Optimizing a DIscrete Loss (ODIL) to solve forward and inverse problems for partial differential equations using machine learning tools

Petr Karnakov

Postdoctoral Fellow, Koumoutsakos Group

May 11, 2022

arXiv:2205.04611

With:

Sergey Litvinov

Petros Koumoutsakos

Neural networks for PDEs

Collocation method for PDEs

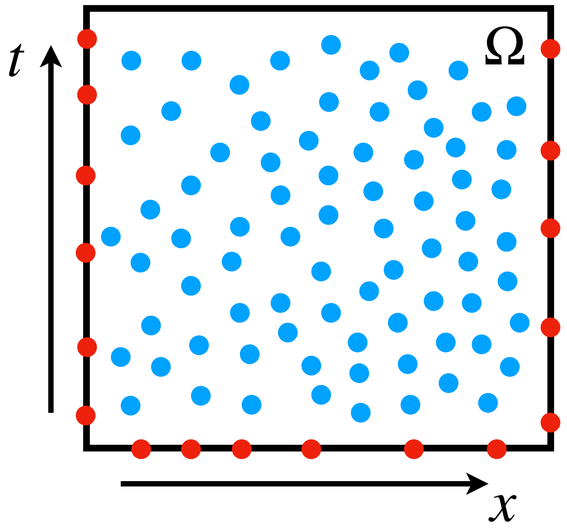
- Equation with boundary conditions

$$G[u] = 0, \quad (x,t) \in \Omega$$
$$u = g, \quad (x,t) \in \delta\Omega$$

- Parameterized solution $u(x,t,\theta)$
  - smooth function of x, t
  - parameters θ
  - polynomial, spline, etc

- Loss function from discrete points

$$L(\theta) = \underbrace{\sum_{(x,t)\in\Omega_1} \big(G[u](x,t,\theta)\big)^2}_{\text{equation at collocation points } \Omega_1 \subset \Omega} + \underbrace{\sum_{(x,t)\in\Omega_2} \big(u(x,t,\theta) - g(x,t)\big)^2}_{\text{boundary conditions at boundary points } \Omega_2 \subset \partial\Omega} \to \min_\theta$$

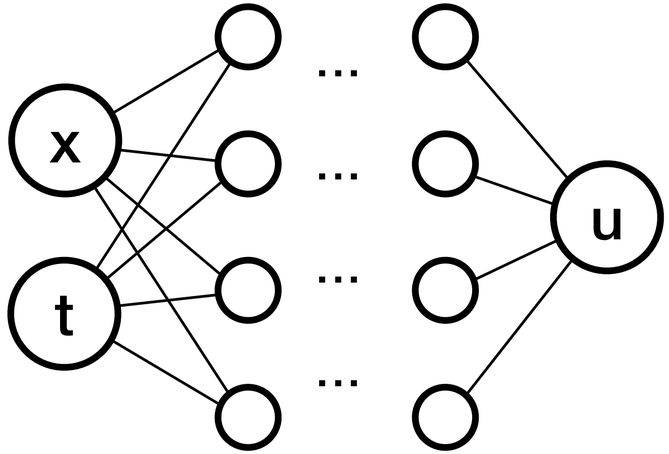


Neural networks for PDEs

- Equation with boundary conditions

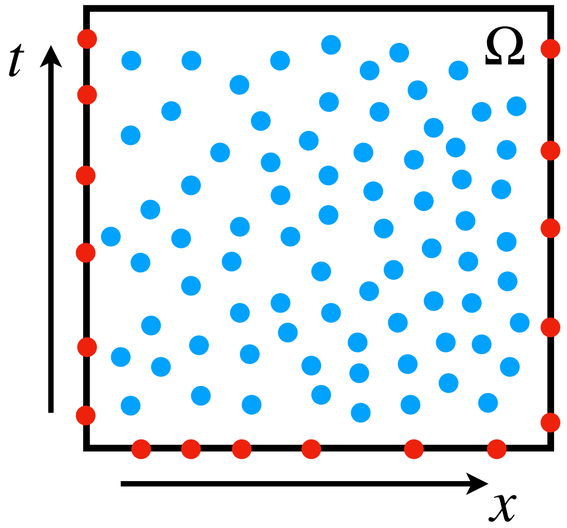
$$G[u] = 0, \quad (x,t) \in \Omega$$
$$u = g, \quad (x,t) \in \delta\Omega$$

- Solution is neural network $u(x,t,\theta)$
  - sequence of matrix multiplications, additions, element-wise nonlinear functions
  - weights θ



- Loss function

$$L(\theta) = \sum_{(x,t)\in\Omega_1} \Big(G[u](x,t,\theta)\Big)^2 + \sum_{(x,t)\in\Omega_2} \Big(u(x,t,\theta) - g(x,t)\Big)^2 \to \min_{\theta}$$



Neural networks for PDEs

Journal of Computational Physics 378 (2019) 686–707

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations

M. Raissi [a], P. Perdikaris [b,*], G.E. Karniadakis [a]

[a] Division of Applied Mathematics, Brown University, Providence, RI, 02912, USA
[b] Department of Mechanical Engineering and Applied Mechanics, University of Pennsylvania, Philadelphia, PA, 19104, USA

IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 9, NO. 5, SEPTEMBER 1998 987

Artificial Neural Networks for Solving Ordinary and Partial Differential Equations

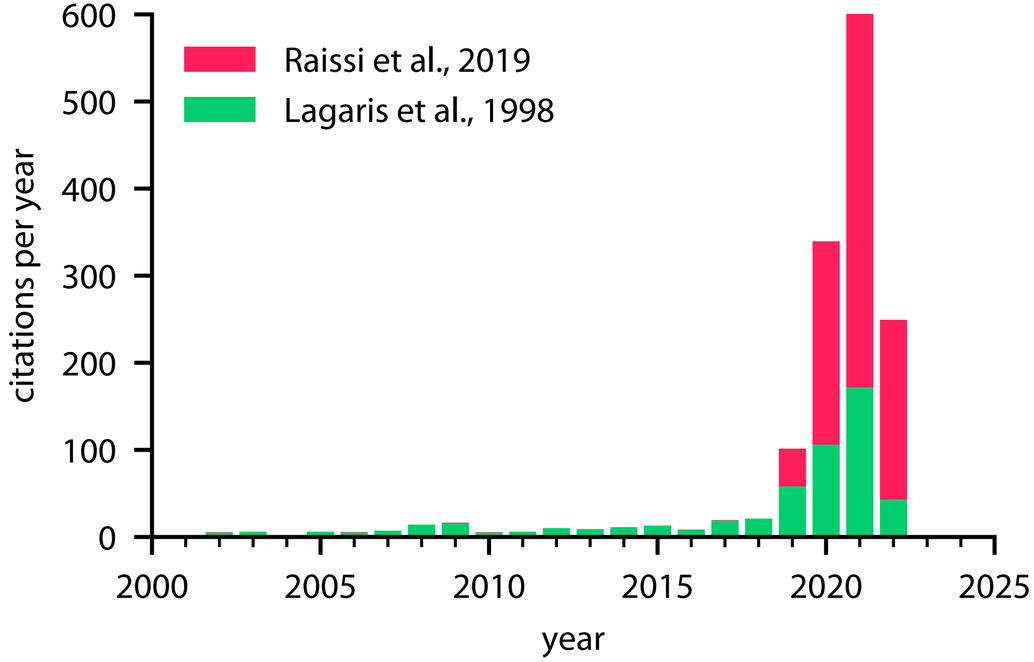
Isaac Elias Lagaris, Aristidis Likas, *Member, IEEE,* and Dimitrios I. Fotiadis

***Abstract*— We present a method to solve initial and boundary value problems using artificial neural networks. A trial solution of the differential equation is written as a sum of two parts. The first part satisfies the initial/boundary conditions and contains no adjustable parameters. The second part is constructed so as not to affect the initial/boundary conditions. This part involves a feedforward neural network containing adjustable parameters (the weights). Hence by construction the initial/boundary conditions are satisfied and the network is trained to satisfy the differential equation. The applicability of this approach ranges from single ordinary differential equations (ODE's), to systems of coupled ODE's and also to partial differential equations (PDE's).**

of a feedforward neural network by replacing each spline with the sum of piecewise linear activation functions that correspond to the hidden units. This method considers local basis-functions and in general requires many splines (and consequently network parameters) in order to yield accurate solutions. Furthermore, it is not easy to extend these techniques to multidimensional domains.

In this article we view the problem from a different angle. We present a method for solving both ordinary differential equations (ODE's) and partial differential equations (PDE's)

PINN: Physics-informed neural networks

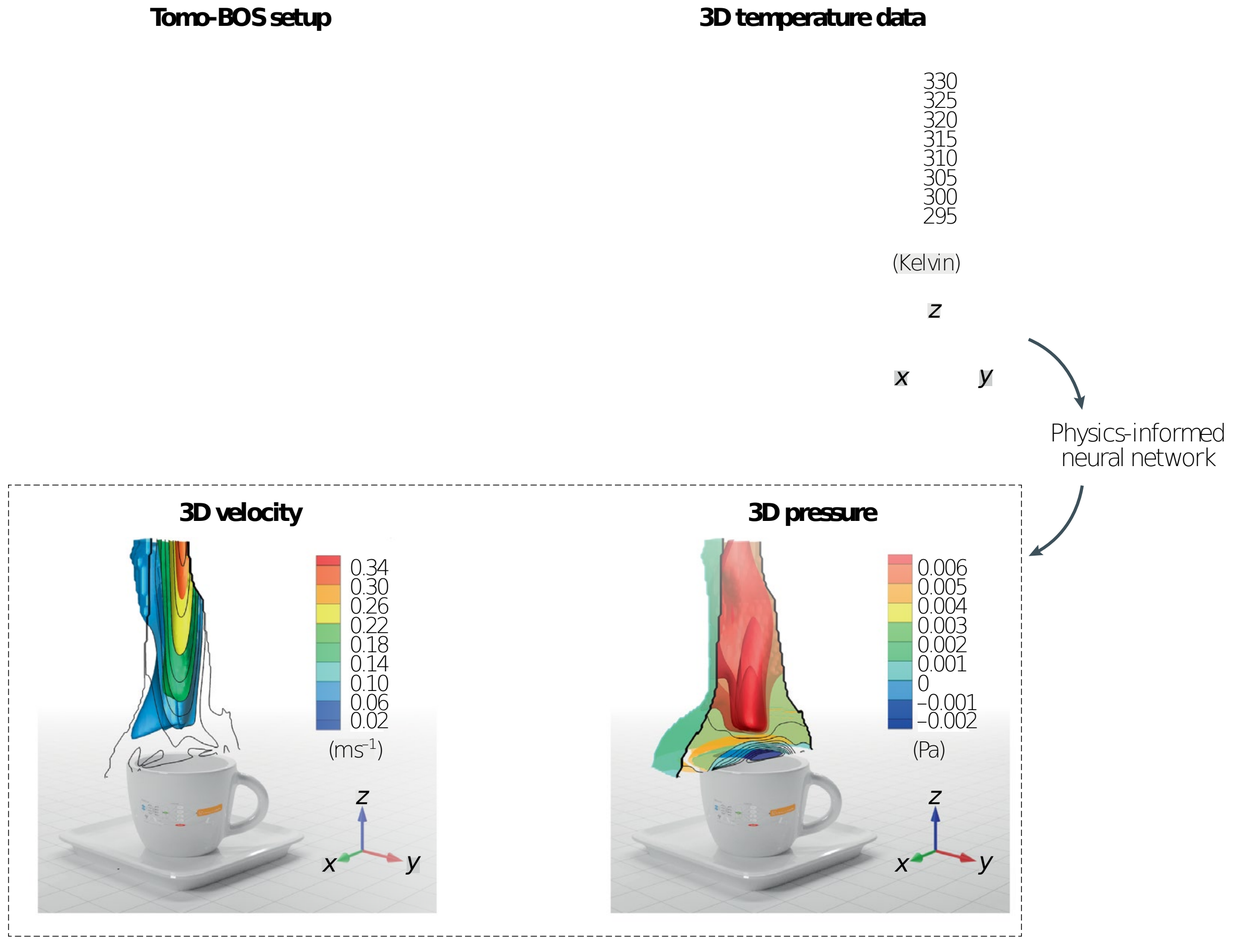


source: Web of Science

Why use PINNs?

- Inverse and ill-posed problems
- Loss function may include equations for all fields, and **omit** unknown data

$$L(\theta) = \sum_{(x,t)\in\Omega_1} \left(F_1[u,v](x,t,\theta)\right)^2 + \sum_{(x,t)\in\Omega_1} \left(F_2[u,v](x,t,\theta)\right)^2 + \sum_{(x,t)\in\Omega_2} \left(u(x,t,\theta) - u_{\text{ref}}(x,t)\right)^2$$

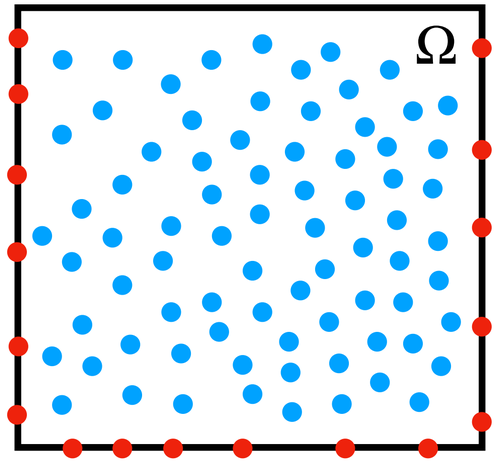


Cai S, Wang Z, Fuest F, Jeon YJ, Gray C, Karniadakis GE.
Flow over an espresso cup: inferring 3-D velocity and pressure fields from tomographic background oriented Schlieren via physics-informed neural networks.
Journal of Fluid Mechanics. 2021

Proposed: Optimizing a DIscrete Loss (ODIL)

arXiv:2205.04611

- Continuous problem $\quad \frac{\partial u}{\partial t} - k\frac{\partial^2 u}{\partial x^2} = 0\Big|_{\Omega} \qquad u = g\Big|_{\partial\Omega}$

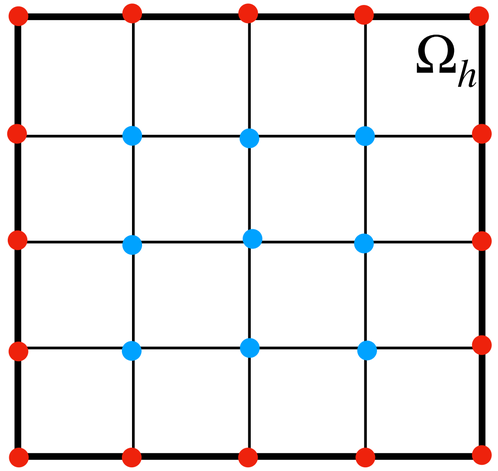


- PINN, solution is neural network $u(x,t,\theta)$

$$L(\theta) = \sum_{(x,t)\in\Omega_1} \left(\frac{\partial u}{\partial t} - k\frac{\partial^2 u}{\partial x^2}\right)^2 + \sum_{(x,t)\in\Omega_2} \left(u(x,t,\theta) - g(x,t)\right)^2 \to \min_{\theta}$$

- FD, solution is discrete field u_i^n

$$L(u) = \sum_{(i,n)\in\Omega_h} \left(\frac{u_i^{n+1} - u_i^n}{\Delta t} - k\frac{u_{i-1}^n - 2u_i^n + u_{i+1}^n}{h^2}\right)^2 + \sum_{(i,n)\in\partial\Omega_h} \left(u_i^n - g_i^n\right)^2 \to \min_{u}$$



  - General optimizer (gradient descent, quasi-Newton)

$$L(u) = \left(Au - b\right)^2 + (u-g)^T\Lambda(u-g) \to \min_{u}$$

  - Newton with sparse linear system

$$\frac{\partial L}{\partial u} = 0 \;\Leftrightarrow\; (A^TA + \Lambda)u - A^Tb - \Lambda g = 0 \qquad (\Lambda u)_i = \begin{cases} u_i & i \in \partial\Omega_h \\ 0 & \text{else} \end{cases}$$

Performance of optimization algorithms

Optimizers: algebraic equation

$$\mathbf{F}(\mathbf{x}) = 0$$

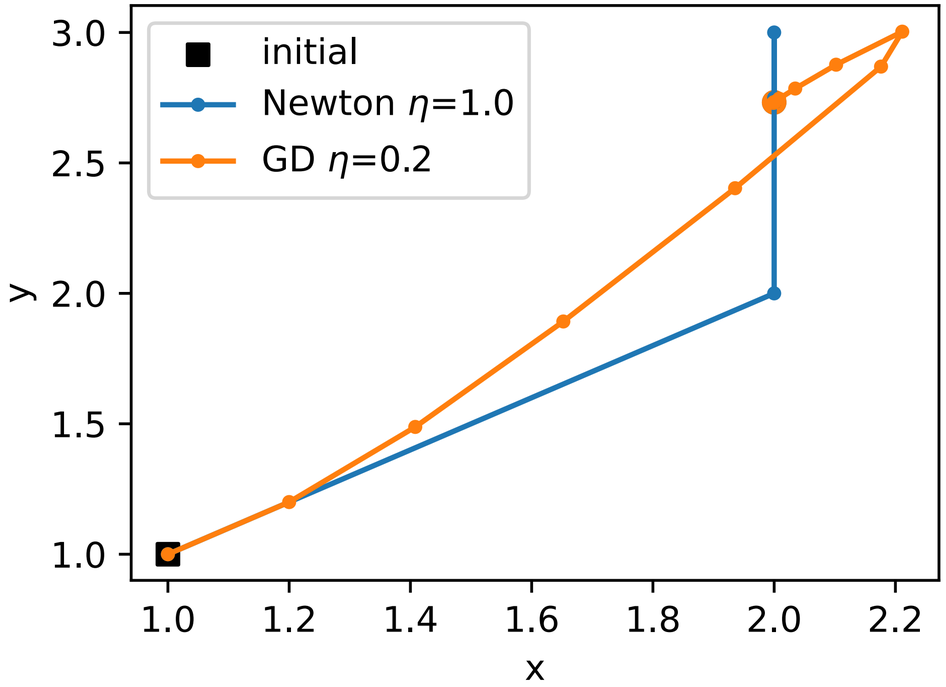
$$x - y^2 + xy = 0$$
$$x - 2 = 0$$

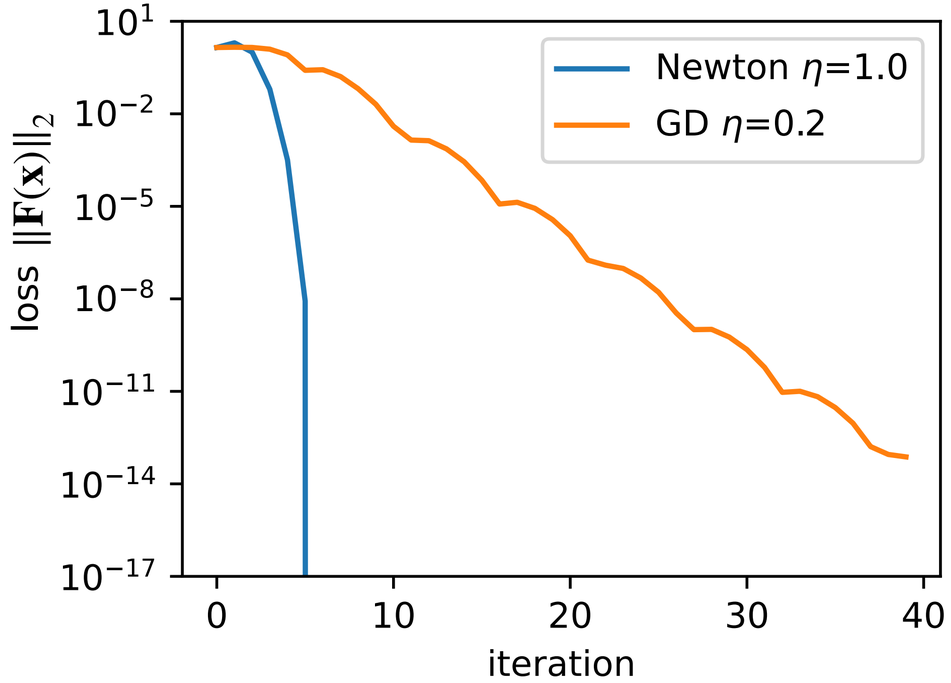
Gradient descent

$$\mathbf{x}^{s+1} = \mathbf{x}^s - \eta \left(\frac{\partial \mathbf{F}^s}{\partial \mathbf{x}} \right)^T \mathbf{F}^s$$

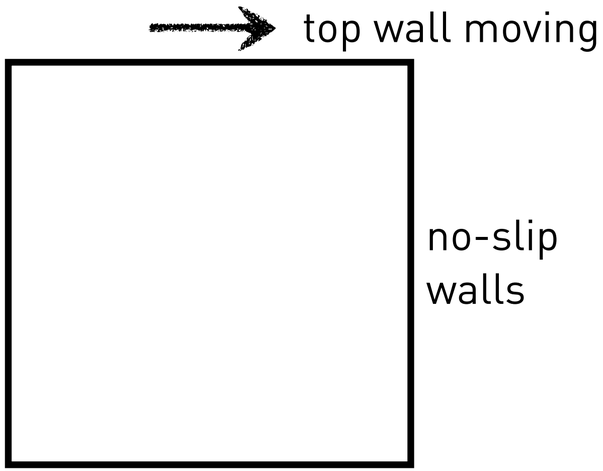
Newton

$$\frac{\partial \mathbf{F}^s}{\partial \mathbf{x}} (\mathbf{x}^{s+1} - \mathbf{x}^s) = \mathbf{F}^s$$





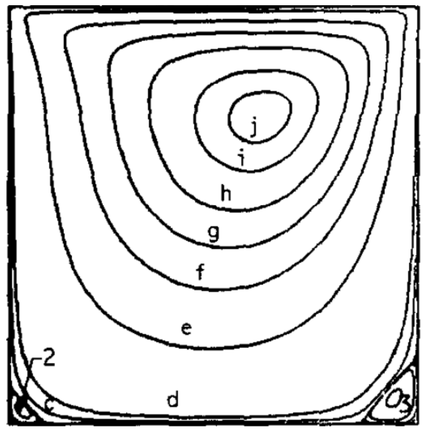
Lid-driven cavity



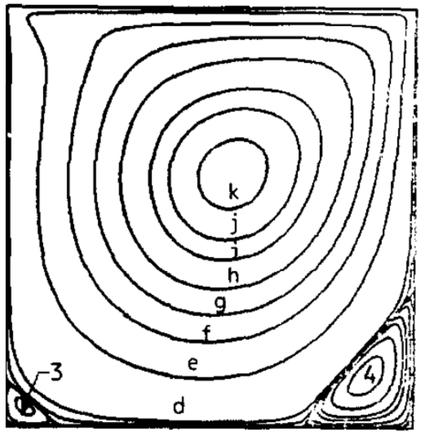
2D steady Navier-Stokes

$$\nabla \cdot \mathbf{u} = 0$$

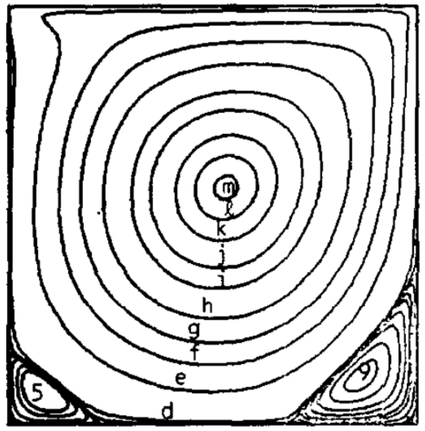
$$(\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla p + \frac{1}{\mathrm{Re}} \nabla^2 \mathbf{u}$$



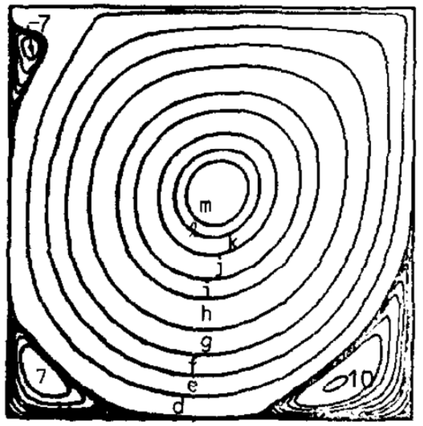
Re=100



Re=400



Re=1000



Re=3200

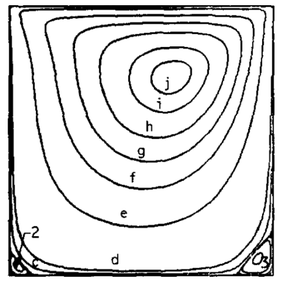
Ghia UK, Ghia KN, Shin CT. High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. Journal of computational physics. 1982

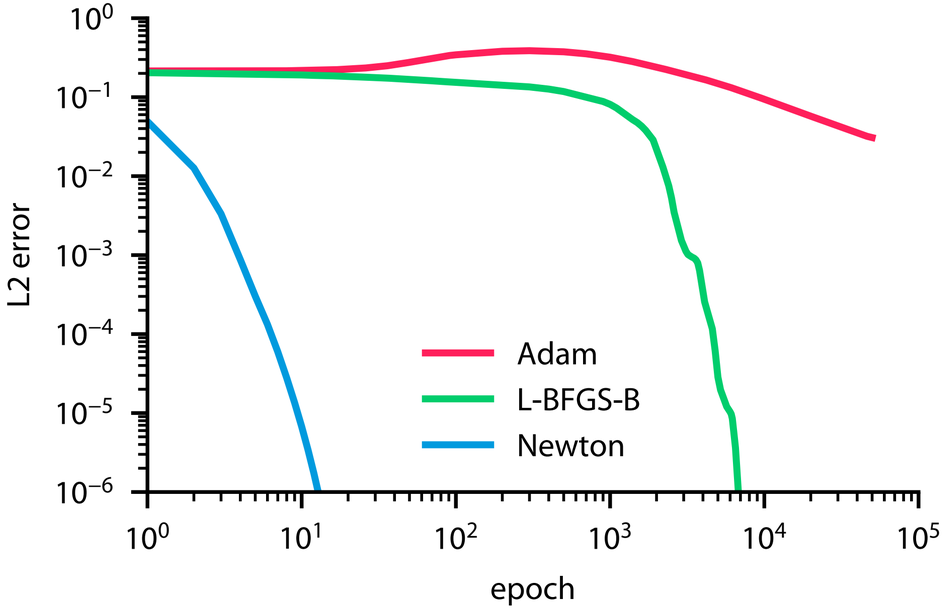
ODIL: Comparison of optimizers, lid-driven cavity

Convergence history of various optimizers

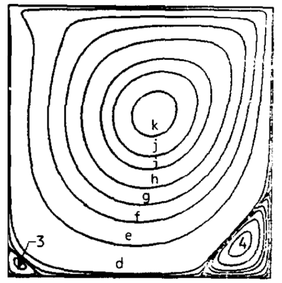
- Adam (variant of gradient descent)
- L-BFGS-B (quasi-Newton)
- Newton with sparse linear system

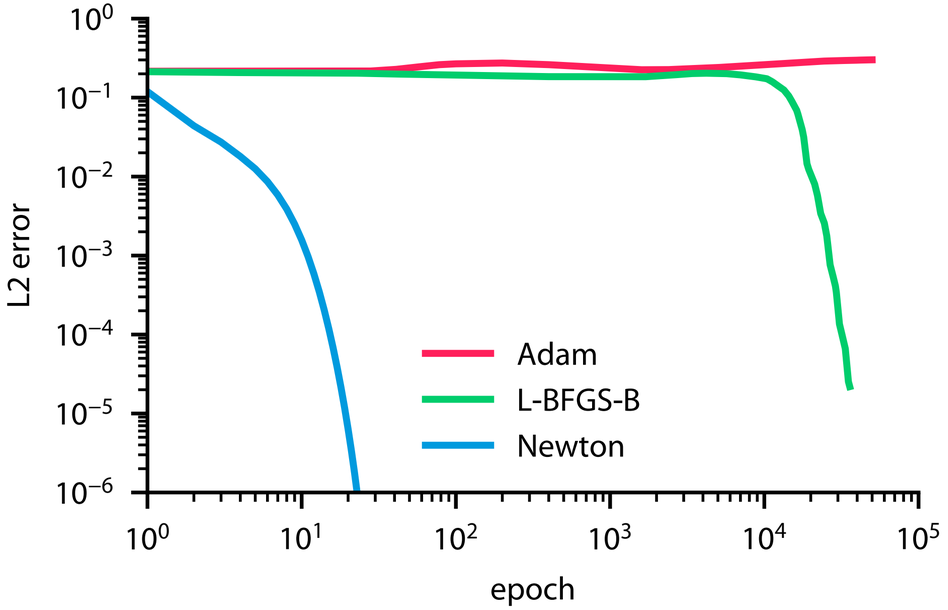
Re=100





Re=400





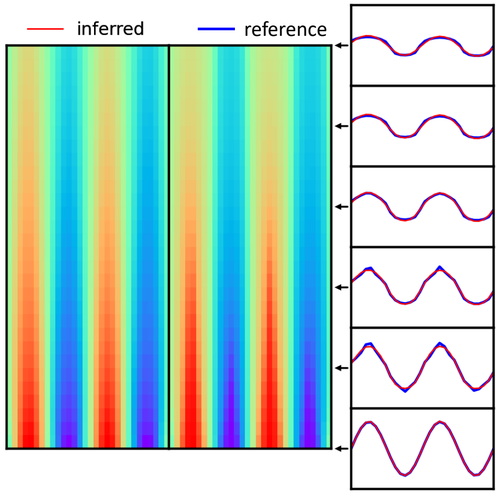
Kingma DP, Ba J. Adam: A method for stochastic optimization. ICLR. 2015

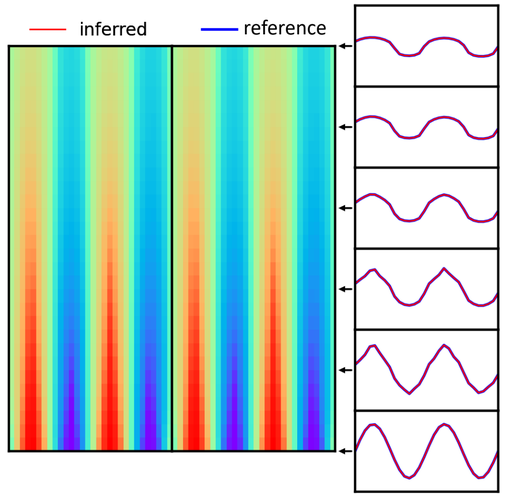
Zhu C, Byrd RH, Lu P, Nocedal J.
Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization.
ACM Transactions on mathematical software. 1997

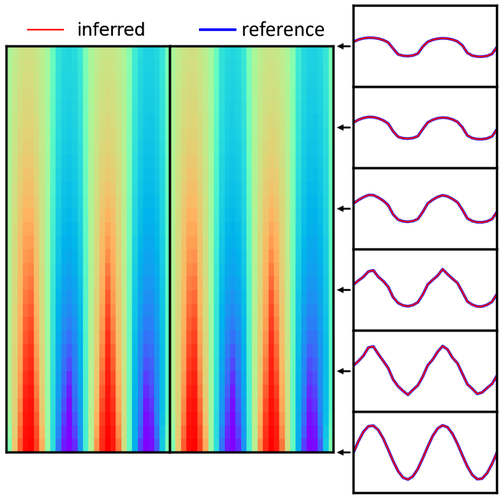
ODIL: Nonlinear diffusion

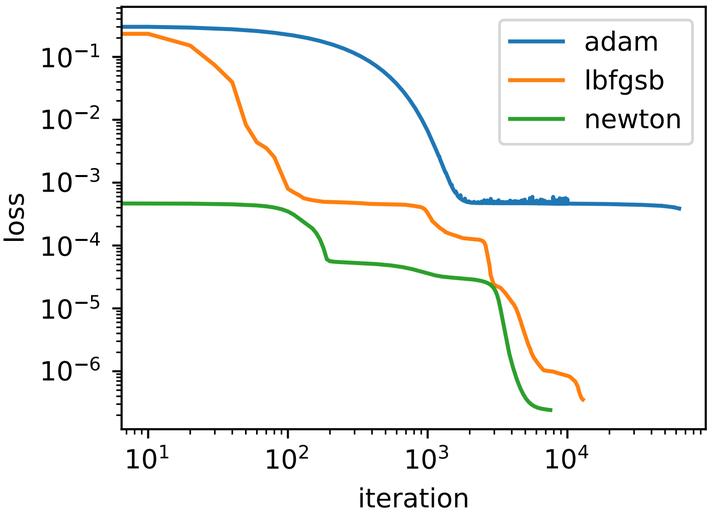
$$\frac{\partial u}{\partial t} - \frac{\partial}{\partial x}\left(k(u)\frac{\partial u}{\partial x}\right) = 0$$

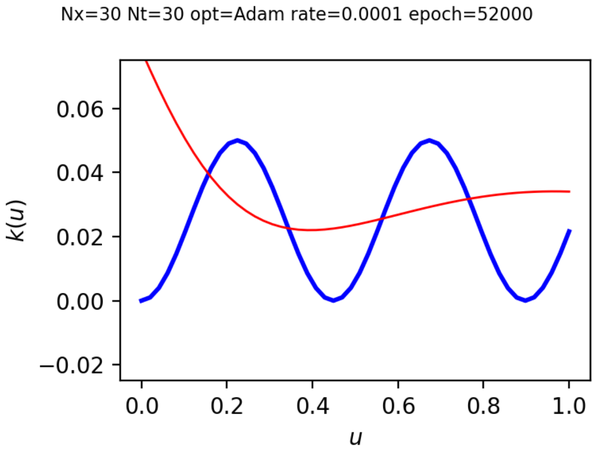
- imposed known solution at 6 time instants
- unknown $k(u)$ as neural network

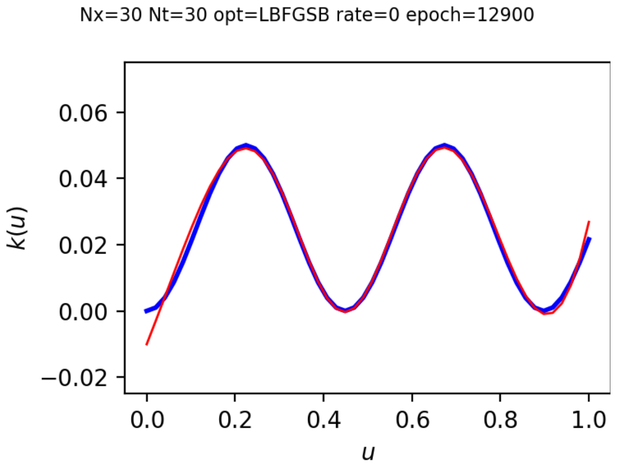


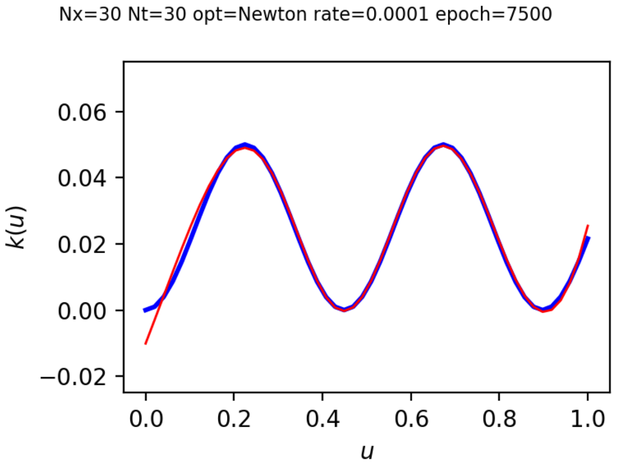












Adam

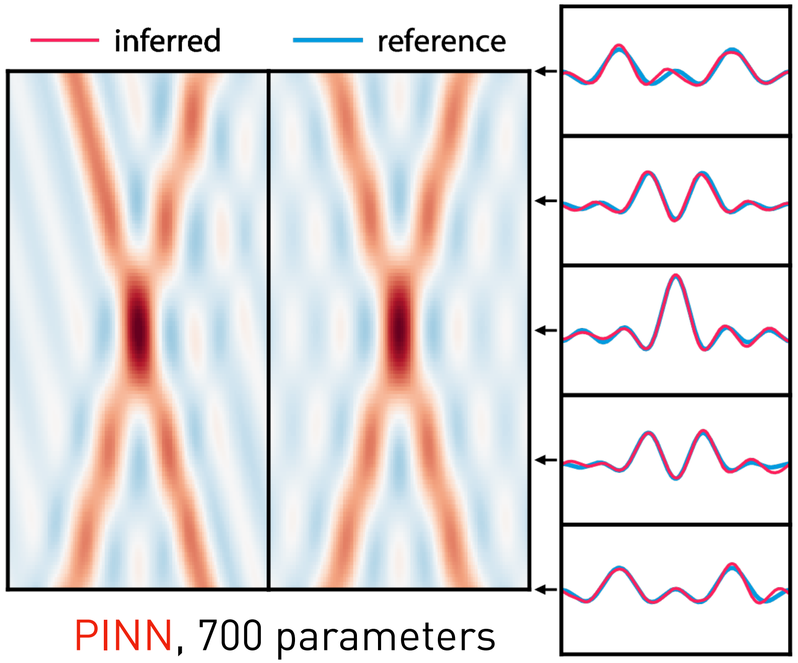
L-BFGS-B

Newton

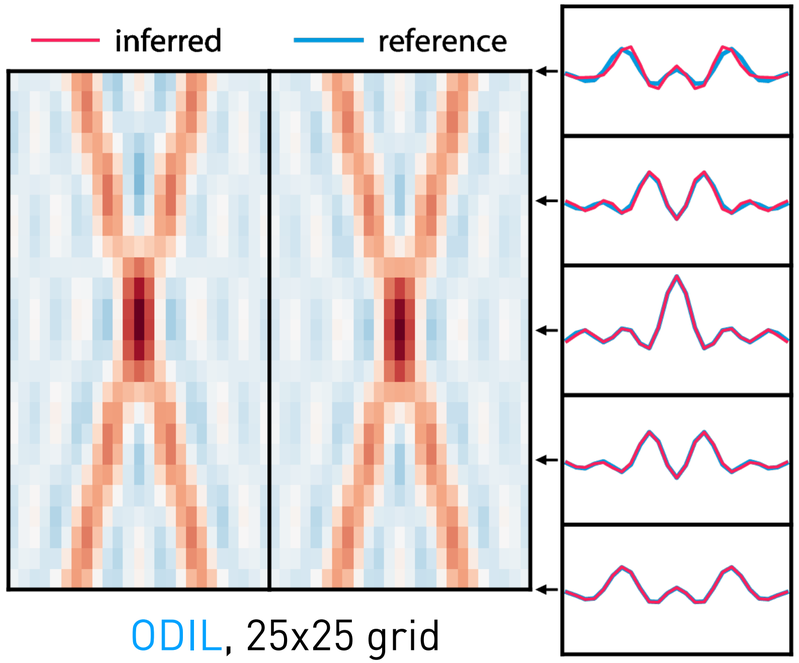
PINN vs ODIL

PINN (L-BFGS-B)
ODIL (L-BFGS-B)
ODIL (Newton)

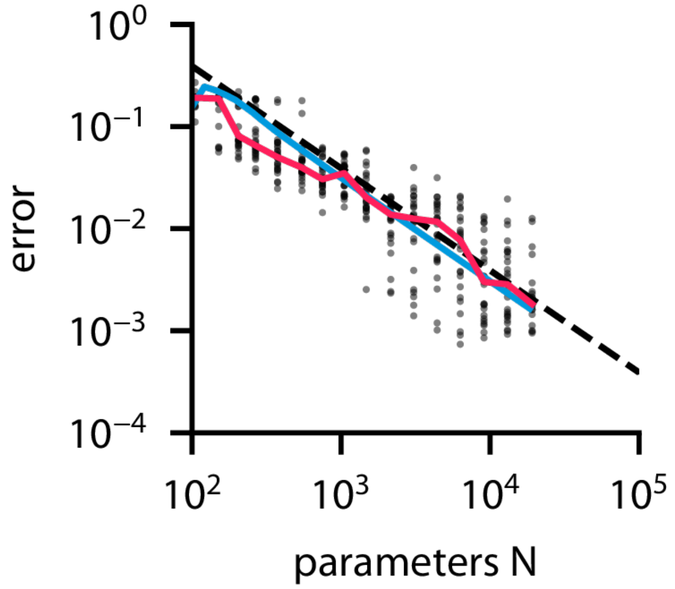
Wave equation $\frac{\partial^2 u}{\partial t^2} - \frac{\partial^2 u}{\partial x^2} = 0$

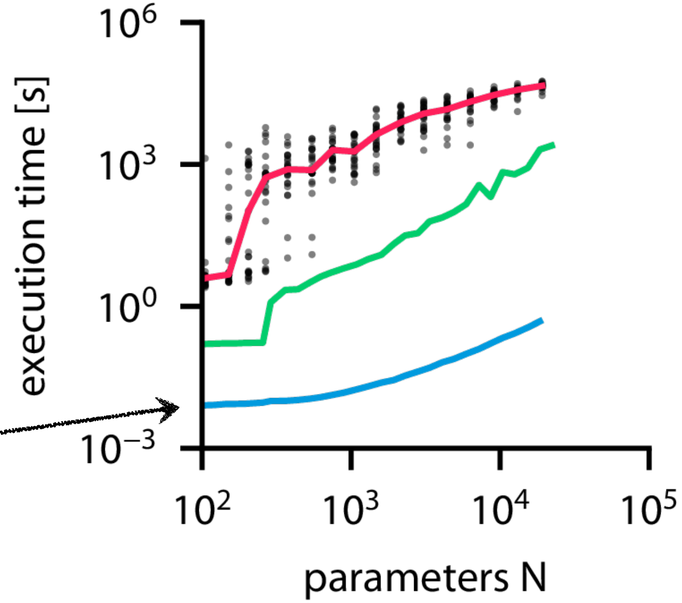


PINN, 700 parameters



ODIL, 25x25 grid





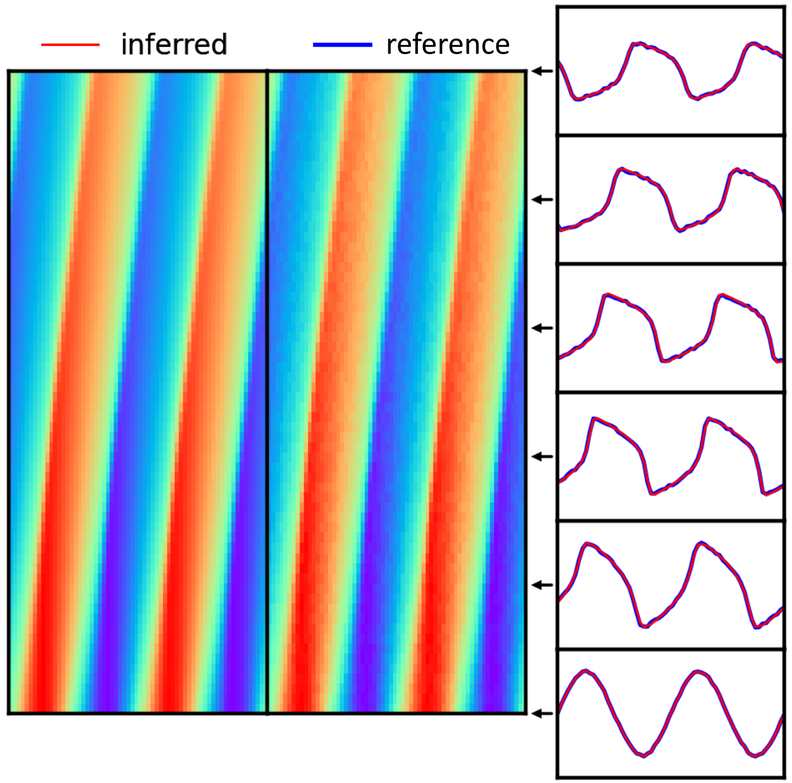
PINN has x100000 higher cost than ODIL

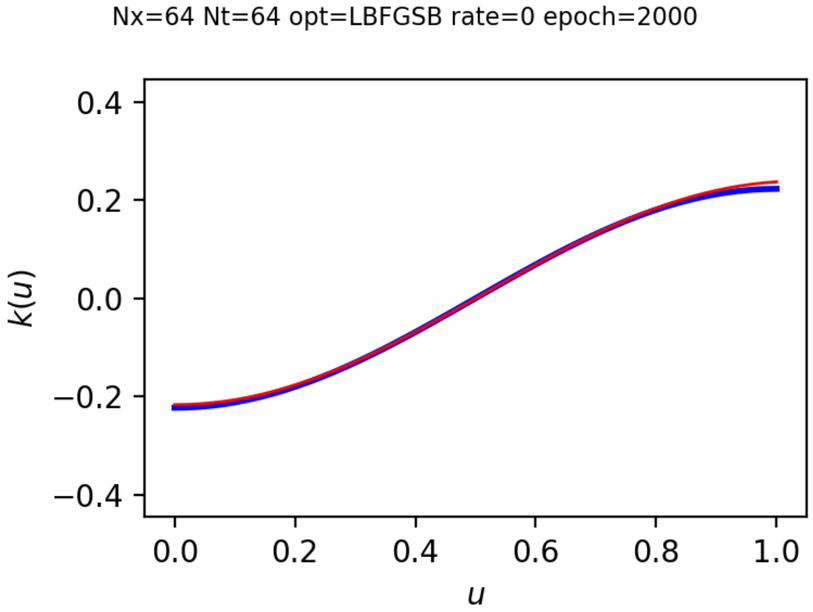
Applications

Inferring nonlinear flux

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x} k(u) = 0$$

- imposed solution at initial and final time
- unknown flux $k(u)$ as neural network

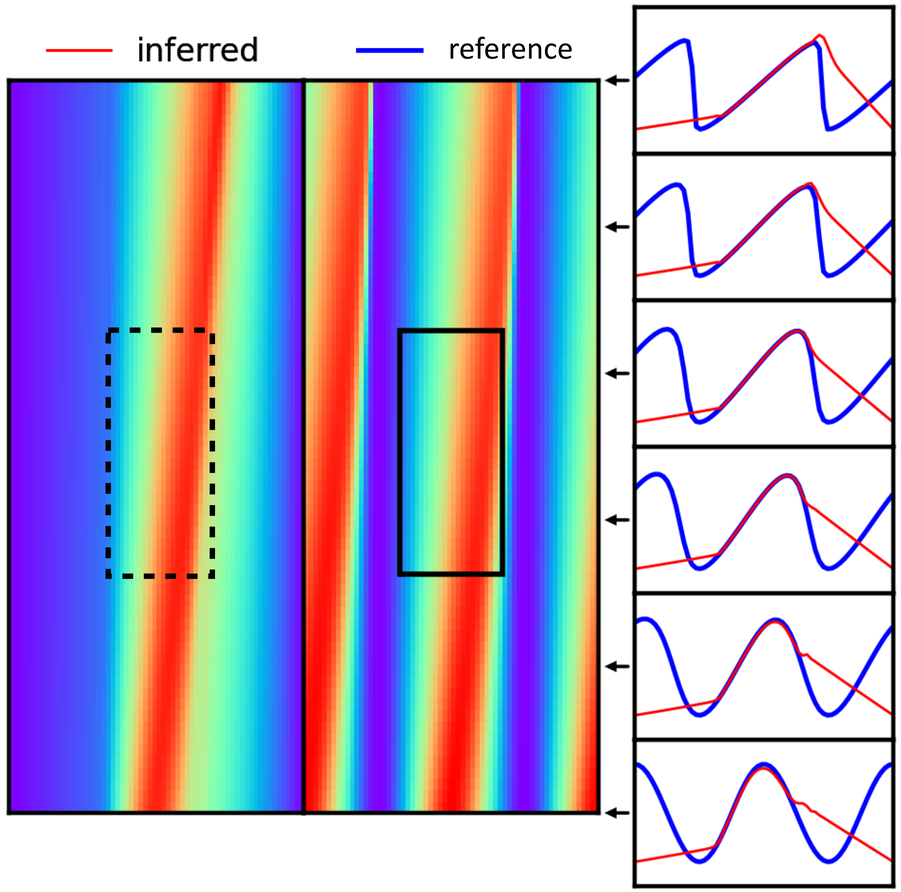




Burgers reconstruction

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0$$

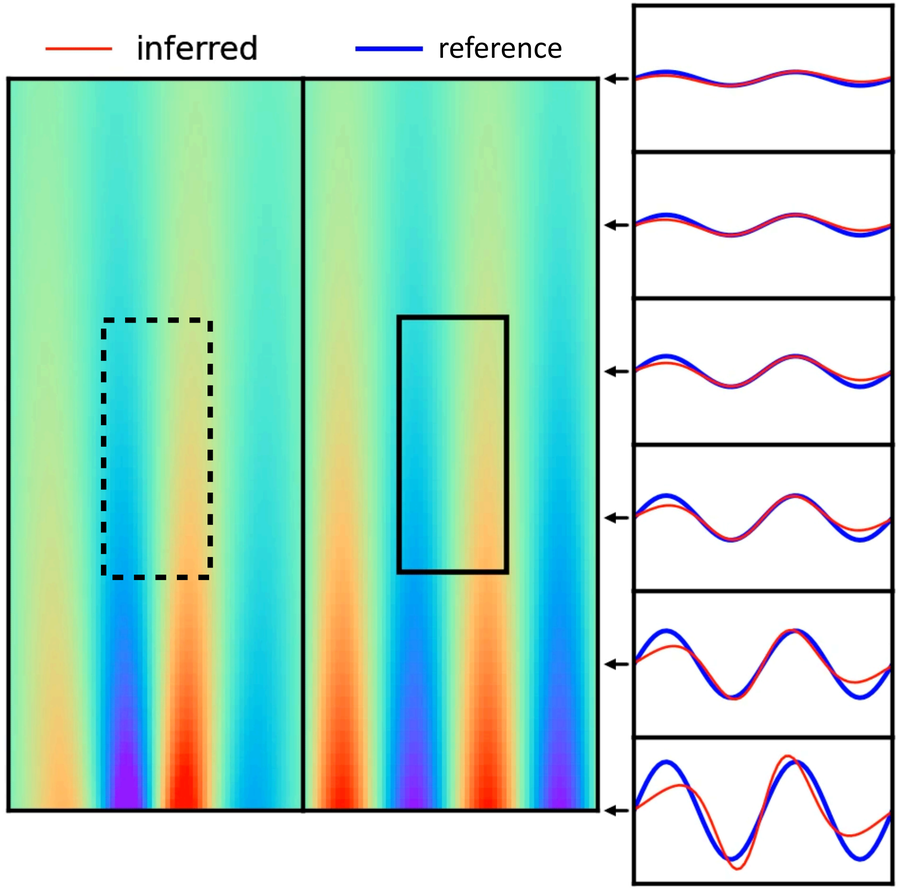
imposed
reference solution
inside rectangle



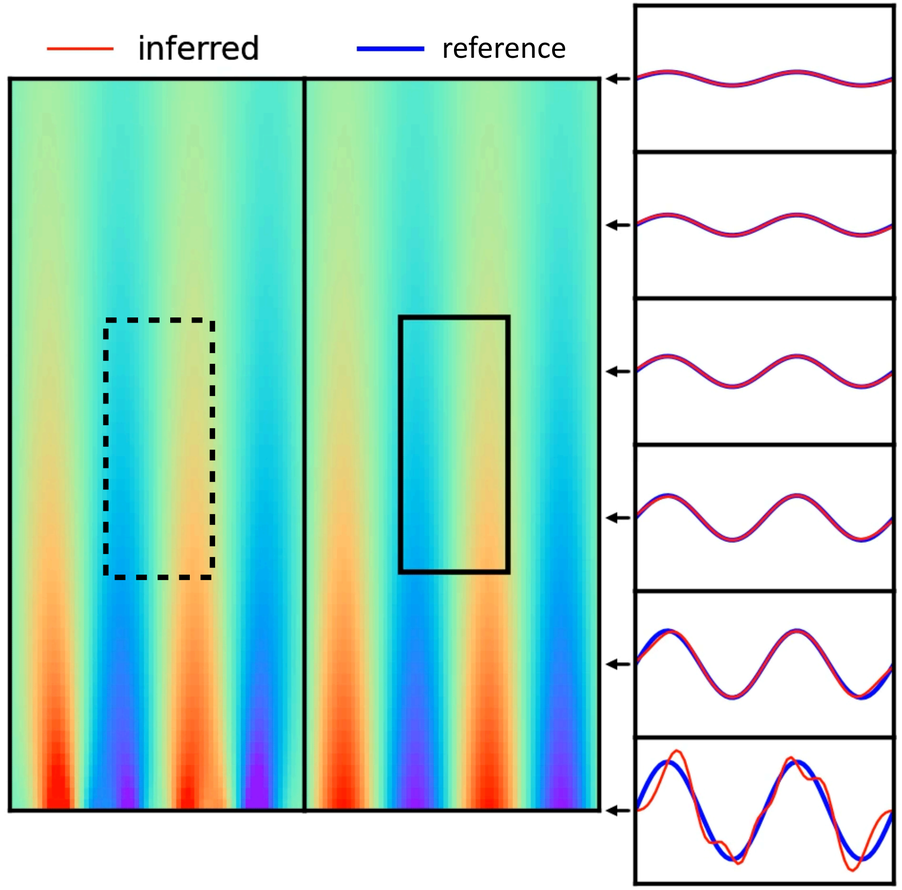
Diffusion reconstruction: ODIL vs PIN

$$\frac{\partial u}{\partial t} - k\frac{\partial^2 u}{\partial x^2} = 0$$

imposed
reference solution
inside rectangle



PINN, 3700 iterations



FD Newton, 150 iterations

Velocity from tracer

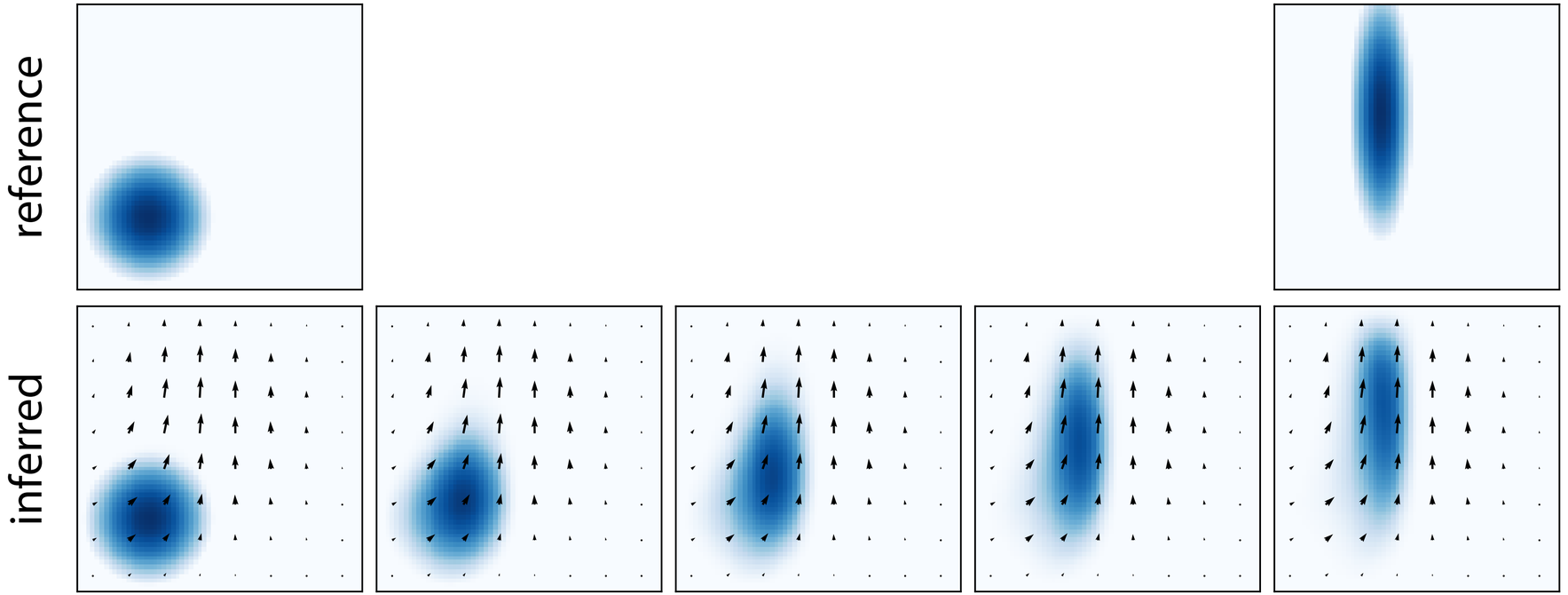
- Find velocity $\mathbf{u}(x, y)$ and tracer $c(x, y, t)$

- Given

$$\frac{\partial c}{\partial t} + \mathbf{u} \cdot \nabla c = 0$$

$$c(x, y, 0) = c_0(x, y)$$

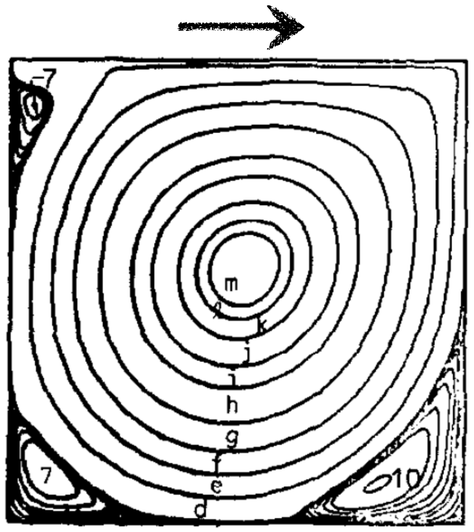
$$c(x, y, 1) = c_1(x, y)$$



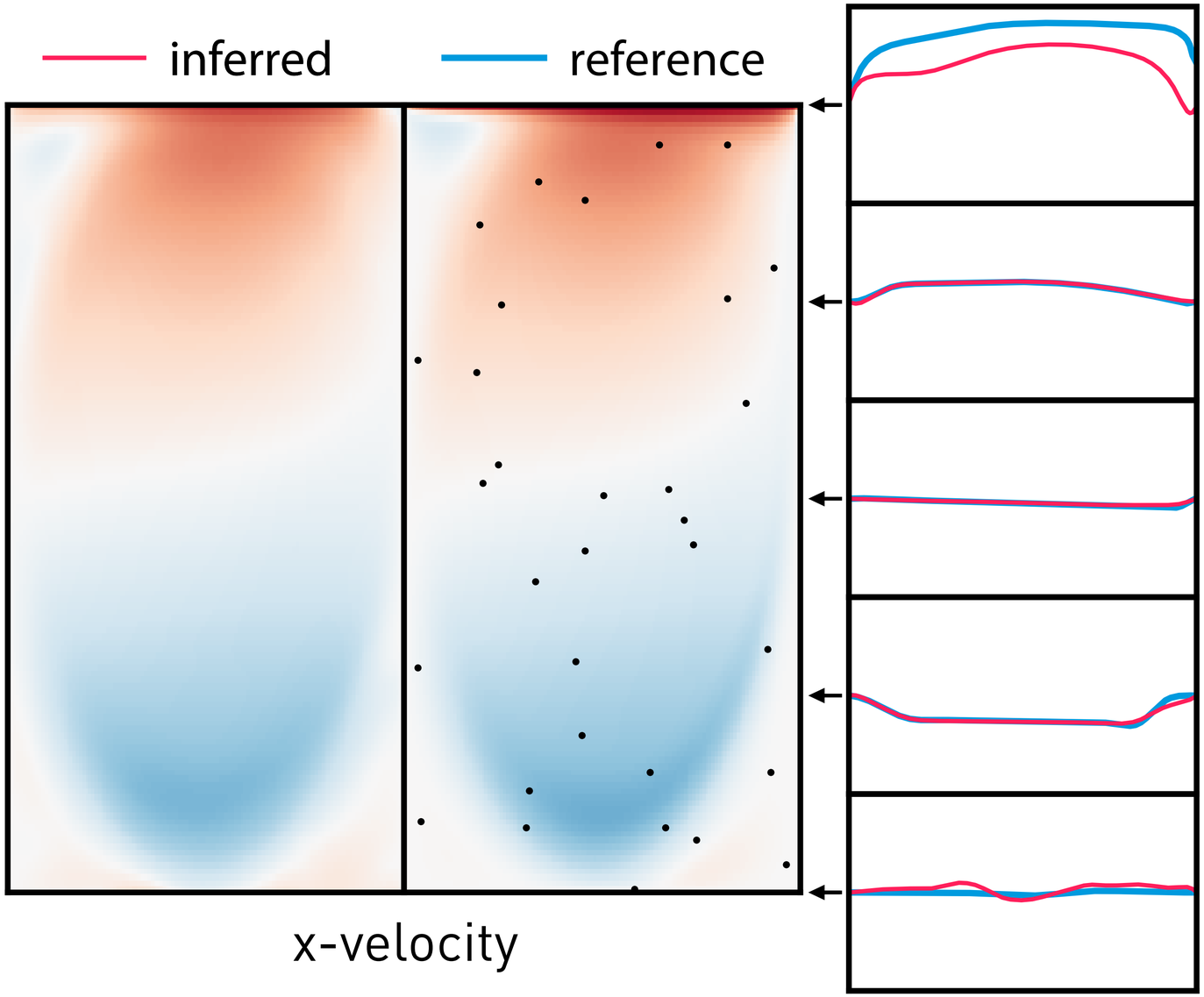
Lid-driven cavity reconstruction

- imposed reference solution in 32 points
- imposed equations everywhere,

  plus regularization $\|\partial^2 u_i / \partial x_j^2\|$
- free-slip walls



Re=3200



x-velocity

Measure of flow complexity

- What's the best accuracy of reconstruction from K points?

$$E(K) = \min_{|X|=K} \|\mathbf{u}_{\text{ref}} - \mathbf{u}(X)\|$$

$\mathbf{u}(X)$ is velocity reconstructed from points X

- How many points are needed to reconstruct the flow?

$$K_{\min}(\varepsilon) = \min\{K \mid E(K) < \varepsilon\}$$