Neural ODEs: Approximation Theory and Operator Learning

Ziqian Li

School of Mathematics, Jilin University

Chair for Dynamics, Control, Machine Learning, and Numerics
Alexander von Humboldt-Professorship
Friedrich-Alexander-Universität Erlangen-Nürnberg

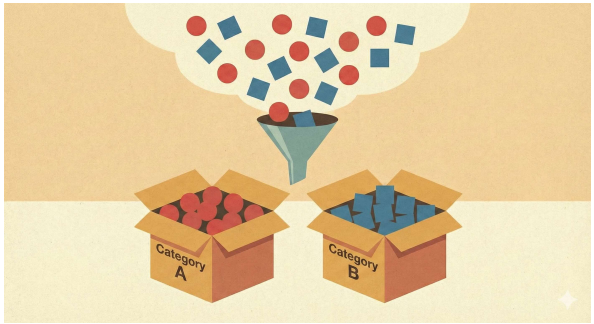
Fudan University
March 02, 2026

Outline

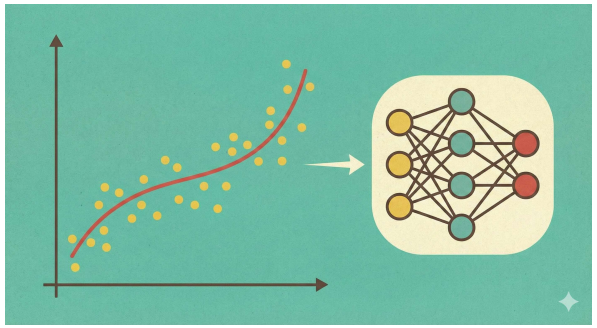
Outline

Examples of Machine Learning

Classification



Regression



Generation

- **Why it works** : Universal approximation property (UAP),

$$f(x) \approx f_\Theta(x);$$

- **How it works** : Optimization (training),

$$\inf_\Theta \sum_{i=1}^{N} \text{Loss}(f_\Theta(x_i), f(x_i)) + r(\Theta).$$

Figures are generated by Gemini 3.1 Pro.

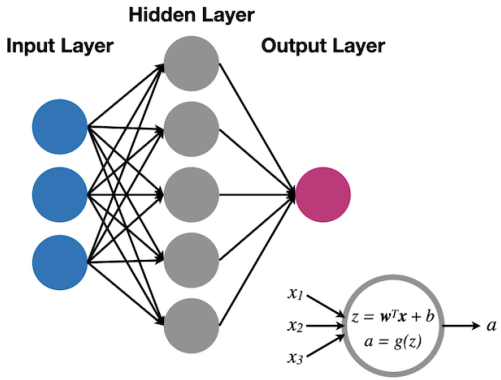
Shallow (One-hidden-layer) Neural Network

Formulation:

$$f(x;\Theta) = \sum_{i=1}^{P} w_i \sigma(\langle a_i, x\rangle + b_i),$$

where

- $\Theta = \{(w_i, a_i, b_i) \in \mathbb{R}^{d+2}\}_{i=1}^{P}$;
- σ is an activation function.



UAP: A Historical Overview

Qualitative Results

- Wiener (1932)
- Cybenko (1989)
- Hornik (1991)
- ...
- Pinkus (1999)

Quantitative Bounds

- Barron (1993)
- Bach (2017)
- Klusowski-Barron (2018)
- E–Ma–Wu (2022)
- Siegel-Xu (2024)...

Neural ODEs

The neural ODEs (NODEs) (E 2017, Chen et al. 2018) read

$$\begin{cases} \dot{\boldsymbol{x}} = \displaystyle\sum_{i=1}^{P} W_i(t) \circ \boldsymbol{\sigma}(A_i(t)\boldsymbol{x} + B_i(t)), \quad \boldsymbol{x} \in \mathbb{R}^d, t \in [0,T], \\ \boldsymbol{x}(0) = x_0 \in \mathbb{R}^d. \end{cases}$$

Here $\circ$ is the Hadamard product $(a,b) \circ (c,d) = (ac, bd)$, and P is the width of the neural network. The parameters $A_i(t)$, $W_i(t)$ and $B_i(t)$ depend on time.

Related works:

- Controllability: [E 2017], [Agrachev et al. 2022], [Geshkovski et al. 2022]
- Approximation: [Osher et al. 2022], [Álvarez-López et al. 2024]
- Long-time behavior: [Geshkovski et al. 2022]
- Formal limit of ResNets: [Massaroli et al. 2020], [Sander et al. 2022]

Outline

Contents

Neural ODEs

The neural ODEs (NODEs) read

$$\begin{cases} \dot{\boldsymbol{x}} = \sum_{i=1}^{P} W_i(t) \circ \boldsymbol{\sigma}(A_i(t)\boldsymbol{x} + B_i(t)), \quad \boldsymbol{x} \in \mathbb{R}^d, t \in [0, T], \\ \boldsymbol{x}(0) = x_0 \in \mathbb{R}^d. \end{cases}$$

Disadvantages of NODEs:

- The number of parameters $((d+3)dMP)$ scales as the number of time steps $M \Rightarrow$ High complexity
- Impossible to calculate the solution after T

Motivation:

- Can NODE approximate an ODE system for $t \in [0, T]$?
- How to compute $\boldsymbol{x}(t)$ after T?

SA-NODEs

The semi-autonomous neural ODEs (SA-NODEs) read

$$\begin{cases} \dot{\boldsymbol{x}} = \sum_{i=1}^{P} W_i \circ \boldsymbol{\sigma}(A_i^1 \boldsymbol{x} + A_i^2 t + B_i), \quad \boldsymbol{x} \in \mathbb{R}^d, t \in [0, T], \\ \boldsymbol{x}(0) = x_0 \in \mathbb{R}^d. \end{cases}$$

Here the parameters W_i, A_i^1, A_i^2 and B_i are constant matrixes.

Advantages of SA-NODEs:

- The number of parameters $(Pd(d+3))$ is independent of time steps.
  $\Rightarrow$ Low complexity
- Able to calculate the solution after T

Contents

Universal Approximation Theory

To approximate an ODE system

$$\begin{cases} \dot{\boldsymbol{z}}(t) = f(\boldsymbol{z}(t), t), \quad t \in [0, T], \\ \boldsymbol{z}(0) = \boldsymbol{z}_0, \end{cases}$$

by SA-NODEs, we obtain the universal approximation theory

Theorem (Li, Liu, L., Zuazua, 2024)

Let f be uniformly Lipschitz in $\boldsymbol{z}$ with respect to t. For any compact set $K \subseteq \mathbb{R}^d$ and any $\varepsilon > 0$, there exists a constant $P_{\varepsilon,T,K,f}$ such that for any $P \geq P_{\varepsilon,T,K,f}$, there exist parameters $(W_i, A_i^1, A_i^2, B_i) \in \mathbb{R}^d \times \mathbb{R}^{d\times d} \times \mathbb{R}^d \times \mathbb{R}^d$, for $i = 1, \ldots, P$, such that

$$\|\boldsymbol{z}_{z_0}(\cdot) - \boldsymbol{x}_{z_0}(\cdot)\|_{\mathbb{L}^\infty([0,T];\mathbb{R}^d)} \leq \varepsilon, \quad \forall z_0 \in K.$$

Approximation Rate

Further, we obtain the approximation rate with respect to the number of neurons P

Theorem (Li, Liu, L., Zuazua, 2024)

Let $f \in \mathcal{H}^k_{\text{loc}}(\mathbb{R}^d \times [0,T]; \mathbb{R}^d)$, for $k > (d+1)/2 + 2$. Fix any compact set $K \subseteq \mathbb{R}^d$. Then, for any $P \geq 3$, there exist parameters $(W_i, A_i^1, A_i^2, B_i) \in \mathbb{R}^d \times \mathbb{R}^{d\times d} \times \mathbb{R}^d \times \mathbb{R}^d$, for $i = 1, \ldots, P$, such that

$$\|\boldsymbol{z}_{z_0}(\cdot) - \boldsymbol{x}_{z_0}(\cdot)\|_{\mathbb{L}^\infty([0,T];\mathbb{R}^d)} \leq \frac{C_{T,K,f}}{\sqrt{P}},, \quad \forall z_0 \in K,$$

where $C_{T,K,f}$ is a constant independent of P.

Transport Equations

The transport equation of divergence form (continuity equation):

$$\begin{cases} \partial_t \rho(x,t) + \mathrm{div}_x(f(x,t)\rho(x,t)) = 0, & (x,t) \in \mathbb{R}^d \times [0,T], \\ \rho(x,0) = \rho_0(x), & x \in \mathbb{R}^d. \end{cases}$$

The characteristic system of the transport equation is

$$\begin{cases} \dfrac{d}{dt}\begin{pmatrix} X \\ \rho \end{pmatrix} = \begin{pmatrix} f(X,t) \\ -\mathrm{div}_x(f(X,t))\rho \end{pmatrix}, & t \in [0,T], \\ \begin{pmatrix} X(0) \\ \rho(0) \end{pmatrix} = \begin{pmatrix} x_0 \\ \rho_0(x_0) \end{pmatrix}. \end{cases}$$

Transport Equations

The approximated characteristic system:

$$\begin{cases} \dfrac{d}{dt}\begin{pmatrix} X_\Theta \\ \rho_\Theta \end{pmatrix} = \begin{pmatrix} f_\Theta(X_\Theta, t) \\ -\mathrm{div}_x(f_\Theta(X_\Theta, t))\rho_\Theta \end{pmatrix}, \quad t \in [0, T], \\ \begin{pmatrix} X_\Theta(0) \\ \rho_\Theta(0) \end{pmatrix} = \begin{pmatrix} x_0 \\ \rho_0(x_0) \end{pmatrix}. \end{cases}$$

The corresponding neural transport equation [Ruiz-Balet, Zuazua 2024]:

$$\begin{cases} \partial_t \rho_\Theta(x,t) + \mathrm{div}_x\left(f_\Theta(x,t)\rho_\Theta(x,t)\right) = 0, \quad (x,t) \in \mathbb{R}^d \times [0,T], \\ \rho_\Theta(x,0) = \rho_0(x), \quad x \in \mathbb{R}^d. \end{cases}$$

Transport Equations

We obtain the approximation theory of transport equations

Theorem (Li, Liu, L., Zuazua, 2024)

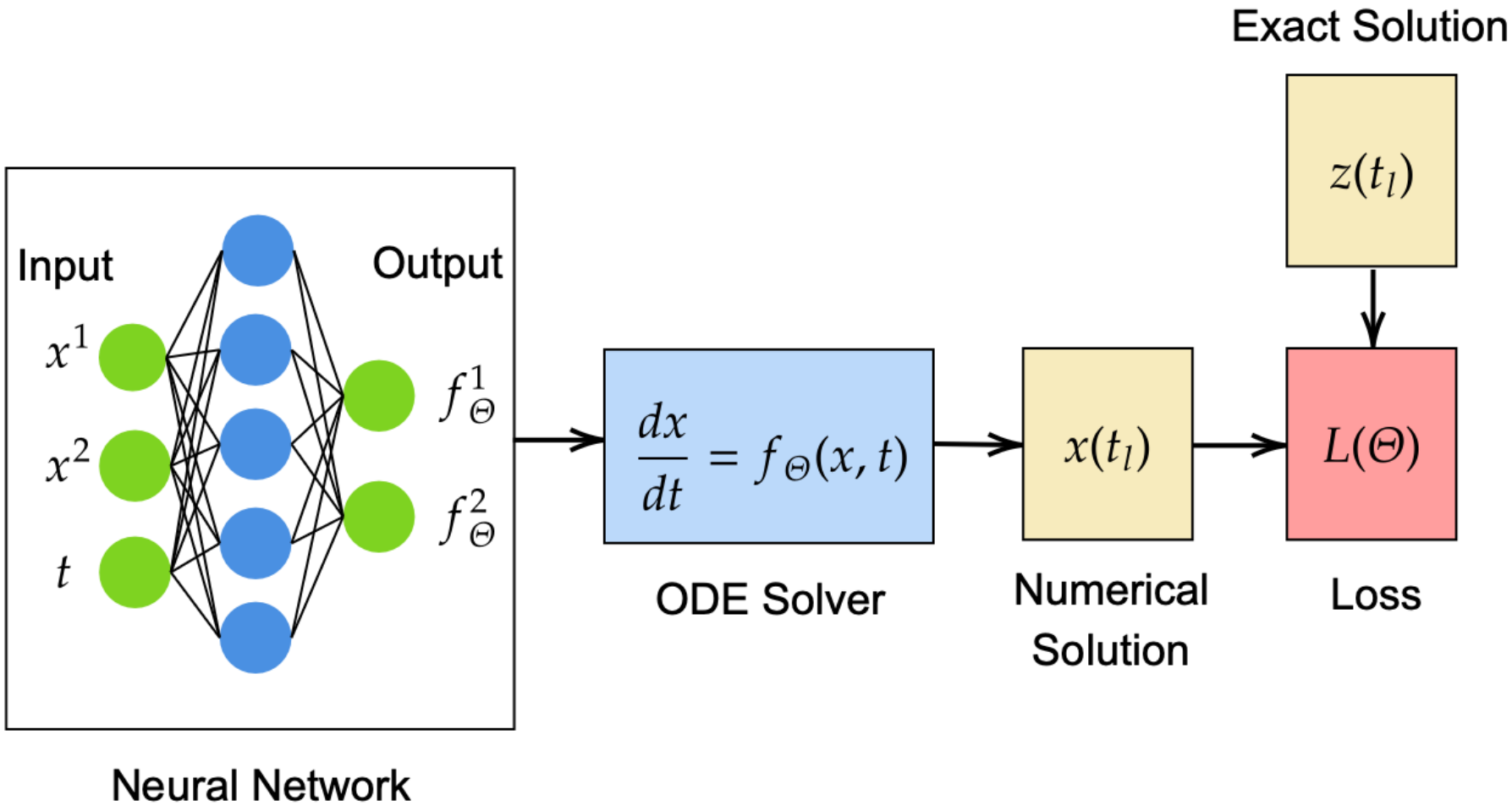
Let ρ_0 be a probability measure supported in a compact set K such that $\rho_0 \in \mathbb{L}^2(K)$. Then, for any $P \in \mathbb{N}_+$, there exist parameters $\Theta = \{(W_i, A_i^1, A_i^2, B_i)\}_{i=1}^P$ such that

$$\sup_{t\in[0,T]} \mathbb{W}_1(\rho(\cdot,t), \rho_\Theta(\cdot,t)) \leq \frac{C_{T,f,\rho_0}}{\sqrt{P}},$$

where C_{T,f,ρ_0} is a constant independent of P, $\mathbb{W}_1(\cdot,\cdot)$ is the Wasserstein-1 distance, and $\rho(\cdot,t)$ (resp. $\rho_\Theta(\cdot,t)$) is the solution of the transport equation (resp. the neural transport equation) at the time $t \in [0,T]$.

Contents

Approximating ODEs: Workflow



Approximating ODEs: Training Dataset

For an ODE system

$$\begin{cases} \dot{\boldsymbol{z}}(t) = f(\boldsymbol{z}(t), t), \quad t \in [0, T], \\ \boldsymbol{z}(0) = \boldsymbol{z}_0. \end{cases}$$

Data: N trajectories $\mathcal{D} = \{\boldsymbol{z}_k(\cdot)\}_{k=1}^N \subset C([0, T]; \mathbb{R}^d)$.

In practice: $\mathcal{D} = \{\boldsymbol{z}_k(t_l)\}_{k,l} \subset \mathbb{R}^d$, for $k = 1, \ldots, N$, $l = 1, \ldots, M$.

Approximating ODEs: Loss Function

Data: $\mathcal{D} = \{z_k(t_l)\}_{k,l} \subset \mathbb{R}^d$, for $k = 1, \dots, N$, $l = 1, \dots, M$.

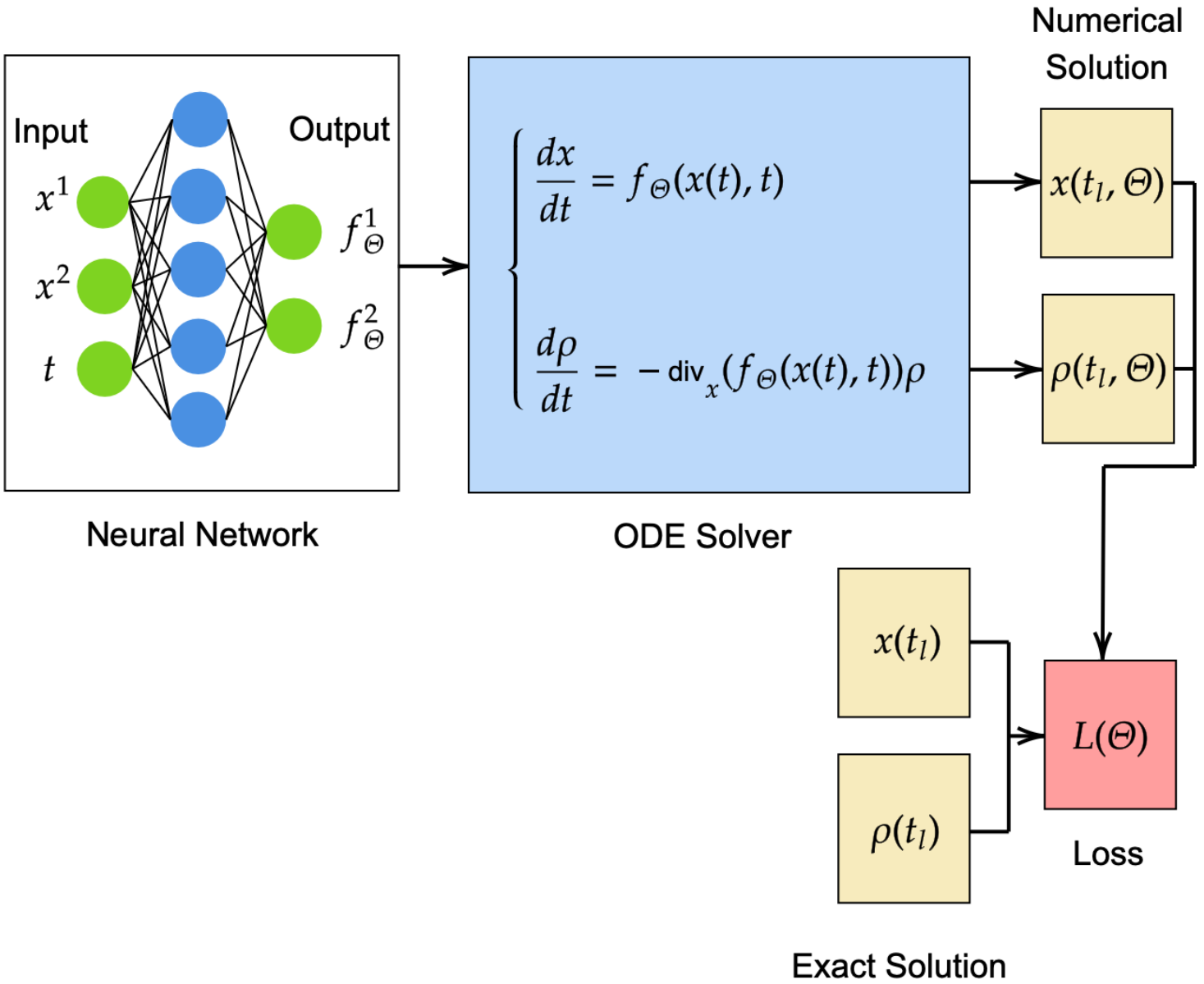
Lipschitz Constant: $\left\| \sum_{i=1}^{P} |W_i| \circ \|A_i^1\|_{\ell^2} \right\|$

Loss Function:

$$L(\Theta) = \frac{1}{NM} \sum_{k=1}^{N} \sum_{l=1}^{M} \left(z_k(t_l) - x_k(t_l, \Theta)\right)^2 + \lambda \left\| \sum_{i=1}^{P} |W_i| \circ \|A_i^1\|_{\ell^2} \right\|$$

$\rightsquigarrow$ Stochastic gradient descent

Approximating Transport: Workflow



Approximating Transport: Data & Loss Function

Data: $\mathcal{D} = \{x_k(t_l), \rho_k(t_l)\}, k = 1, 2, \cdots, N,\ l = 1, 2, \cdots, M.$

Loss Function:

$$L(\Theta) = \frac{1}{NM} \sum_{k=1}^{N} \sum_{l=1}^{M} \Big((x_k(t_l) - x_k(t_l, \Theta))^2 + (\rho_k(t_l) - \rho_k(t_l, \Theta))^2 \Big) + \lambda \left\| \sum_{i=1}^{P} |W_i| \circ \|A_i^1\|_{\ell^2} \right\|,$$

$\rightsquigarrow$ Stochastic gradient descent

Contents

ODEs: Autunomous & Nonlinear ODEs

Approximate autonomous and nonlinear ODE system:

$$\begin{cases} \dot{z}_1 = z_2, \\ \dot{z}_2 = -\sin(z_1). \end{cases}$$

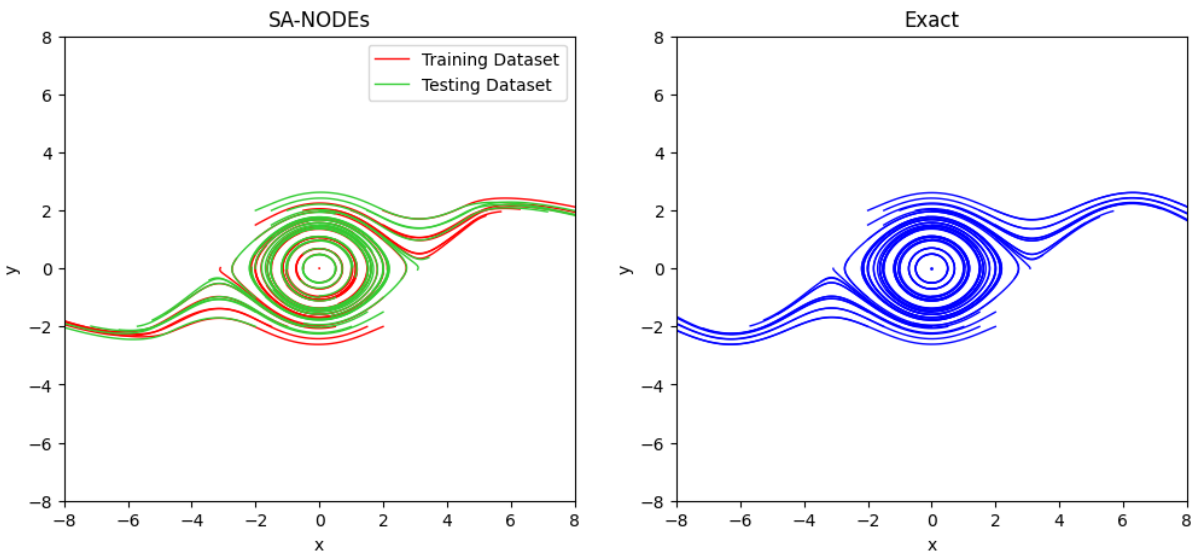


Figure: SA-NODEs and exact solution

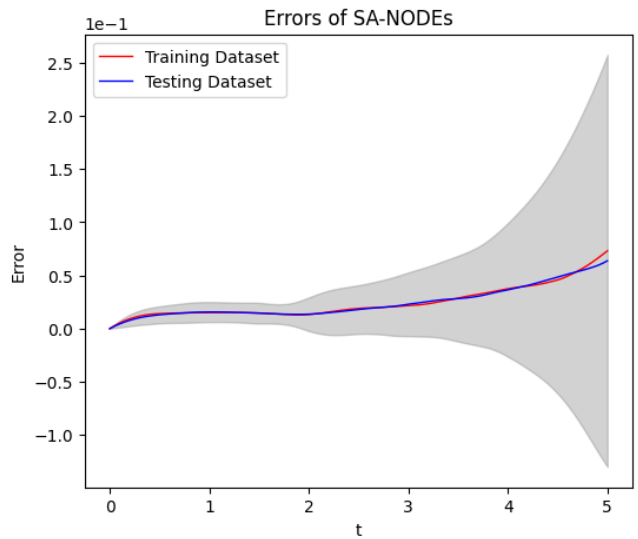


Figure: Errors

ODEs: Non-Autunomous & Nonlinear Case

Approximate non-autonomous and nonlinear ODE system:

$$\begin{cases} \dot{z}_1 = z_2, \\ \dot{z}_2 = z_1 - z_1^3 + \delta \cos(\omega t). \end{cases}$$

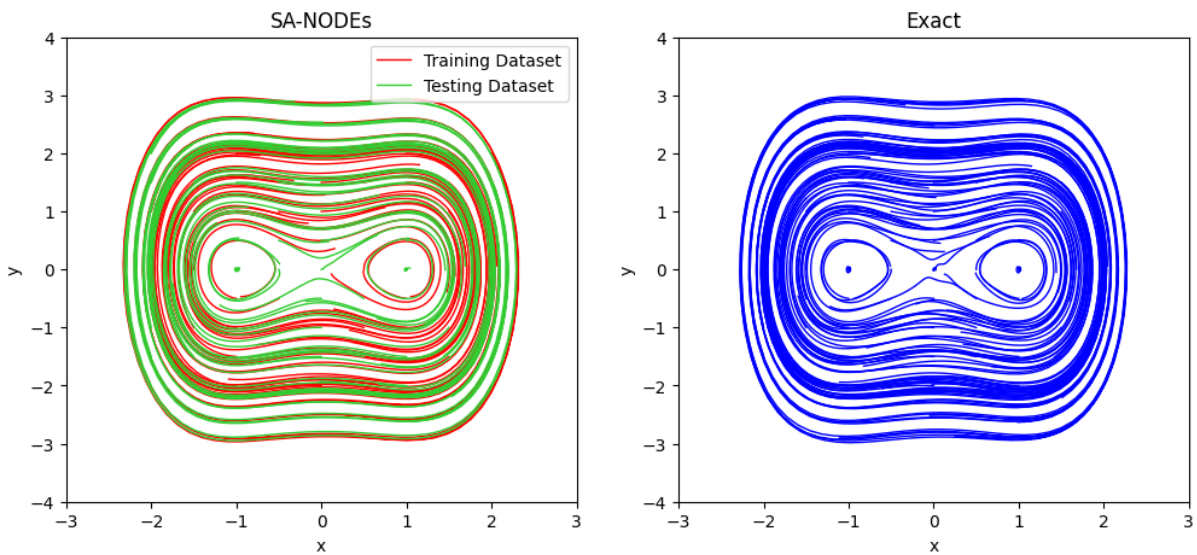


Figure: SA-NODEs and exact solution

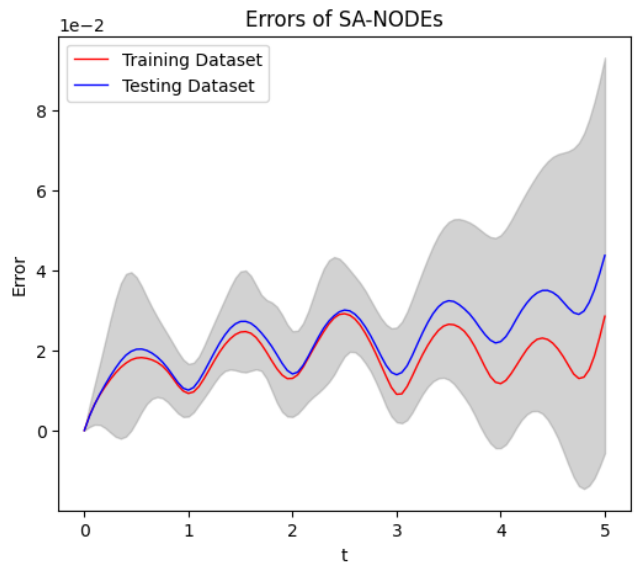
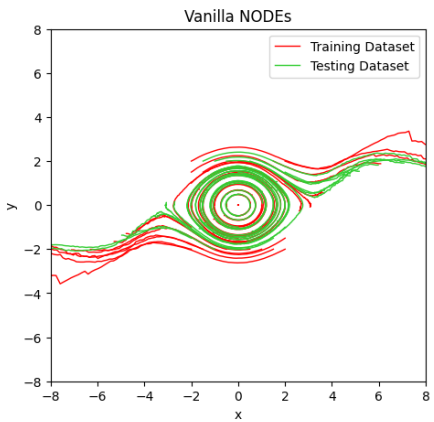


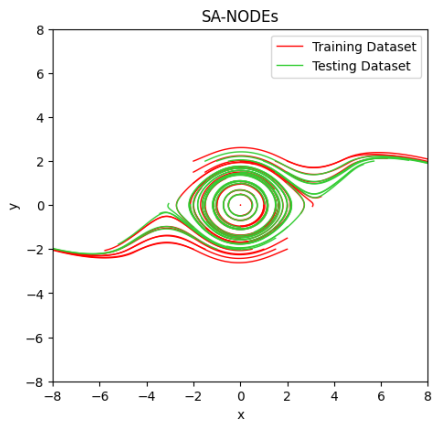
Figure: Errors

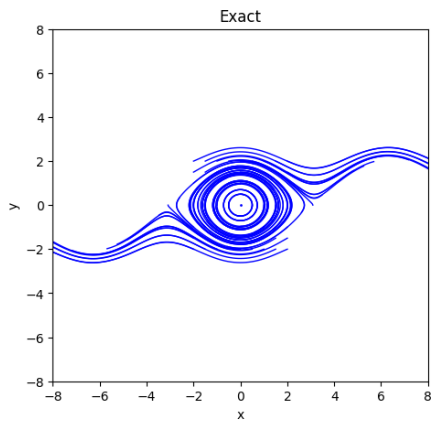
Comparison with Vanilla NODEs

Approximate autonomous and nonlinear ODE system:

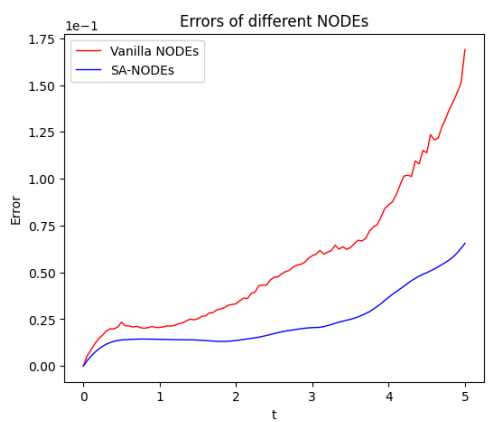
$$\begin{cases} \dot{z}_1 = z_2, \\ \dot{z}_2 = -\sin(z_1). \end{cases}$$







(a) Vanilla NODEs, SA-NODEs and exact solution.



(b) Testing errors.

Figure: Comparison of vanilla NODEs and SA-NODEs.

Comparison with Vanilla NODEs

Approximate autonomous and nonlinear ODE system:

$$\begin{cases} \dot{z}_1 = z_2, \\ \dot{z}_2 = -\sin(z_1). \end{cases}$$

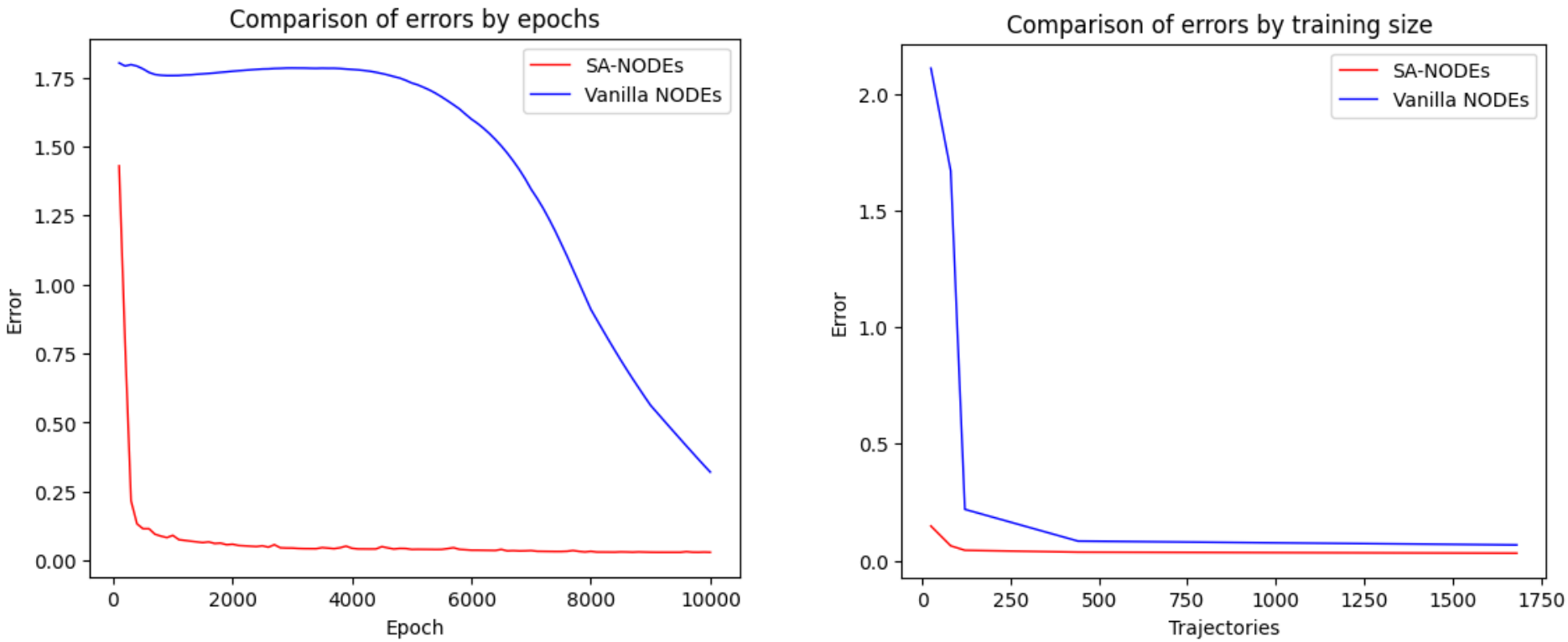


Figure: Comparison on epochs and training size.

Transport: Non-Autonomous Case

For the non-autonomous transport equation:

$$\begin{cases} \partial_t \rho(x,y,t) + \operatorname{div}\left(\left(\frac{\sin(x)}{1+t^2}, \frac{\sin(y)}{1+t^2}\right)\rho(x,y,t)\right) = 0, \\ \rho(\cdot,0) = \rho_0. \end{cases}$$

Initial measure for trainxing:

$$\rho_0^{\text{train}}(x,y) = 0.5.$$

Initial measure for testing:

$$\rho_0^{\text{test}}(x,y) = e^{-\frac{x^2+y^2}{4}}.$$

Error for testing:

$$e_{\text{test}}(t) = \frac{\|\rho_\Theta(\cdot,t) - \rho(\cdot,t)\|_{\mathbb{L}^1(\mathbb{R}^2)}}{\|\rho(\cdot,0)\|_{\mathbb{L}^1(\mathbb{R}^2)}}.$$

Transport: Non-Autonomous Case

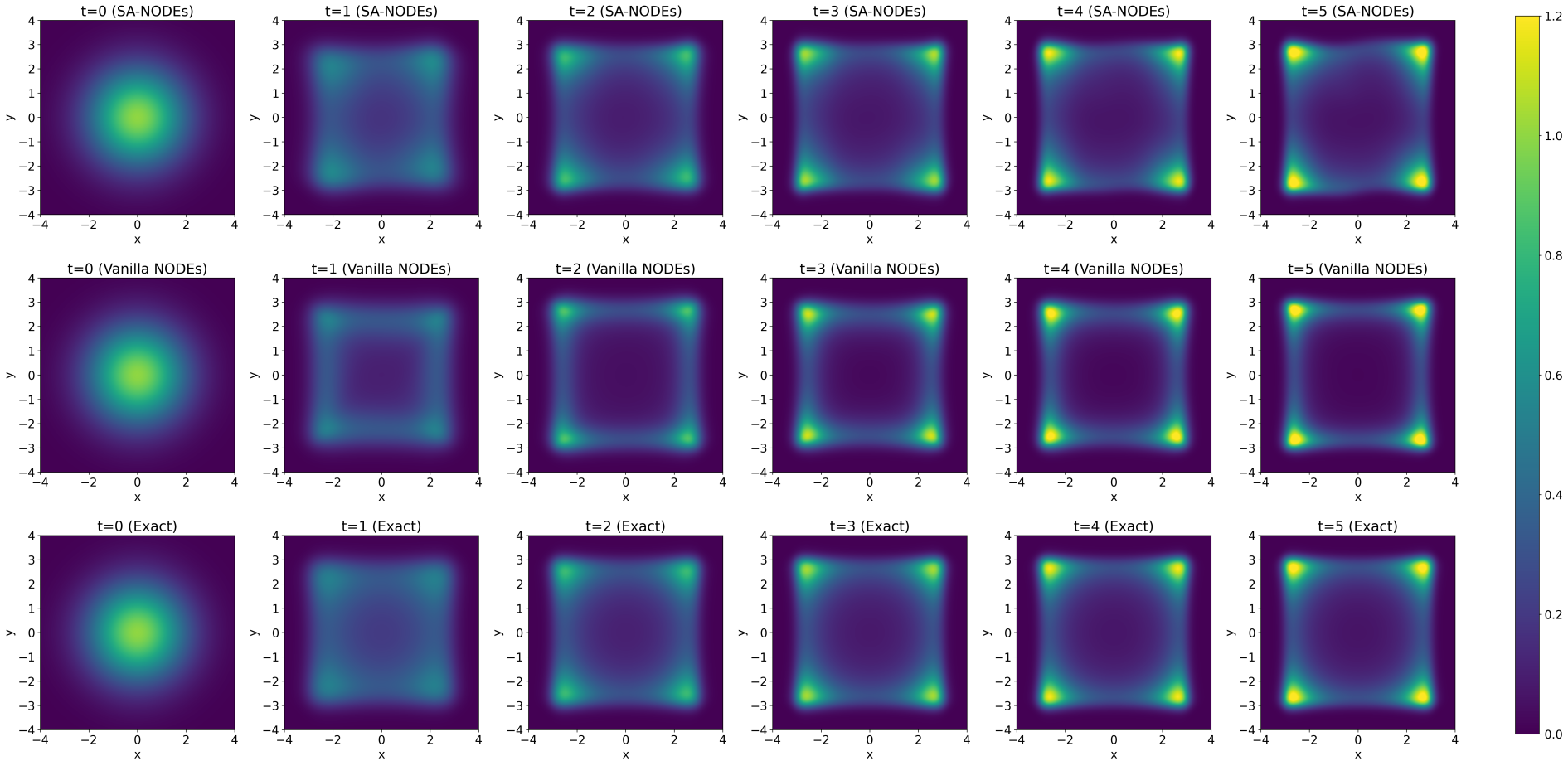


Figure: SA-NODEs, vanilla NODEs and exact solutions

Transport: Non-Autonomous Case

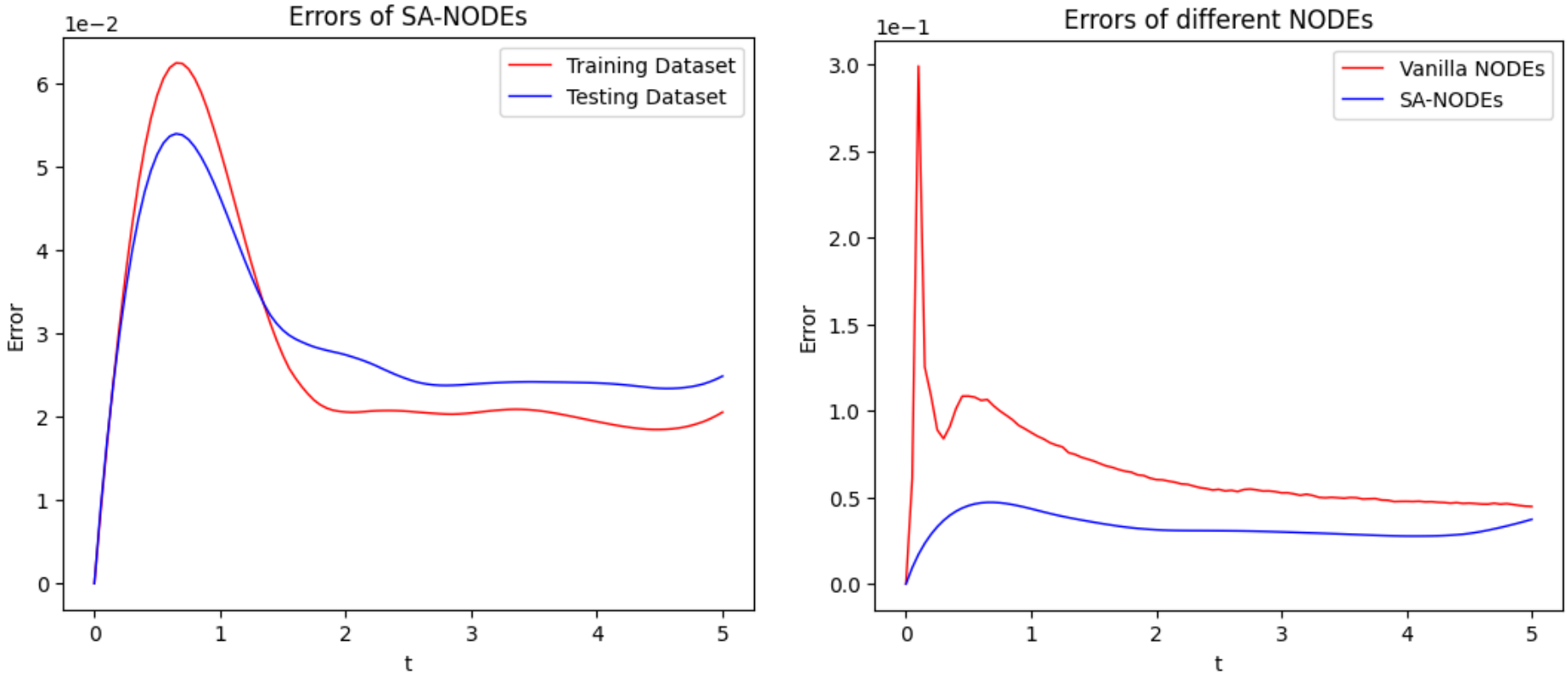


Figure: Errors of SA-NODEs and vanilla NODEs on transport equation.

Z. Li, K. Liu, L. Liverani, E. Zuazua. *Universal Approximation of Dynamical Systems by Semiautonomous Neural ODEs and Applications*. SIAM Journal on Numerical Analysis, 64(1), 2026.

Outline

PDEs and Traditional Numerical solvers

- We consider a generic class of PDEs modeled by

$$\begin{cases} \partial_t u(t,x) + \mathcal{L}[a](u)(t,x) = f(t,x) & \forall (t,x) \in [0,T] \times \Omega, \\ u(0,x) = u_0(x) & \forall x \in \Omega, \\ \mathcal{B}u(t,x) = u_b(t,x) & \forall (t,x) \in [0,T] \times \partial\Omega. \end{cases}$$

  A numerical solver of the PDE tries to find the numerical approximation of

$$(t,x) \to u \approx u_\theta.$$

- **Traditional Numerical Methods:**
  - Finite Element Method (FEM),
  - Finite Difference Method (FDM),
  - Finite Volumn Method (FVM).
- **Challenges of Traditional Methods:**
  - PDEs in high-dimensional spaces and complex domains.
  - Problems requiring repeated but expensive simulations, e.g., inverse problems and optimal control of PDEs.

Scientific Machine Learning for PDEs - function learning

Solution learning methods: using NNs to approximate the solution function of a PDE

- Deep Ritz method [E and Yu, 2017]
- Deep Galerkin method [Sirignano and Spiliopoulos, 2017]
- Physics-informed neural networks (PINNs) [Raissi, Perdikaris, and Karniadakis, 2017]
- Weak adversarial networks [Zang, Bao, Ye, and Zhou, 2019]
- Random feature methods [Chen, Chi, E, and Yang, 2022]
- ... ...

Typical applications:

- High-dimensional PDEs.
- Complex geometries.
- ... ...

Operator Learning

We consider a class of non-stationary PDEs modeled by

$$\begin{cases} \partial_t u(t,x) + \mathcal{L}[a](u)(t,x) = f(t,x) & \forall (t,x) \in [0,T] \times \Omega, \\ u(0,x) = u_0(x) & \forall x \in \Omega, \\ \mathcal{B}u(t,x) = u_b(t,x) & \forall (t,x) \in [0,T] \times \partial\Omega. \end{cases}$$

- **Parameters:** $v \subset \{f, a, u_0, u_b\}$.
- **Goal of operator learning:**

$$\Psi_\theta \approx \Psi^\dagger : v \mapsto u,$$

where $\Psi^\dagger$ is the solution operator , which maps v to the solution u, by a neural-network–based functional Ψ_θ with trainable parameters θ.

Scientific Machine Learning for PDEs - operator learning

Operator learning methods: using NNs to approximate the solution operator of a PDE

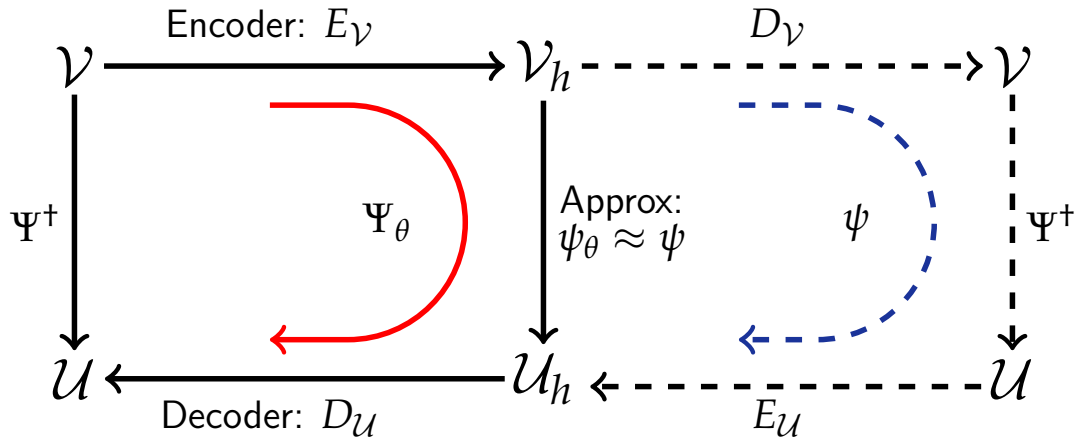
- Deep Operator Networks (DeepONets) [Lu, Jin, Pang, Zhang, and Karniadakis, 2019],
- Physics-Informed DeepONets [Wang, Wang, and Perdikaris, 2021]
- Fourier Neural Operator, Graph Neural Operator, [Li, Kovachki, Azizzadenesheli, Liu, Bhattacharya, Stuart, and Anandkumar,2021]
- The Random Feature Approach [Nelsen and Stuart, 2021]
- In-Context Operator Networks [Yang, Liu, Meng, and Osher, 2024]
- ... ...

Typical applications:

- PDEs discovery: model and predict unknown physics through data.
- Acceleration: speed-up computationally expensive simulations.
- ... ...

Encoder-decoder-net architectures

- Learning an infinite-dimensional operator $\Psi^\dagger : \mathcal{V} \to \mathcal{U}$.



- An encoder-decoder pair $(E_\mathcal{V}, D_\mathcal{V})$ on $\mathcal{V}$, i.e., $E_\mathcal{V} : \mathcal{V} \to \mathbb{R}^{d_\mathcal{V}}$, $D_\mathcal{V} : \mathbb{R}^{d_\mathcal{V}} \to \mathcal{V}$,
- An encoder-decoder pair $(E_\mathcal{U}, D_\mathcal{U})$ on $\mathcal{U}$, i.e., $E_\mathcal{U} : \mathcal{U} \to \mathbb{R}^{d_\mathcal{U}}$, $D_\mathcal{U} : \mathbb{R}^{d_\mathcal{U}} \to \mathcal{U}$.
- These encoder/decoder pairs imply a finite-dimensional function

$$\psi : \mathcal{V}_h \to \mathcal{U}_h, \quad \psi(\zeta) = E_\mathcal{U} \circ \Psi^\dagger \circ D_\mathcal{V}(\zeta), \quad \forall \zeta \in \mathcal{V}_h$$

  and then $D_\mathcal{U} \circ \psi \circ E_\mathcal{V} \approx \Psi^\dagger$.

- Using a neural network $\psi_\theta : \mathcal{V}_h \to \mathcal{U}_h$ to approximate ψ, we obtain an encoder-decoder network $\Psi_\theta : \mathcal{V} \to \mathcal{U}$:

$$\Psi_\theta := D_\mathcal{U} \circ \psi_\theta \circ E_\mathcal{V} \approx \Psi^\dagger.$$

Outline

Contents

A concrete example: Heat equation

We consider the 1D heat equation

$$\begin{cases} \partial_t u(t,x) - \Delta u(t,x) = 0, \quad \forall (t,x) \in [0,1] \times [0,1], \\ u(t,0) = u(t,1) = 0, \quad \forall t \in [0,1], \\ u(0,x) = u_0(x), \quad \forall x \in [0,1]. \end{cases}$$

The obective is to use a neural-network–based functional Ψ_θ to learn the mapping:

$$\Psi_\theta \approx \Psi^\dagger : u_0 \mapsto u.$$

Learn from numerical solutions with different initial conditions. (need discretization)

Step 1: Encoding (Discretization and preparing training dataset)

We discretize space and time into uniform mesh:

- Space: $[0,1] \mapsto N_x$ points: $(x_1, x_2, \ldots, x_{N_x})$
- Time: $[0,1] \mapsto N_T$ points: $(t_1, t_2, \ldots, t_{N_T})$

Let N_{u_0} denote the number of samples of u_0, then the training dataset is:

$$\{U_{0,h}^k, U_h^k\}, \quad k = 1, 2, \ldots, N_{u_0}$$

where $U_{0,h} = \left(u_0(x_i)\right)_{i=1}^{N_x} \in \mathbb{R}^{N_x}$, $U_h^k = \left(u(t_j, x_i)\right)_{i=1,j=1}^{N_x,N_T} \in \mathbb{R}^{N_x \times N_T}$. All the training data is computed by the Finite Difference Method (FDM).

Step 2: NODE surrogate (Approximation and training)

Suppose that U_h^k can be seen as values at different times of the solutions of an ODE with initial data $U_{0,h}^k$. We use U_θ to approximate U_h and we present this ODE by the approximation of the following NODE:

$$\begin{cases} \dfrac{d}{dt}U_\theta = \displaystyle\sum_{p=1}^{P} w_p \circ \sigma(A_p^1 U_\theta + A_p^2 t + B_p), \\ U_\theta(0,x) = U_{0,h} \in \mathbb{R}^n. \end{cases}$$

Then the loss function to train the NODE is:

$$L(\theta) = \frac{1}{N_v N_x N_T} \sum_{k=1}^{N_v} \sum_{i=1}^{N_x} \sum_{j=1}^{N_T} \left| \left(U_\theta^k(t_j)\right)_i - U_h^k(x_i,t_j) \right|^2 + \mathcal{R}(\theta).$$

Here, $\mathcal{R}(\theta)$ is a general regularization term.

Step 3: Decoding (Reconstructe PDE solution)

After the latent dynamics are learned, the PDE solution u is reconstructed from the NODE outputs u_θ

$$u(t,x) \approx u_\theta(t,x) \;=\; \sum_{i=1}^{N_x} \big(U_\theta(t)\big)_i \, \alpha_i(x), \quad \forall (t,x) \in [0,1] \times [0,1],$$

where α_i is the P_1-FEM basis centered at x_i.

Contents

General framework for $v \mapsto u$

For a class of non-stationary PDEs modeled by

$$\begin{cases} \partial_t u(t,x) + \mathcal{L}[a](u)(t,x) = f(t,x) & \forall (t,x) \in [0,T] \times \Omega, \\ u(0,x) = u_0(x) & \forall x \in \Omega, \\ \mathcal{B}u(t,x) = u_b(t,x) & \forall (t,x) \in [0,T] \times \partial\Omega. \end{cases}$$

The parameters of the PDE is defined by $v \subset \{f, a, u_0, u_b\}$. The architecture of the NODE-ONet:

Architecture

$$\begin{cases} \text{Encoding:} \quad E_{\mathcal{V}}(v) := \{v_\ell(t)\}_{\ell=1}^{d_{\mathcal{V}}} \in \mathbb{R}^{d_{\mathcal{V}}} \text{ for any } t \in [0,T]; \\ \text{Physics-encoded NODE:} \quad \begin{cases} \dot{\boldsymbol{\psi}}(t) = \mathcal{N}_{\theta_\psi}(\boldsymbol{\psi}(t), \mathcal{P}_v \boldsymbol{v}(t), t), \\ \boldsymbol{\psi}(0) = \mathcal{P}_u \boldsymbol{u}_0 \in \mathbb{R}^{d_{\mathcal{U}}}, \end{cases} \\ \text{Decoding:} \quad \Psi_{\text{NODE-ONet}}(v;\theta)(t,x) = D_{\mathcal{U}}(t,\alpha) = \sum_{j=1}^{d_{\mathcal{U}}} \alpha_j(x)\psi_j(t), \end{cases}$$

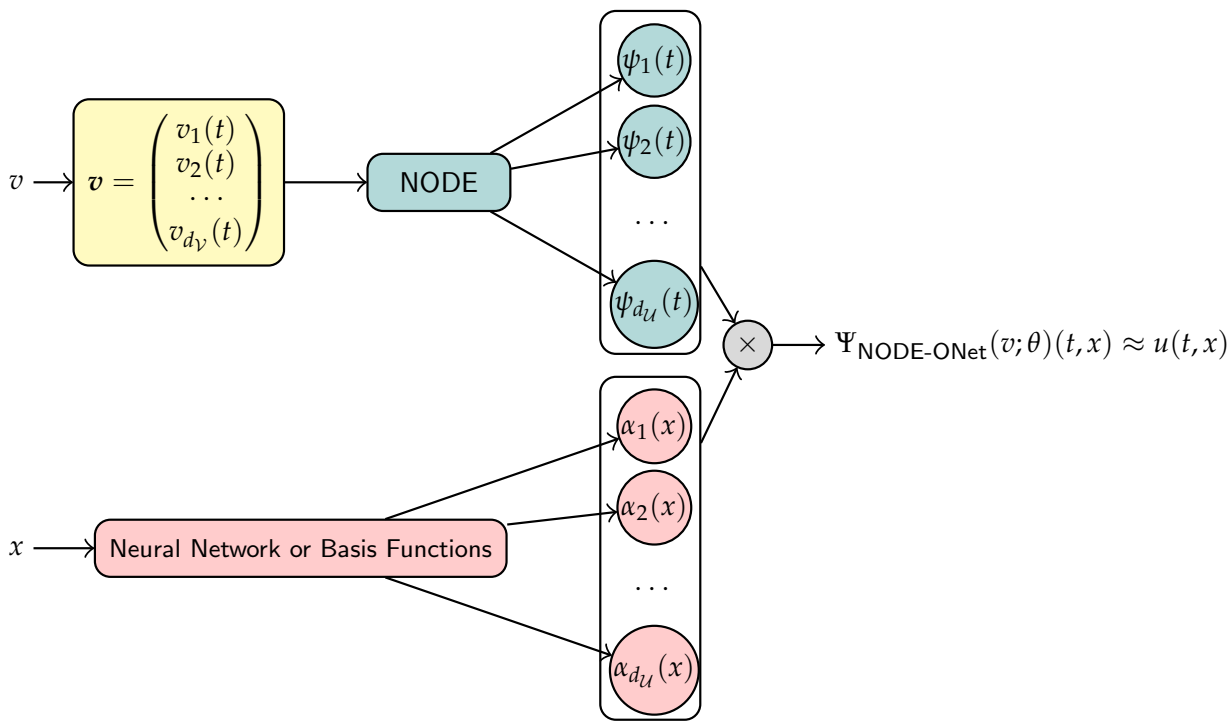
with $\{\alpha_j\}_{j=1}^{d_{\mathcal{U}}}$ a set of spatial basis functions.

General framework for $v \mapsto u$

The training setting is summarized as follows:

$$\textbf{Training:} \begin{cases} \text{Dataset: } \{v_i, x_j, \Psi^\dagger(v_i)(t_k, x_j)\}_{1\le i\le N_v, 1\le k\le N_t, 1\le j\le N_x} \\ \text{Loss function: } \mathcal{L}(\theta) = \\ \dfrac{1}{N_v N_x N_t} \displaystyle\sum_{i=1}^{N_v}\sum_{j=1}^{N_x}\sum_{k=1}^{N_t} \|\Psi_{\text{NODE-ONet}}(v_i)(t_k, x_j) - \Psi^\dagger(v_i)(t_k, x_j)\|_2^2. \end{cases}$$

The generic architure of NODE-ONet:



Contents

1D diffusion-reaction - Setup

We consider the following diffusion-reaction equation

$$\begin{cases} \partial_t u(t,x) - \nabla \cdot (D(t,x)\nabla u(t,x)) + R(t,x)u^2(t,x) = f(t,x) & \forall (t,x) \in [0,T] \times \Omega, \\ u(0,x) = u_0(x) & \forall x \in \Omega, \\ u(t,x) = u_b(t,x) & \forall (t,x) \in [0,T] \times \partial\Omega, \end{cases}$$

In the following experiments, we validate the efficiency of NODE-ONets to approximate the following operators:

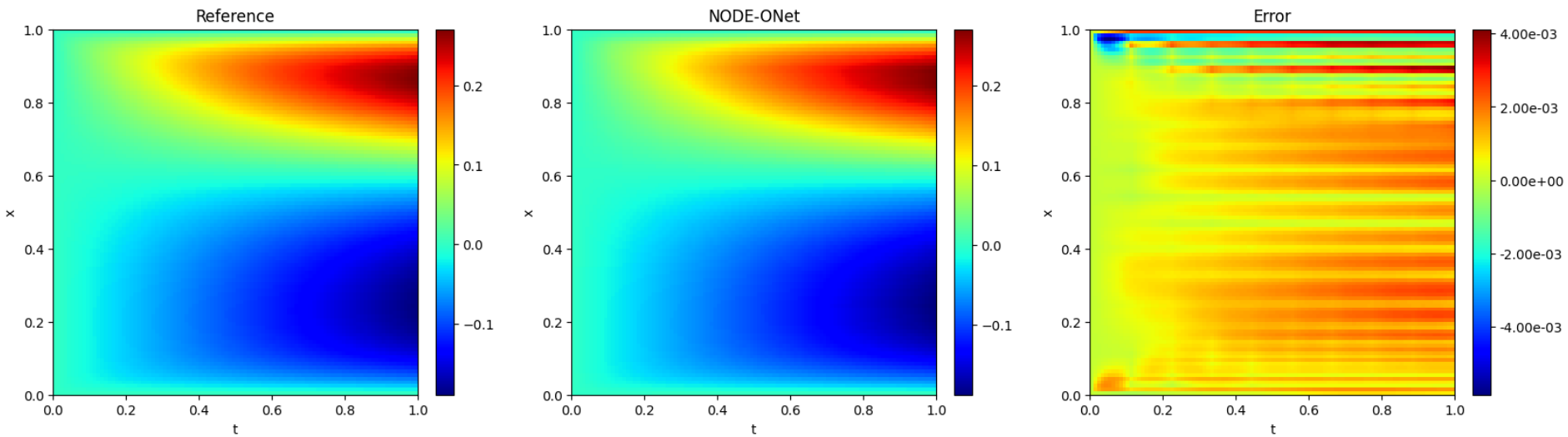
- Source-to-solution operator: $f \mapsto u$. (Comparison with DeepONet)
- Solution operator with multi-inputs: $\{D, f \mapsto u\}$. (Comparison with MIONet)
- Prediction beyond the training time. (Comparison with DeepONet and MIONet)

1D diffusion-reaction - Source-to-solution operator

Physics-encoded NODE:

$$\begin{cases} \dot{\boldsymbol{\psi}}(t) = \sum_{i=1}^{P} W_i \odot \sigma(A_i \odot \boldsymbol{\psi} + \boldsymbol{a}_i^1 t + B_i) + \mathcal{P}_f \boldsymbol{f}, \\ \boldsymbol{\psi}(0) = \mathbf{0} \in \mathbb{R}^{d_U}. \end{cases}$$

Figure: Test results of the deep NODE operator network for the learned source-to-solution operator $\Psi_f^* : f(x) \to u(x,t)$ with one random $f(x)$.



1D diffusion-reaction - Source-to-solution operator - Comparison

Table: Comparisons of the NODE-ONet with DeepONet for learning the source-to-solution operator $\Psi_f^\dagger : f(x) \mapsto u(t,x)$. Testing resolution: $N_x = 100$, $N_t = 100$.

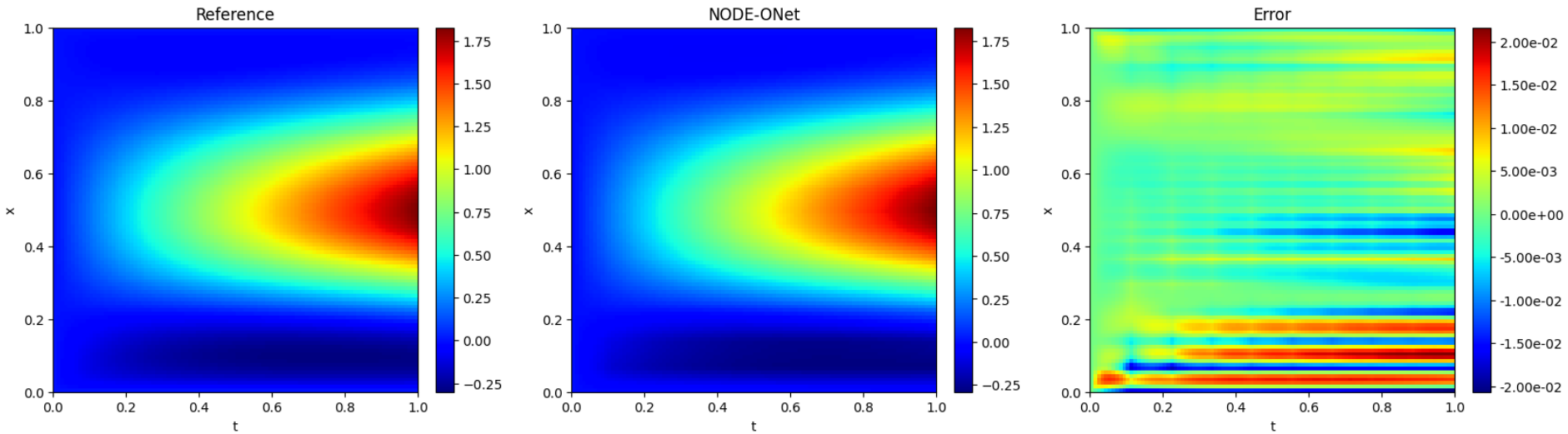
	Training resolutions	#Trainable parameters	#Training input f	Absolute error	Relative error
NODE-ONet	$N_x = 10$ $N_t = 5$ $d_{\mathcal{V}} = 20$	27,550	100	4.248×10^{-3}	7.370×10^{-3}
NODE-ONet	$N_x = 100$ $N_t = 10$ $d_{\mathcal{V}} = 20$	27,550	500	1.368×10^{-3}	2.675×10^{-3}
DeepONet	$N_x = 100$ $N_t = 10$ $K = 50$ $d_{\mathcal{V}} = 100$	40,600	100	6.352×10^{-3}	1.230×10^{-2}
DeepONet	$N_x = 100$ $N_t = 100$ $K = 1{,}000$ $d_{\mathcal{V}} = 100$	40,600	500	1.313×10^{-3}	2.582×10^{-3}

1D diffusion-reaction - Multi-inputs operator

Physics-encoded NODE:

$$\begin{cases} \dot{\boldsymbol{\psi}}(t) = \sum_{i=1}^{P} W_i \odot \sigma(A_i \odot [\mathcal{P}_D \boldsymbol{D}] \odot \boldsymbol{\psi} + \boldsymbol{a}_i^1 t + B_i) + \mathcal{P}_f \boldsymbol{f}, \\ \boldsymbol{\psi}(0) = \mathbf{0} \in \mathbb{R}^{d_U}. \end{cases}$$

Figure: Test results of the deep NODE operator network for the learned solution operator Ψ_m^* with one random multi-input function: $\{D(x), f(x)\} \to u(x,t)$.



1D diffusion-reaction - Multi-inputs operator - Comparison

Table: Comparisons of the NODE-ONet with the MIONet for learning the solution operator with multi-input functions: $\Psi_m^\dagger : \{D(x), f(x)\} \mapsto u(t,x)$. Testing resolution: $N_x = 100$, $N_t = 100$.

	Training resolutions	#Trainable parameters	#Training $\{D, f\}$	Absolute error	Relative error
NODE-ONet	$N_x = 50$ $N_t = 10$	28,550	100	2.362×10^{-2}	5.297×10^{-2}
NODE-ONet	$N_x = 100$ $N_t = 10$	28,550	1,000	4.626×10^{-3}	1.032×10^{-2}
MIONet	$N_x = 100$ $N_t = 100$	161,600	100	1.212×10^{-1}	2.661×10^{-1}
MIONet	$N_x = 100$ $N_t = 100$	161,600	1,000	9.491×10^{-3}	2.072×10^{-2}

1D diffusion-reaction - Prediction

Table: Prediction results of for $t \in [0,2]$. $\Psi_f^*: f(x) \mapsto u(x,t)$: the learned source-to-solution operator ,$\Psi_m^*: \{D(x), f(x)\} \mapsto u(x,t)$: the learned solution operator with multi-input functions. Testing resolution: $N_x = 100$, $N_t = 100$.

		#Training input functions	Training time frame	Test time frame	Absolute error	Relative error
Ψ_f^*	NODE-ONet	500	$t \in [0,1]$	$t \in [0,2]$	6.839×10^{-3}	7.113×10^{-3}
	DeepONet				2.302×10^{-1}	2.360×10^{-1}
Ψ_m^*	NODE-ONet	1,000	$t \in [0,1]$	$t \in [0,2]$	1.392×10^{-2}	1.732×10^{-2}
	MIONet				1.012×10^{-1}	1.251×10^{-1}

1D diffusion-reaction - Prediction - Compare with DeepONet

Figure: Prediction of $\Psi_f^* : f \to u$ beyond $t \in [0, 1]$ by NODE-ONet.

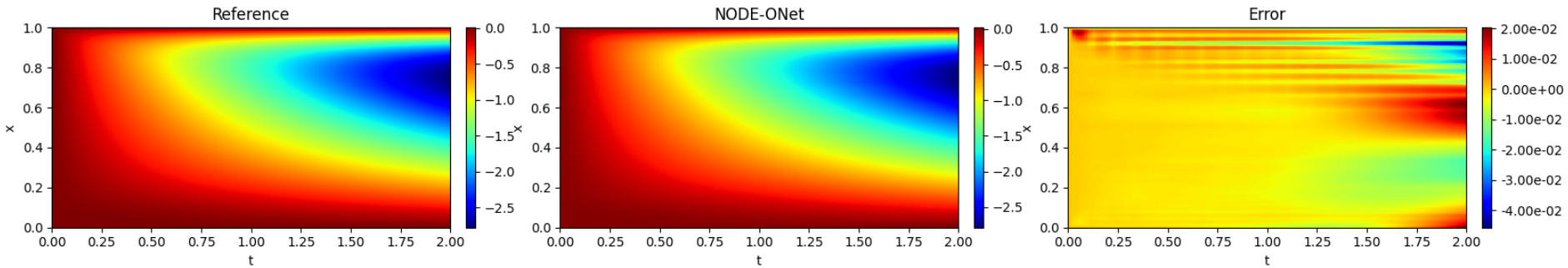
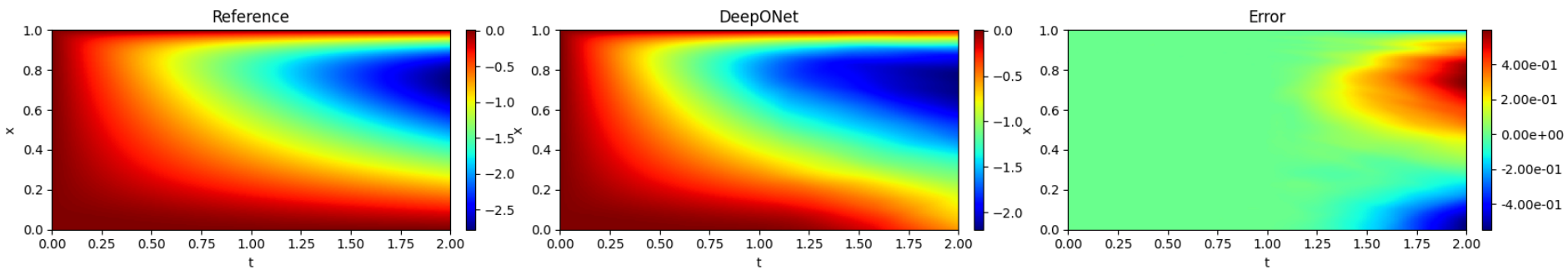


Figure: Prediction of $\Psi_f^* : f \to u$ beyond $t \in [0, 1]$ by DeepONet.



1D diffusion-reaction - Prediction - Compare with MIONet

Figure: Prediction of $\Psi_m^* : \{D, f\} \to u$ beyond $t \in [0, 1]$ by NODE-ONet.

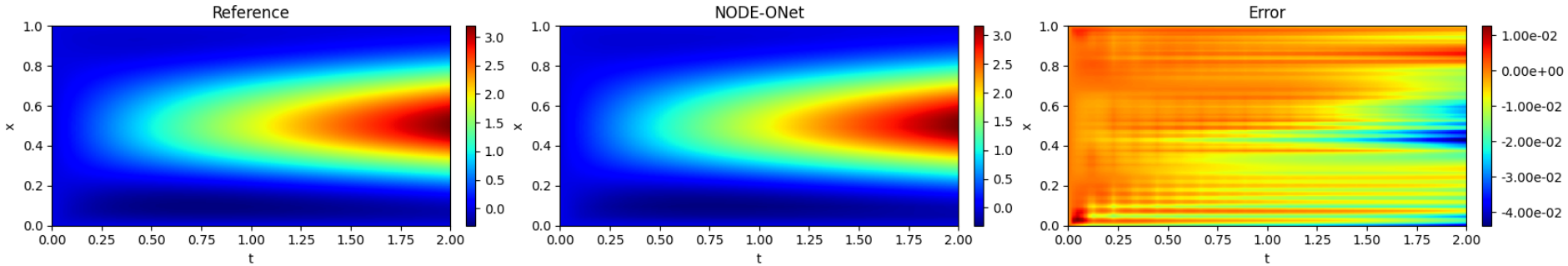
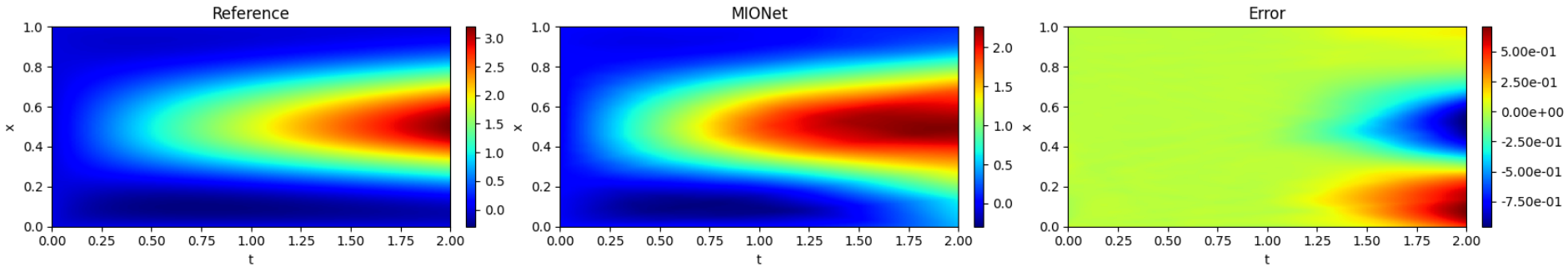


Figure: Prediction of $\Psi_m^* : \{D, f\} \to u$ beyond $t \in [0, 1]$ by MIONet.



2D Navier-Stokes - Setup

We consider the following Navier-Stokes equation

$$\begin{cases} \partial_t u(t,x) + \boldsymbol{V}(t,x) \cdot \nabla u(t,x) = \nu \Delta u(t,x) + f(t,x), & \forall (t,x) \in [0,T] \times \Omega, \\ u(t,x) = \nabla \times \boldsymbol{V}(t,x) := \partial_{x_1} \boldsymbol{V}_2 - \partial_{x_2} \boldsymbol{V}_1, & \forall (t,x) \in [0,T] \times \Omega, \\ \nabla \cdot \boldsymbol{V}(t,x) = 0, & \forall (t,x) \in [0,T] \times \Omega, \\ u(0,x) = u_0(x), & \forall x \in \Omega, \end{cases}$$

with proper boundary conditions. In the following experiments, we validate the efficiency of NODE-ONets to approximate the following operators:

- Initial-to-solution operator: $u_0 \mapsto u$.
- Source-to-solution operator: $f \mapsto u$.
- Multi-inputs operator: $\{u_0, f \mapsto u\}$.

All the NODE-ONets are trained on $t \in [0,10]$ and tested on $t \in [0,20]$.

2D Navier-Stokes - Initial-to-solution operator

2D Navier-Stokes - Source-to-solution operator

2D Navier-Stokes - Multi-inputs operator

Z. Li, K. Liu, Y. Song, H. Yue, E. Zuazua. *Deep Neural ODE Operator Networks for PDEs*. arXiv:2510.15651, 2025. (To appear: Mathematical Models and Methods in Applied Sciences)

Outline

Conclusions

SA-NODEs:

- **Approximation analysis:** ODEs and transport equations.
- **Numerical experiments:** Able to approximate ODEs and transport equations, better than Vanilla NODEs.

NODE-ONet:

- **Physics-Encoded NODEs:** By embedding PDE-specific knowledge, these NODEs achieve superior generalization while maintaining low model complexity.
- **Numerical Efficiency:** The NODE-ONets outperform state-of-the-art methods (e.g., DeepONets, MIONet).
- **Generalization:** Trained encoders/decoders can be transferred to related PDEs without retraining, and predictions remain satisfactory beyond the training time horizon.

Perspectives

SA-NODEs:

- **Predictive properties of SA-NODEs:** Establish theoretical prediction error estimation for SA-NODE over continuous infinite time intervals.
- **Model Predictive Control (MPC):** Design a MPC strategy to approximate a complex dynamical system over a long period of time using multiple SA-NODEs.

NODE-ONet:

- **Further Error Analysis:** The error analysis in this work is mainly devoted to the generic encoder-decoder architecture. It is relevant to analyze the (approximate and generalization) errors for the NODE-ONet framework.
- **Optimal NODEs:** Note that the physics-encoded NODEs used in our experiments are not unique. Hence, it is of great theoretical and practical significance to establish a mathematical principle to determine the optimal physics-encoded NODE for a specific PDE solution operator.


Danke
Merci
Cảm ơn bạn
Teşekkür ederiz
Grazie
谢谢
Obrigado
Thank you
شكريا
謝謝
Gracias
Terima kasih
ధన్యవాదాలు
Спасибо
شكرا
நன்றி
ধন্যবাদ
ありがとう
ขอบคุณ
감사합니다