Data-driven numerical simulation with application in computational fluid dynamics

Jiaxi Zhao

Interview for Sea AI Lab

7th Dec, 2023

Data-driven scientific computing

What are the problems we are interested in?

1. Forward problem: Increase the stability and accuracy of machine learning-augmented simulation

2. Inverse problem: Perform effective sensitivity analysis to do inverse design

What are the method we focus on?

1. 100% data-driven: Physics-informed neural networks (PINN), Fourier neural operator, DeepONet.

2. 50 % Numerical + 50 % data-driven: Machine learning turbulence modeling, DeepPotential, Quasipotential.

An example

Let us use incompressible NS equation as an example

$$
\begin{aligned}
\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} - \nu \Delta \mathbf{u} = \nabla p, \quad T \in [0, 1], \\
\nabla \cdot \mathbf{u} = 0.
\end{aligned}
\tag{1}
$$

Consider solving it using the projection method, in each step, we need to solve the following equation

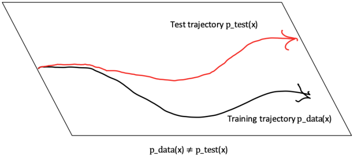
$$
\begin{aligned}
\mathbf{u}_{k+1} &= \mathbf{u}_k + \Delta t(\nu \Delta \mathbf{u}_k - (\mathbf{u}_k \cdot \nabla)\mathbf{u}_k - \nabla p_k), \\
p_k &= \phi(\mathbf{u}_k) = \Delta^{-1}(\nabla \cdot (\nu \Delta \mathbf{u}_k - (\mathbf{u}_k \cdot \nabla)\mathbf{u}_k)),
\end{aligned}
\tag{2}
$$

The most important features are:

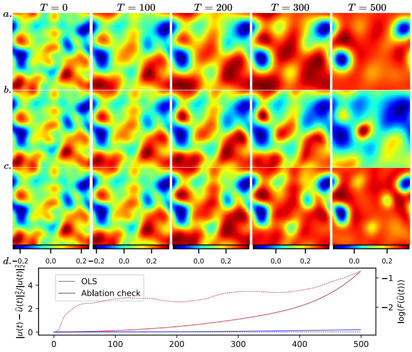
1. **iterative solver**
2. **data-driven**

Dilemma of data-driven scientific computing

In the data-driven scientific computing, **dynamics structure** can cause **distribution mismatch** between the training and testing data. Similarly to the **extrapolation, OOD** issue in NLP.

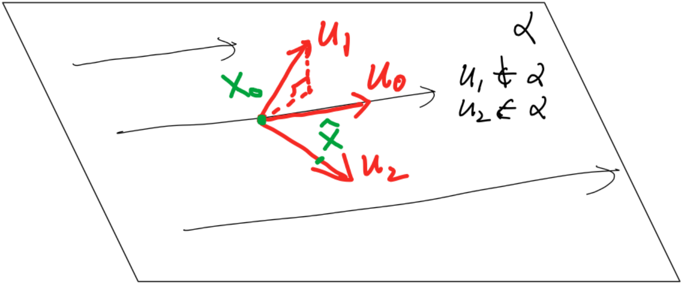


(a) Distribution shift illustration



(b) Distribution shift in reaction-diffusion equation

An heuristic solution



We design an algorithm that favours $\mathbf{u}_2$ than $\mathbf{u}_1$ by adding some regularization.

Network architecture

We choose U-net for

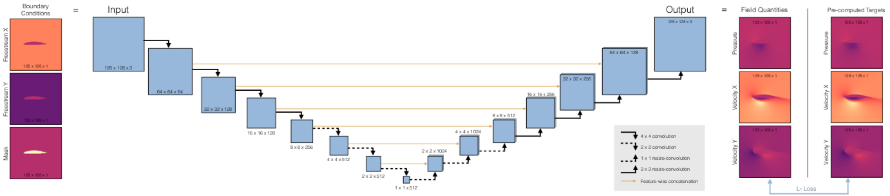


Figure: U-net structure for flow prediction[1]

[1] Thuerey, Nils, et al. "Deep learning methods for Reynolds-averaged Navier–Stokes simulations of airfoil flows." AIAA Journal 58.1 (2020): 25-36.

Performance comparison

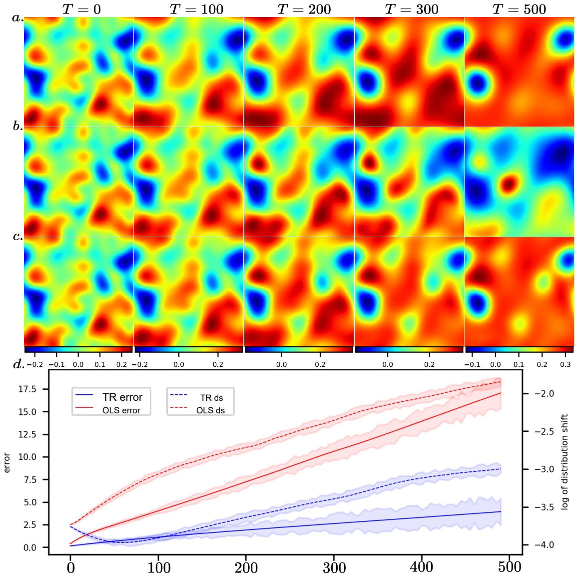


Figure: Comparison of our method and naive method

Performance comparison

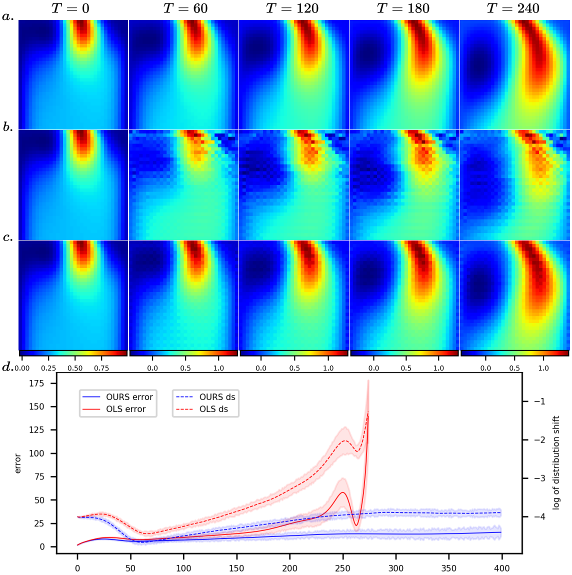


Figure: Comparison of our method and naive method

Further Application

1. Various turbulence modeling: Subgrid modeling, Wall modeling, Transition modeling, etc.
2. Coupled CFD: Fluid-structure interaction (multiphase flow), flow with heat transfer, etc.

Iterative solver and data-driven are also presented in these applcations.

The idea of shadowing

Consider a parametric dynamics

$$\partial_t u = f(u, s). \qquad (3)$$

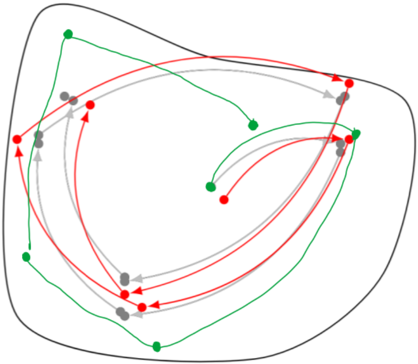


Figure: Shadowing trajectory: perturbed $s + \delta s$-trajectory and shadowing $s + \delta s$-trajectory

Solve LSS: Least square

Writing the linearized equation as a linear constraint

$$
\min \sum_{t=1}^{T} v_t^T v_t
$$

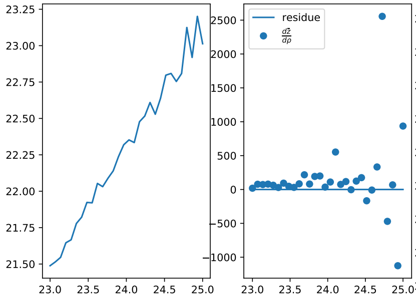
$$
\begin{pmatrix}
\mathbf{I} & -\nabla_u f(u_{T-1}) & \cdots & 0 & 0 \\
0 & \mathbf{I} & \cdots & 0 & 0 \\
0 & 0 & \cdots & 0 & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \cdots & \mathbf{I} & -\nabla_u f(u_1) \\
0 & 0 & \cdots & 0 & 0
\end{pmatrix}
\begin{pmatrix}
v_T \\ v_{T-1} \\ v_{T-2} \\ \vdots \\ v_2 \\ v_1
\end{pmatrix}
=
\begin{pmatrix}
\partial_s f(u_{T-1}) \\ \partial_s f(u_{T-2}) \\ \partial_s f(u_{T-3}) \\ \vdots \\ \partial_s f(u_1) \\ 0
\end{pmatrix},
\tag{4}
$$

This is just a least square problem of size $T \times N$. The huge linear system is the linearized equation.

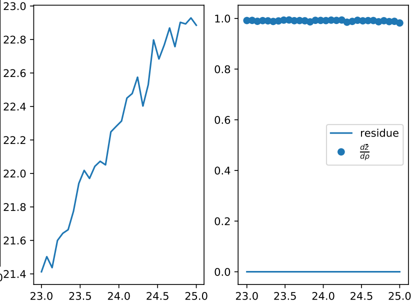
Sensitivity analysis of the Lorenz system

Let us start with the simple 3D Lorenz system (a reduced-order model for heat transfer flow) with a parameter ρ (Reyleigh number)

$$J(\rho) := \lim_{T\to\infty} \frac{1}{T} \int_0^T z_\rho(t)dt. \tag{5}$$



(a) Gradient calcuated using naive method



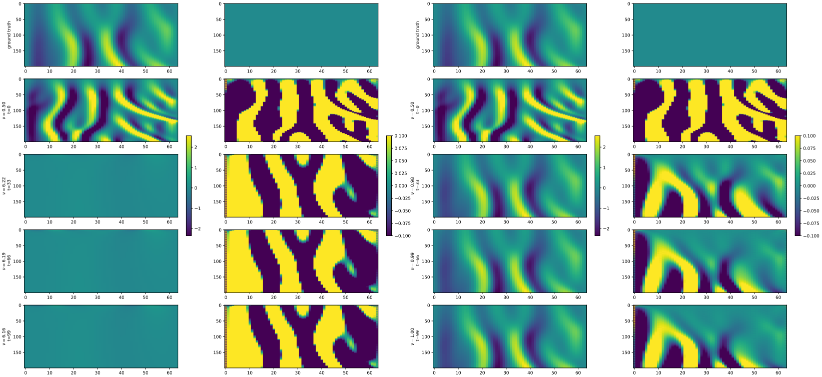
(b) Gradient calcuated using LSS

Figure: Comparison of naive and LSS methods for sensitivity analysis

Inverse problem for KS equation

Consider the 1D Kuramoto–Sivashinsky equation:

$$\partial_t u + uu_x + u_{xx} + \nu u_{xxxx} = 0, \quad x \in [0, L], \tag{6}$$



(a) Optimization using adjoint method (b) Optimization using LSS

Figure: Performance of LSS

Sensitivity analysis of LES subgrid modeling

Difficulties: We need to solve a linear system of size $N \times T$, with N the number of cells or grid points and T the number of time steps. This is computationally prohibited for even moderate LES.

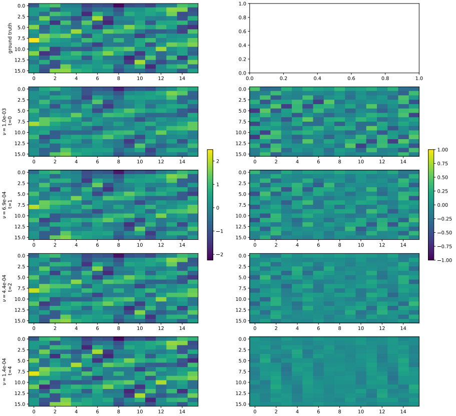


Figure: Subgrid modeling of NS equation in 2D

Sensitivity analysis of LES subgrid modeling

Then, why not have a try with machine learning?

$$\mathbf{u} \xrightarrow{NN} v_1. \tag{7}$$

1. Seq2Seq processing
2. Using the idea of Neural radiance field (NeRF) to directly learn the policy
3. Unsupervised learning or Reinforcement learning

Application

1. Flow control
2. Inverse design & shape optimization
3. Uncertainty quantification of fluid system

Appendix: Reaction-diffusion equation

Consider following FitzHugh-Nagumo reaction diffusion equation:

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} &= \gamma \Delta \mathbf{u} + \mathbf{R}(\mathbf{u}), \quad T \in [0,1], \\ \mathbf{R}(\mathbf{u}) = \mathbf{R}(u, v) &= \begin{pmatrix} u - u^3 - v - \alpha \\ \beta(u - v) \end{pmatrix}, \end{aligned} \tag{8}$$

The initial data is given by $\mathbf{u}_0$ is a random field and generated by i.i.d. sampling from a normal distribution and $\alpha = 0.001, \beta = 1.0, \gamma = \begin{pmatrix} 0.05 & 0 \\ 0 & 0.1 \end{pmatrix}$. We use mesh size 128×128 for the whole problems. Computational domain is given by $[0, 6.4] \times [0, 6.4]$.

Appendix: Shadowing lemma

Theorem (Shadowing lemma)

Let Γ be a hyperbolic invariant set of a diffeomorphism f. There exists a neighborhood U of Γ with the following property: for any $\delta > 0$ there exists $\epsilon > 0$, such that any (finite or infinite) ϵ-pseudo-orbit that stays in U also stays in a δ-neighborhood of some true orbit[2].

In a hyperbolic invariant set, the dynamics exhibit a combination of stable and unstable behavior.

Shadowing trajectory demo:

[2] Pilyugin SY. Shadowing in dynamical systems. Lecture notes in mathematics, vol. 1706. Springer; 1999.