)} + \alpha D^{-1} b,$$

where:

- *A* is the 'stiffness' matrix in SFC ordering per layer,
- $x^{(k)}$ is the current correction,
- *b* is the solution vector in SFC ordering per layer,
- $x^{(k+1)}$ is the next correction of the equation,
- α is the relaxation coefficient,
- *D is* the diagonal of the matrix *A*

Multigrid Solver – Results on a Structured Mesh

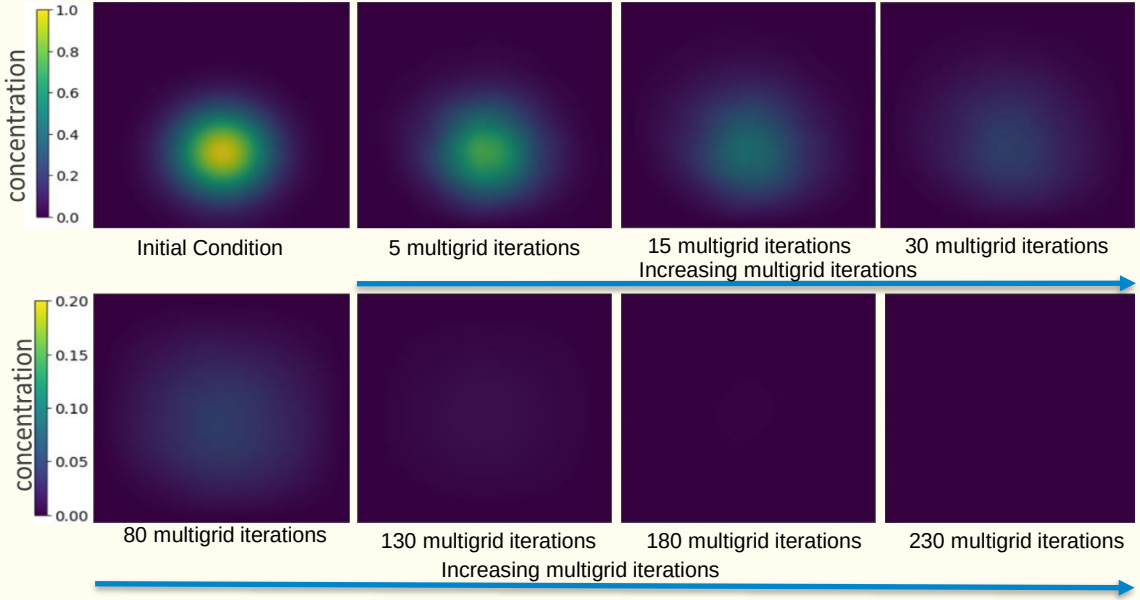


Fig 6: The system state from an initial condition (first image) to the system after 5, 15, 30, 80, 130, 180, and 230 multigrid iterations. Results got on a 128 x 128 structured rectangular mesh as shown in Fig 2.

Multigrid Solver – Results on an Unstructured Mesh

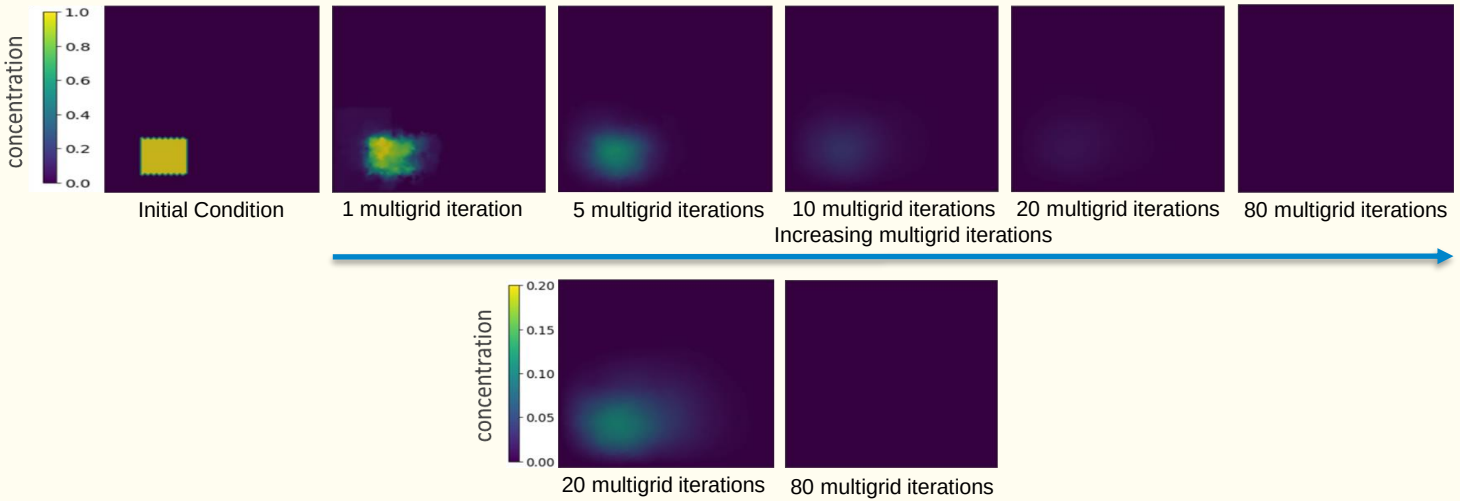


Fig 7: **Top**: The system state from an initial condition (first image) to the system after 1, 5, 10, 20, and 80 multigrid iterations. **Bottom**: System state after 20 and multigrid iterations with a concentration range of 0 - 0.2. Results got on a 64 x 64 unstructured triangular mesh as shown in Fig 3.

Perks of the Multigrid Solver

- Alternative use of two orthogonal space-filling curves (SFCs)
- Change filter size, *nfilt_size_sfc*
- Increase the number of neighbors added to diagonal/self-weight, *relax_keep_off*.

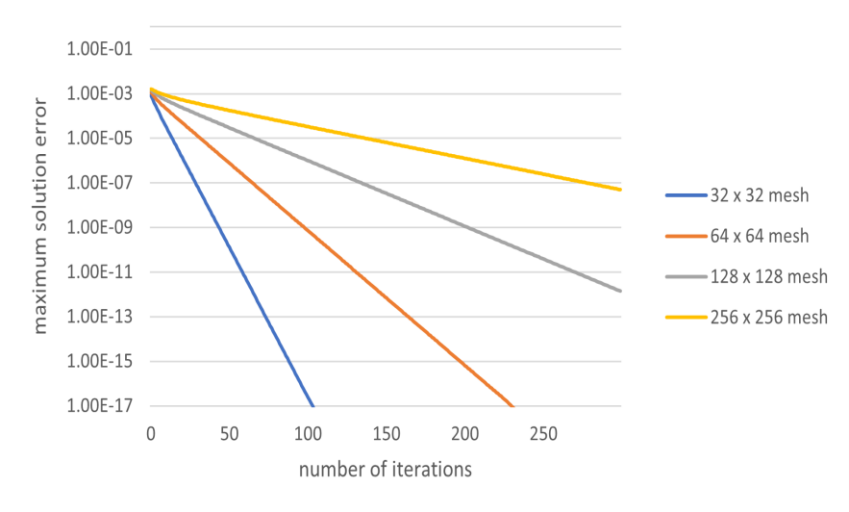


Fig 8: Graph showing the maximum error in each iteration for different problem sizes: 32 x 32, 64 x 64, 128 x 128, and 256 x 256 2D meshes using a nfilt_size_sfc of 129.

Comparison – GNN and Multigrid solver

	GNN	Multigrid Solver
Pros	- Solve the 2D advection-diffusion problem - Relatively accurate	- Fast - Garner accuracy - Solve other partial differential equations.
Drawbacks	- Unable to solve a pure advection case, due to Gibb's oscillations - Slow	- Unable to solve both a pure advection and an advection-diffusion case

Table 1: Pros and Drawbacks of the graph neural network (GNN) and the multigrid solver

S/N	Grid Size	Number of nodes	Time in seconds (s) to reach precision of 10^{-3}	
			Methods	
			Graph Neural Network (GNN)	Multigrid Solver
1.	128x128	16641	>100000	2045
2.	64x64	4095	1534	31.2
3.	32x32	1296	199	4.42

Table 2: Comparison of Computational Time between the graph neural network (GNN) and multigrid solver on different mesh sizes

Conclusion

From the results, there is great promise in using an AI library to solve CFD problems. Future work should include testing code on an actual CFD problem, such as ‘flow past a bluff body’, and attempting to run associated code on an AI computer.

References

[1] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019

[2] Renzo Angles and Claudio Gutierrez. An introduction to graph data management. In Graph Data Management, pages 1–32. Springer, 2018.

[3] Claire E Heaney, Yuling Li, Omar K Matar, and Christopher C Pain. Applying convolutional neural networks to data on unstructured meshes with space-filling curves. arXiv preprint arXiv:2011.14820, 2020.

[4] Irad Yavneh. Why multigrid methods are so efficient. Computing in science & engineering, 8(6):12–22, 2006.

[5] Olek C Zienkiewicz, Robert Leroy Taylor, and Jian Z Zhu. The finite element method: its basis and fundamentals. Elsevier, 2005.

[6] Christophe Geuzaine and J-F Remacle. Gmsh: a three-dimensional finite element mesh generator with built-in pre-and post-processing facilities, 2008.

Imperial College London

16/09/2022