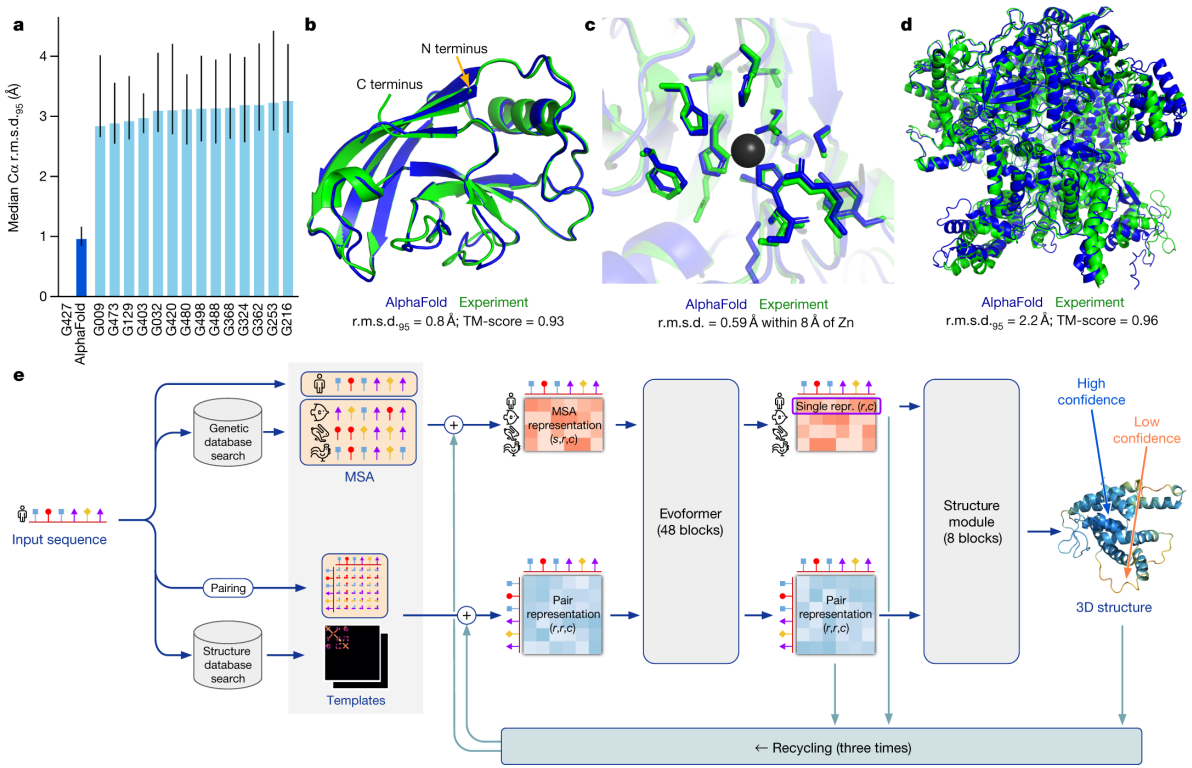
On distribution mismatch in Data-driven Scientific Computing

Jiaxi Zhao

17th Nov, 2022

Data-driven scientific computing

Data-driven method is becoming a prevalent surrogate model in mathematical modeling and scientific computing, due to its power on approximation the data in complicated and high dimensional setting.



Data-driven scientific computing: Three Categories

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.
3. **Discovering physics law from data**: Symbolic regression for conservation law, Principle component analysis for phase transition, cluster algorithm for space-time classification.

Data-driven for Reynolds stresses modeling

$\langle \mathbf{U} \rangle$ = time average of $\mathbf{U}$, $\mathbf{u} = \mathbf{U} - \langle \mathbf{U} \rangle$, Reynolds-averaged Navier-Stokes (RANS) equation reads

$$\begin{aligned} \partial_t \langle \mathbf{U} \rangle + (\langle \mathbf{U} \rangle \cdot \nabla) \langle \mathbf{U} \rangle + \frac{\partial \langle \mathbf{u} u_j \rangle}{\partial x_j} &= -\frac{1}{\rho} \nabla p + \nu \Delta \langle \mathbf{U} \rangle , \\ \nabla \cdot \langle \mathbf{U} \rangle &= 0. \end{aligned} \tag{1}$$

Extra term: $\langle \mathbf{u} u_j \rangle$ (Reynolds stress). **The equation is not closed**!!

Ling et al.[1]proposed a data-driven surrogate model to estimate the Reynolds stresses.

[1]Ling, Julia, Andrew Kurzawski, and Jeremy Templeton. "Reynolds averaged turbulence modelling using deep neural networks with embedded invariance." Journal of Fluid Mechanics 807 (2016): 155-166.

Improve the prediction of flow separation

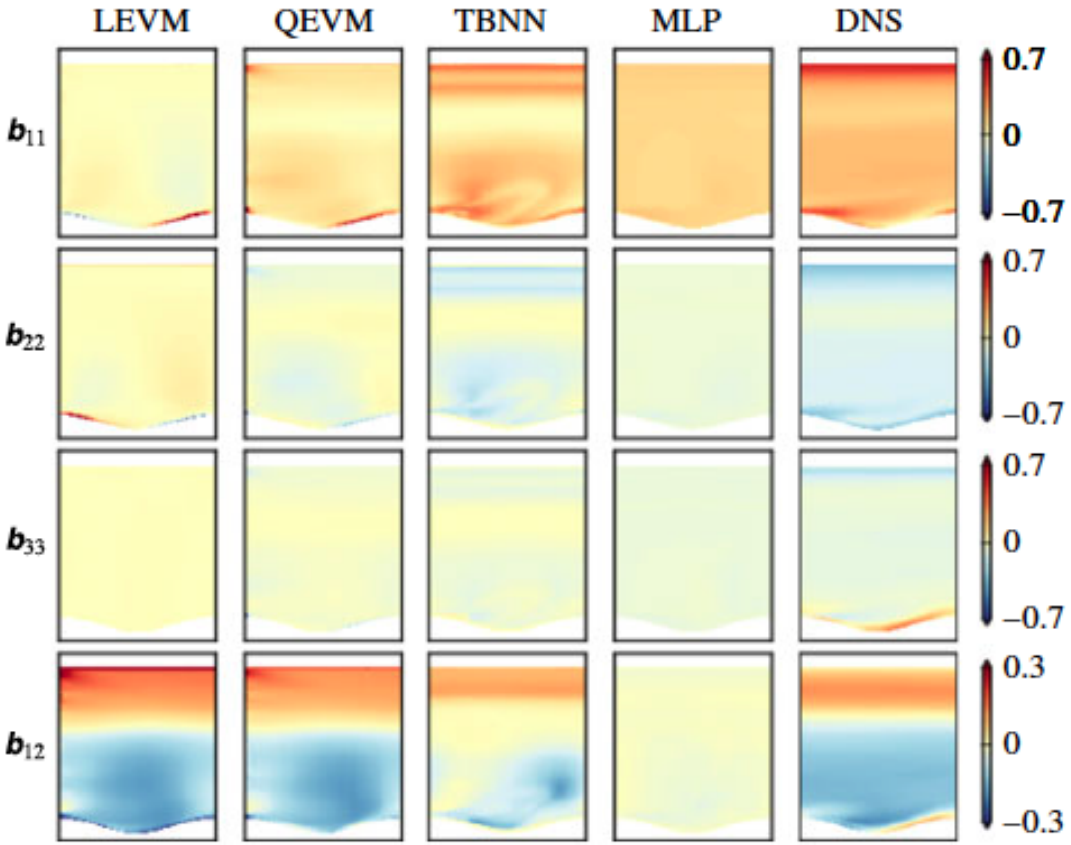


FIGURE 3. Predictions of Reynolds stress anisotropy $\boldsymbol{b}$ tensor on the wavy wall test case. The columns show the predictions of the LEVM, QEVM, TBNN and MLP models. The true DNS anisotropy values are shown in the right-most column for comparison.

Data-driven scientific computing: Our Focus

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.
3. **Discovering physics law from data**: Symbolic regression for conservation law, Principle component analysis for phase transition.

Framework for the second approach

Simulating the dynamics:

$$\partial_t \mathbf{x} = \mathcal{L}(\mathbf{x}, \mathbf{u}, t), \quad \mathbf{x} \in \mathcal{X}, \mathbf{u} \in \mathcal{U}, \mathcal{L} : \mathcal{X} \times \mathcal{U} \times \mathbb{R}_+ \to T\mathcal{X},$$
$$\mathbf{u} = \phi(\mathbf{x}, t), \quad \phi : \mathcal{X} \times \mathbb{R}_+ \to \mathbf{u}.$$

1. $\mathcal{L}$ is known, possibly non-linear.
2. ϕ is un-known.
3. A set of data pairs
$\{(\mathbf{x}_1, \mathbf{u}_1, t_1), (\mathbf{x}_2, \mathbf{u}_2, t_2), \cdots, (\mathbf{x}_N, \mathbf{u}_N, t_N)\}$.
4. Benchmark algorithm solves the ordinary least square:

$$\arg\min_{\theta} \mathbb{E} \left\| \mathbf{u} - \phi_\theta(\mathbf{x}, t) \right\|^2.$$

Inner-outer loop: Linear outer loop

In many scientific computing examples, further simplification:

$$\mathbf{x}_{k+1} = A\mathbf{x}_k + B\mathbf{u}_k, \quad \mathbf{x}_k \in \mathbb{R}^n, \mathbf{u}_k \in \mathbb{R}^m,$$
$$\mathbf{u}_k = f(\mathbf{x}_k).$$

The outer loop is known to us and usually has some good numerical properties, i.e. linear, stable, etc, while the inner loop can be complicated.

Inner-outer loop: examples

Example (RANS)

$$\begin{cases} \partial_t \langle \mathbf{U} \rangle + (\langle \mathbf{U} \rangle \cdot \nabla) \langle \mathbf{U} \rangle + \nabla \cdot \tau = -\frac{1}{\rho} \nabla p + \nu \Delta \langle \mathbf{U} \rangle, \\ \nabla \cdot \langle \mathbf{U} \rangle = 0. \end{cases}$$

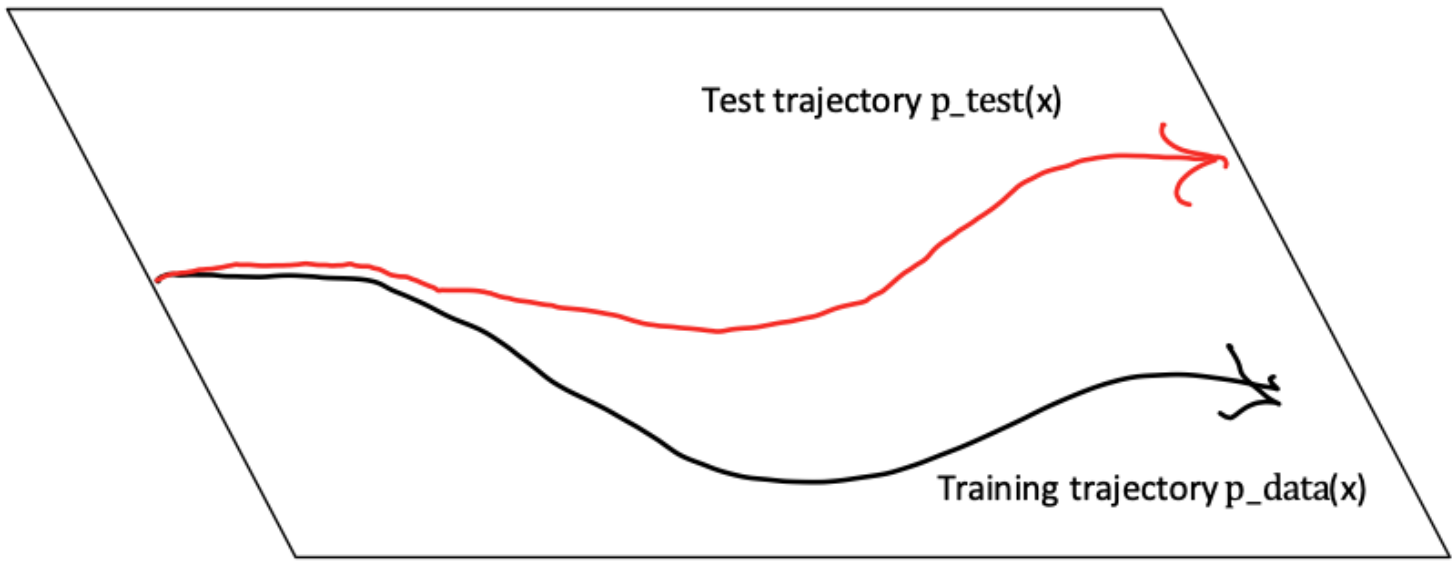
$$\tau = R(\langle \mathbf{U} \rangle).$$

Example (Quasi-potential)

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{x}_k - \Delta t \mathbf{u}_k, \\ \mathbf{u}_k &= \nabla V_\theta(\mathbf{x}_k) + g_\theta(\mathbf{x}_k). \end{aligned} \tag{2}$$

Dilemma of data-driven scientific computing

In the data-driven scientific computing, **dynamics structure** can cause **distribution mismatch** between the training and testing data.



Algorithm to mitigate the distribution mismatch

Modified the training dataset to mitigate the distribution shift.

Algorithm 1: DAgger: Dataset Aggregation

Data: π^*

Result: $\hat{\pi}^*$

$\mathcal{D} \leftarrow 0$

Initialize $\hat{\pi}$

for $i = 1$ **to** N **do**

  $\pi_i = \beta_i \pi^* + (1 - \beta_i)\hat{\pi}$

  Rollout policy π_i to sample trajectory $\tau = \{x_0, x_1, \dots\}$

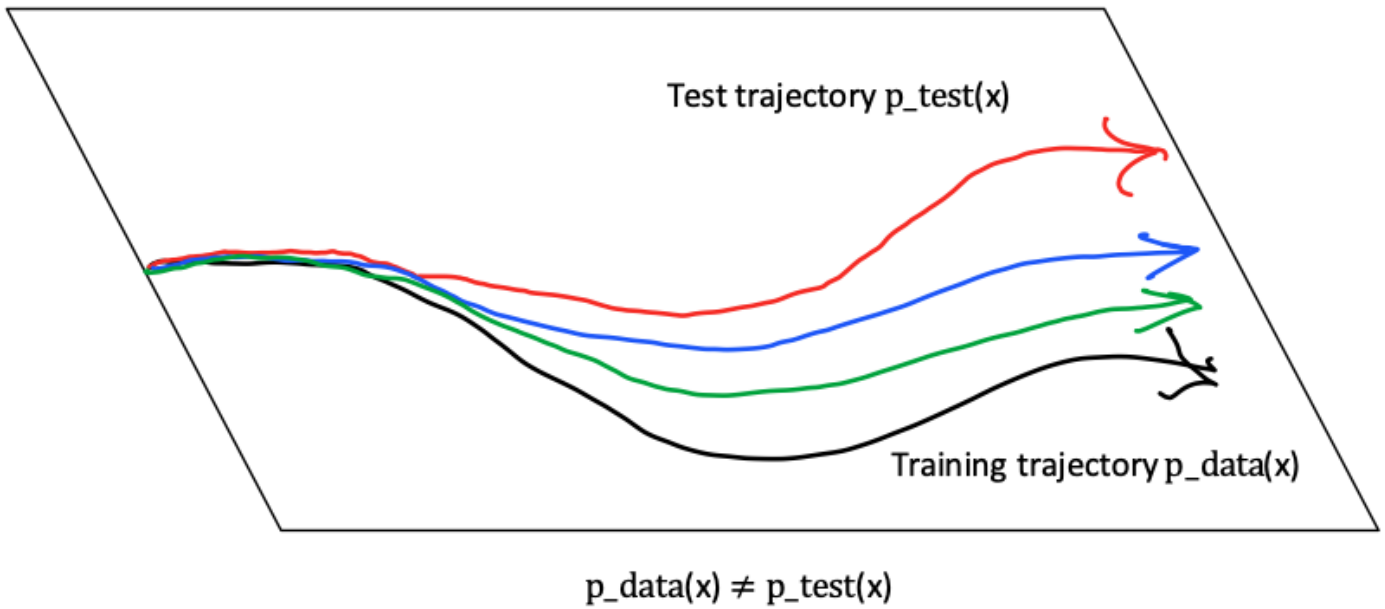
  Query expert to generate dataset $\mathcal{D}_i = \{(x_0, \pi^*(x_0)), (x_1, \pi^*(x_1)), \dots\}$

  Aggregate datsets, $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i$

  Retrain policy $\hat{\pi}$ using aggregated dataset $\mathcal{D}$

return $\hat{\pi}$

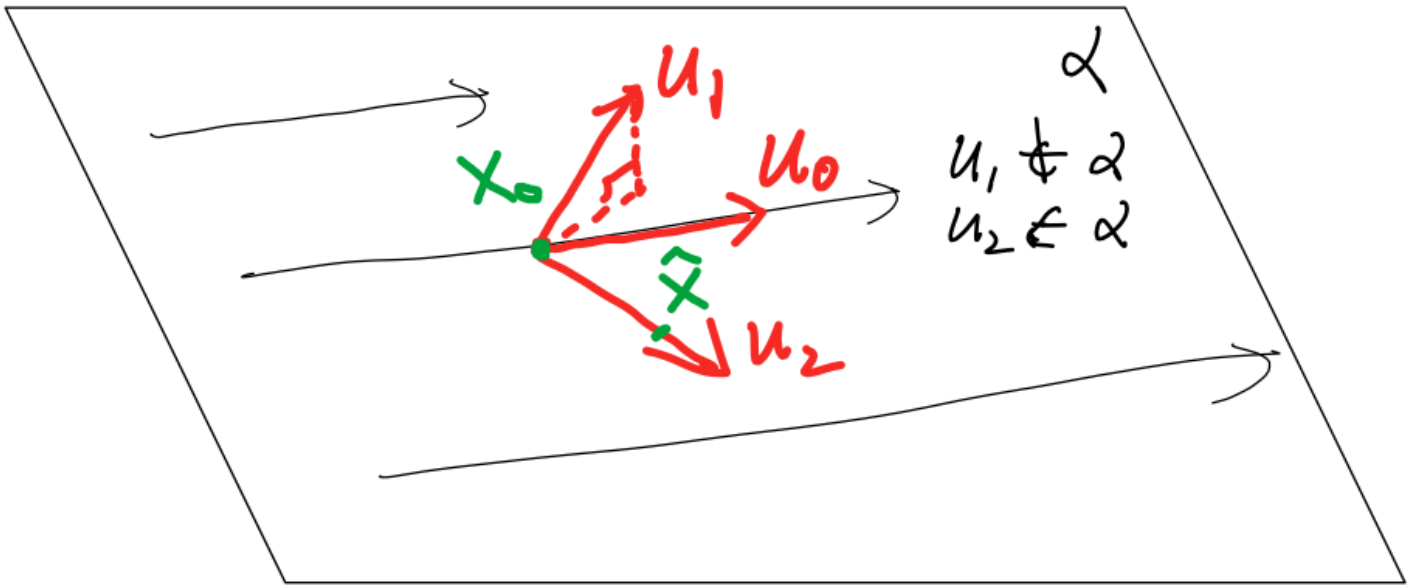
Algorithm to mitigate the distribution mismatch



An intuition

$$\partial_t \mathbf{x} = \mathcal{L}(\mathbf{x}, \mathbf{u}, t), \quad \mathbf{x} \in \mathcal{X}, \mathbf{u} \in \mathcal{U}, \mathcal{L} : \mathcal{X} \times \mathcal{U} \times \mathbb{R}_+ \to T\mathcal{X},$$
$$\mathbf{u} = \phi(\mathbf{x}, t), \quad \phi : \mathcal{X} \times \mathbb{R}_+ \to \mathbf{u}.$$

Ordinary least square $\|\mathbf{u} - \phi(\mathbf{x}, t)\|^2$ cannot distinguish the following case. Add regularization for this as $R(\widehat{\mathbf{x}})$ where $\widehat{\mathbf{x}}$ is next state variable given current state $\mathbf{x}$ and control $\mathbf{u}$, i.e. $\widehat{\mathbf{x}} = \mathbf{x} + \mathcal{L}(\mathbf{x}, \mathbf{u}, t)dt$.



Theoretic analysis: linear case

Theorem

Consider the following least square problem $\mathbf{Y} = \beta\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{m\times n}, \beta \in \mathbb{R}^{m\times m}, \mathbf{X} \in \mathbb{R}^{m\times n}$. *Furthermore, we suppose that the data* X_i *lies in a hyperplane of codimension* r *given by:*

$$c_i^T X_j = d_i, \forall i, j.$$

Then the regularized estimator proposed is given by

$$\widehat{\beta} = (\mathbf{I} + \lambda \mathbf{C}\mathbf{C}^T)^{-1}\mathbf{Y}\mathbf{X}^T(\mathbf{X}\mathbf{X})^{\dagger}.$$

Theoretic analysis

Theorem (Informal)

Assume that the simulated trajectories will not be far away from the data manifold, i.e. $d(\widehat{X}_i, \mathcal{M}) = o(1)$. *The data driven solver will be convergent, i.e.* $d(\widehat{X}_T, X_T) \to 0$ *as samples tends to infinity.*

Algorithms and numerical experiments

Linear control:

$$\begin{aligned} \dot{x}_t &= v_t, \\ \dot{v}_t &= -\alpha(t)v_t + u_t, \\ x_0 &= 1, v_0 = 0, \\ x_1 &= 0, \\ \alpha(t) &= \sin 10t. \end{aligned} \tag{3}$$

And we use a **neural network** with 2 hidden layer to parameterize the mapping from (x_k, v_k, t_k) to u_k. Hence the neurons in each layer is given by $3, n, n, 1$ where n can be varied in the experiments.

Algorithms and numerical experiments

Inner-Outer loop structure of this linear control:

$$\begin{aligned} \begin{pmatrix} x_{k+1} \\ v_{k+1} \end{pmatrix} &= \begin{pmatrix} 1 & dt \\ 0 & 1-\alpha_k dt \end{pmatrix} \begin{pmatrix} x_k \\ v_k \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u_k, \\ u_k &= \phi_{NN}(x_k, v_k, t_k). \end{aligned} \tag{4}$$

And ordinary least square (behavior cloning) simply solve the model by minimizing the following function:

$$l(\phi_{NN}(x_k, v_k, t_k), u_k) = \|\phi_{NN}(x_k, v_k, t_k) - u_k\|_2^2 .$$

Bench mark: Ordinary least square

Averaged error is given by 0.192341.

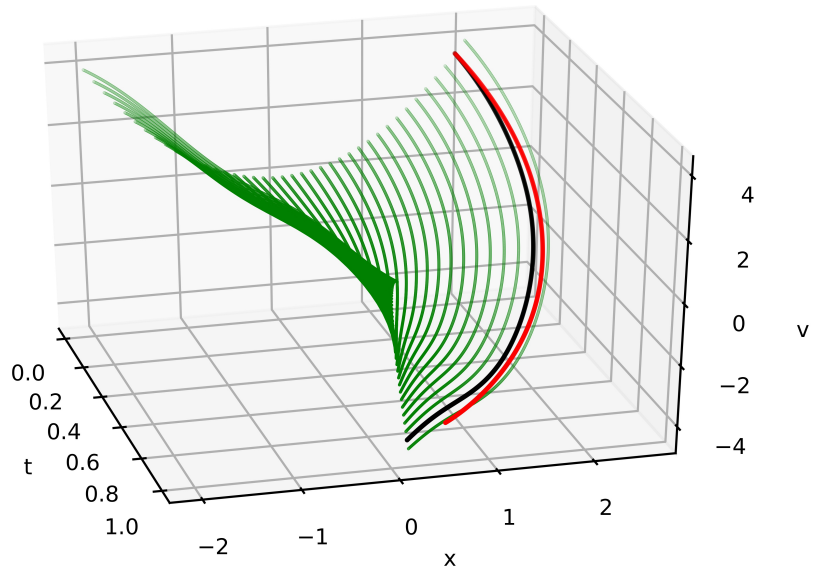


Figure: Sampled training trajectory of an estimator with underlying net of 10 hidden neurons.

Manifold regularization

Several choices of empirical manifold:

Formed by the sample trajectories in the training data, i.e.

$$\mathcal{M}_t := \{(x(t), v(t), t) \quad t \in [0, 1]\},$$

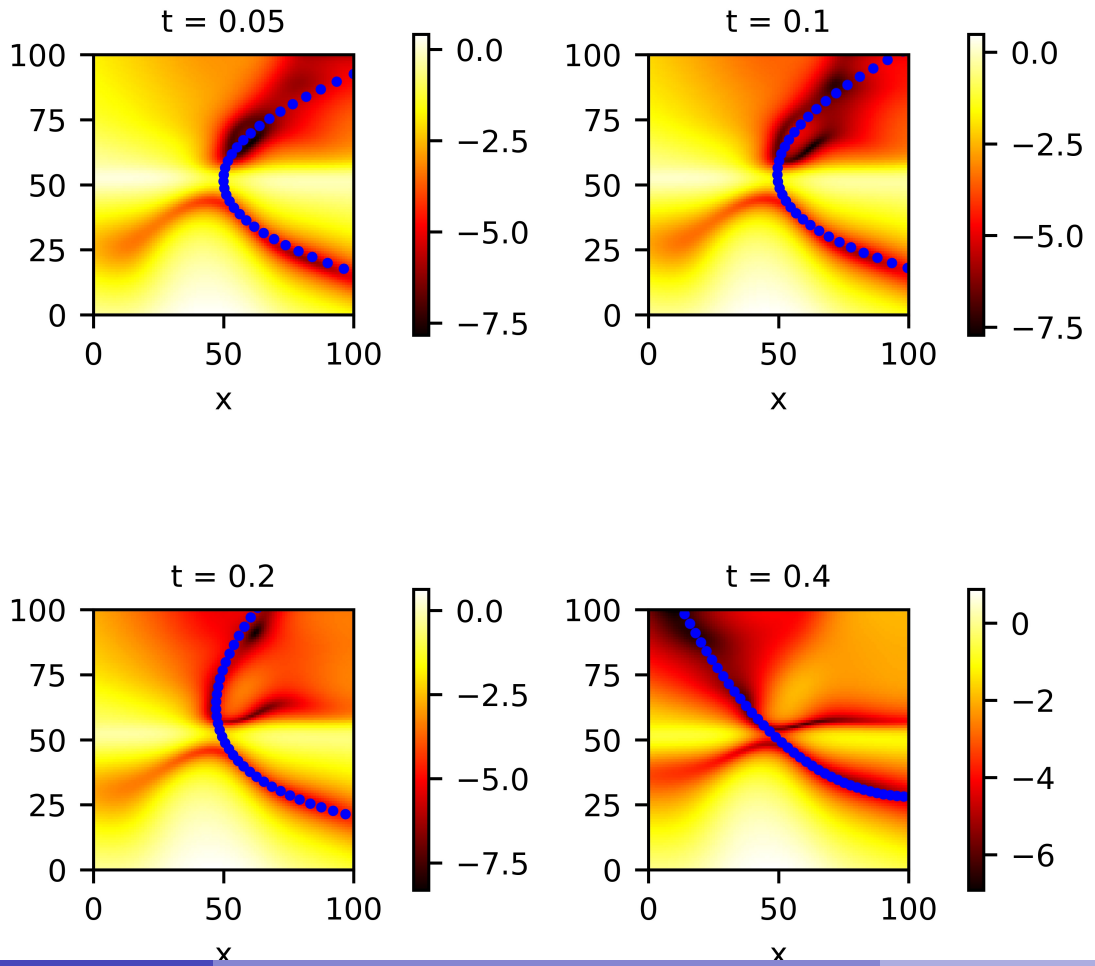
The other empirical manifold related to the data-driven surrogate model and can be defined as following set:

$$\widehat{\mathcal{M}}_t := \left\{(x, v, t) \middle| \, \|\phi_{NN}(x_k, v_k, t_k) - u_k\|_2 < \epsilon \right\}.$$

$\mathcal{M}_t$

$$\min_{\theta} \mathbb{E} \left\| \mathbf{u} - \phi_\theta(\mathbf{x}, t) \right\|^2 + \lambda \left\| D(E(\widehat{\mathbf{x}})) - \widehat{\mathbf{x}} \right\|^2$$

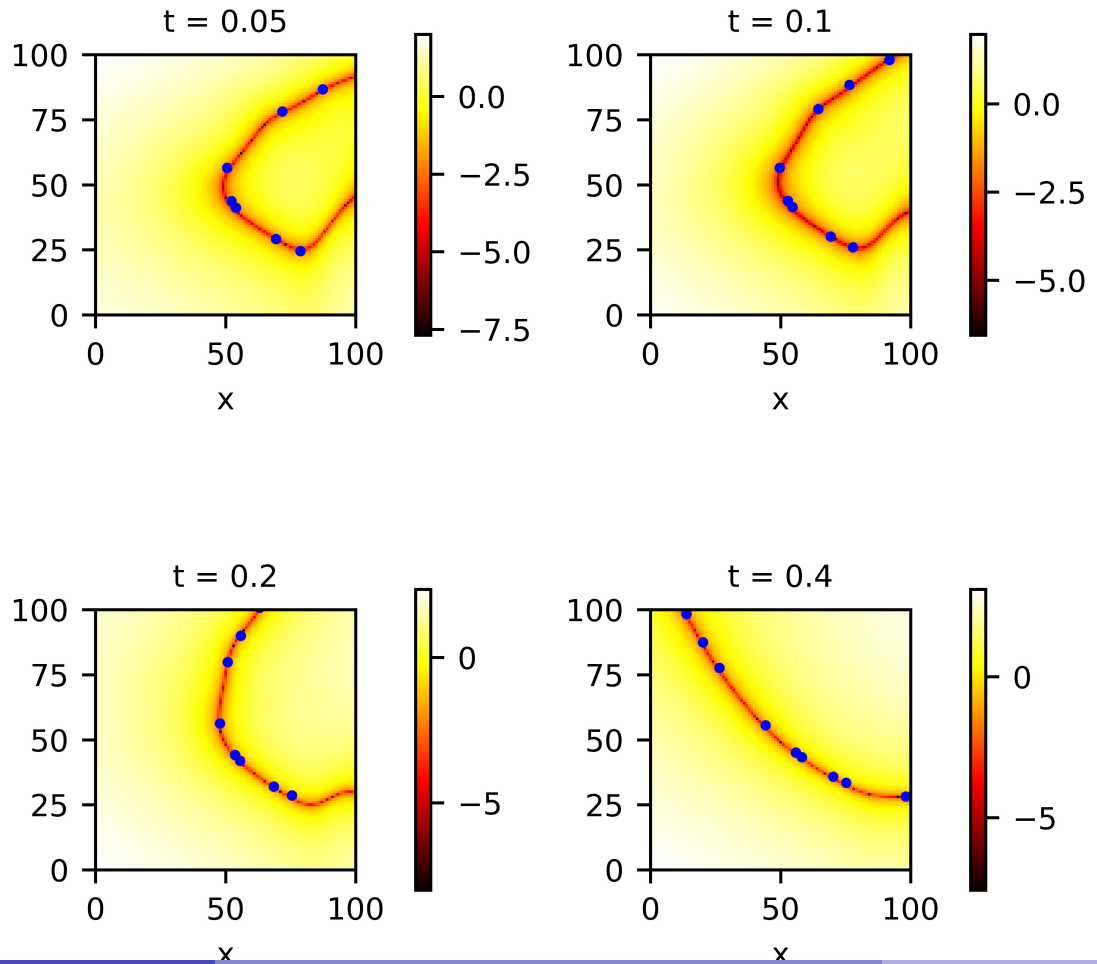
Averaged error is given by 0.173001.



$\widehat{\mathcal{M}}_t$

$$\min_{\theta} \mathbb{E} \left\| \mathbf{u} - \phi_\theta(\mathbf{x}, t) \right\|^2 + \lambda \left\| \widehat{\mathbf{u}} - \phi_\theta(\widehat{\mathbf{x}}, t) \right\|^2$$

Averaged error is given by 0.160128.



Reaction Diffusion equation

Consider following FitzHugh-Nagumo reaction diffusion (RD) equation:

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} &= \gamma \Delta \mathbf{u} + \mathbf{R}(\mathbf{u}), \quad T \in [0,1], \\ \mathbf{R}(\mathbf{u}) = \mathbf{R}(u,v) &= \begin{pmatrix} u - u^3 - v - \alpha \\ \beta(u-v) \end{pmatrix}, \end{aligned} \tag{5}$$

This provides a simple inner-outer loop structure for the numerical solver. 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 $\alpha = 0.01, \beta = 0.25$. We use mesh size 128×128 for the whole problems. Computational domain is given by $[0,1] \times [0,1]$.

Fine-coarse grid

Another loop structure can be formulated as follows:

$$I_n^{2n} : \mathbb{R}^{n\times n} \to \mathbb{R}^{2n\times 2n}$$
$$R_{2n}^{n} : \mathbb{R}^{2n\times 2n} \to \mathbb{R}^{n\times n}.$$

$$\begin{cases} \mathbf{u}_{2n}^{t+1} = I_n^{2n} \circ f(R_{2n}^{n}(\mathbf{u}_{2n}^{t}))) + \widehat{\mathbf{R}}(\mathbf{u}_{2n}^{t}), \\ \widehat{\mathbf{R}}(\mathbf{u}) = \phi_{NN}(u, v). \end{cases} \tag{6}$$

It is akin to the main idea of multigrid. Inner loop now represents the role of accounting for the residue of switching between two grid size.

Network architecture

We choose U-net for

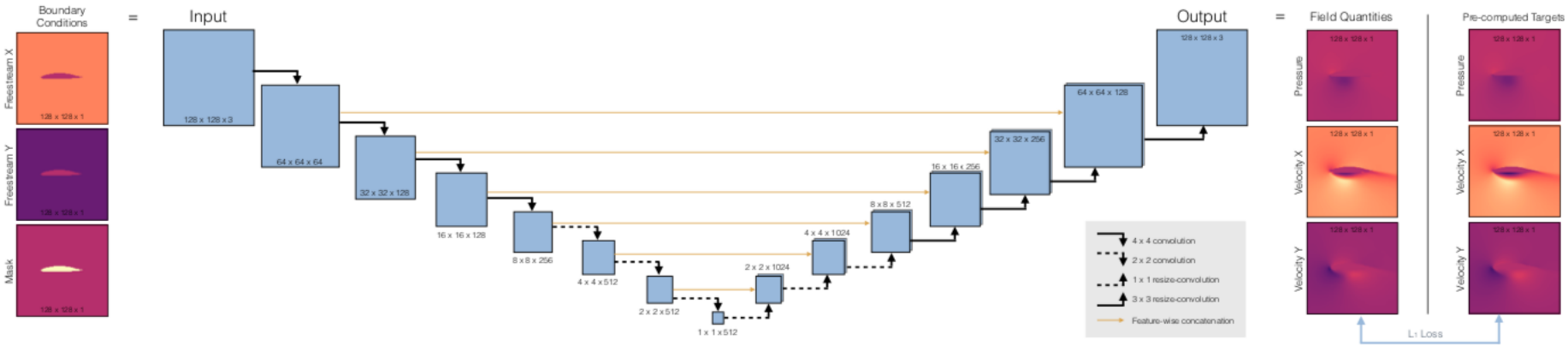


Figure: U-net structure for flow prediction[2]

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

Simulation

On average in 200 experiments, the regularized simulator improve by 40%.

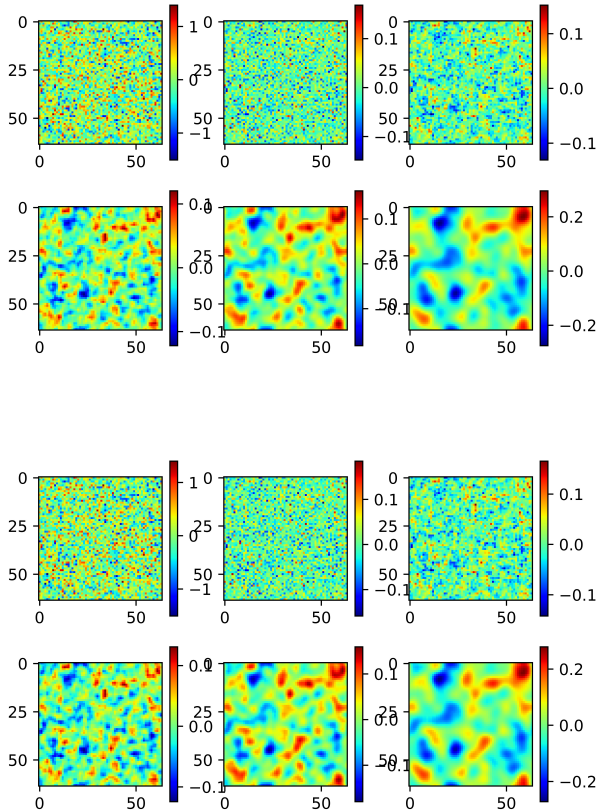


Figure: Upper: Benchmark with relative numerical error 12; Lower: Regularized sampler with numerical error 8.

Incompressible Navier-Stockes equation

Consider incompressible NS equation:

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} - \nu \Delta \mathbf{u} = \nabla p, \quad T \in [0, 1], \\ \nabla \cdot \mathbf{u} = 0, \end{aligned} \tag{7}$$

The computational domain is a rectangular $[0, 4] \times [0, 1]$. The boundary condition on upper and lower boundary is no-slip for velocity. The boundary condition for outlet is zero-gradient on pressure while we specify the inlet velocity.

Typical flow configuration

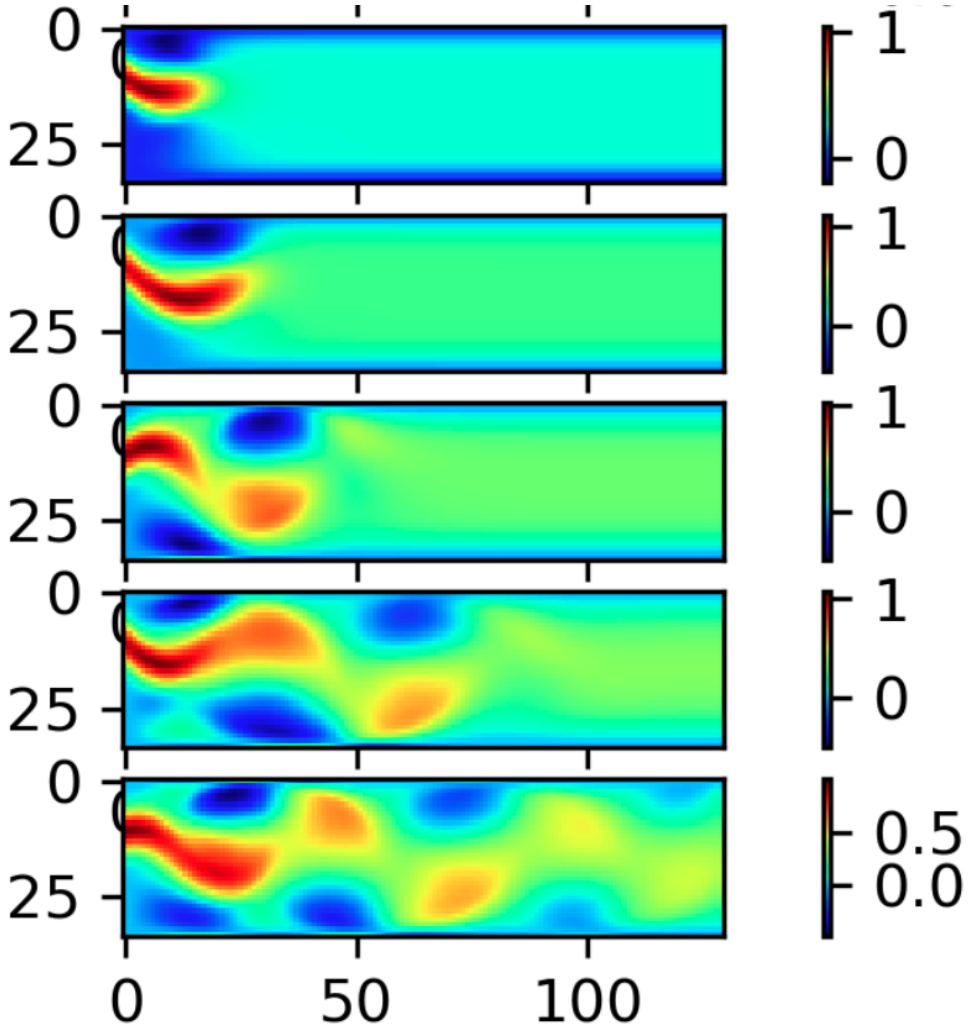


Figure: Simulation with time step 0.01 for time 20, mesh size 128×32

Prediction of the pressure

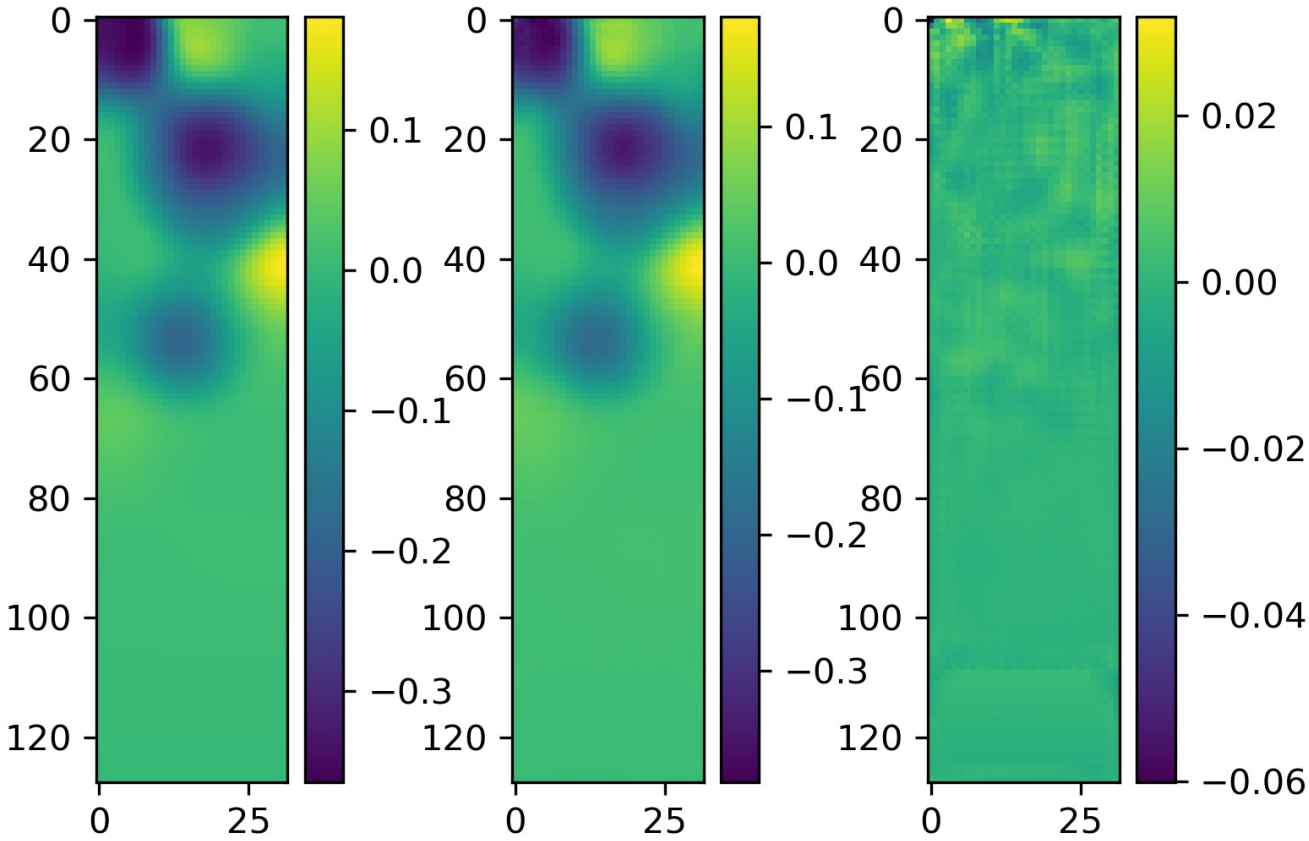


Figure: Prediction of pressure at time $t = 7$; Left: ground truth; Middle: prediction; Right: error

Future work: Algorithm

In the algorithmic perspective

1. Complete the work on encoder-decoder regularized method to mitigate the distribution mismatch in scientific computing.
2. Scientific computing + Manifold learning/self-supervised learning.

Future work: Application

In terms of application, we will further investigate the following thing:

1. Try to implant the best performed algorithm to the problems in fluid mechanics such as turbulence transition and flow separation.
2. Consider interesting problems in physics such as black holes dynamics, space-time classification in general relativity.