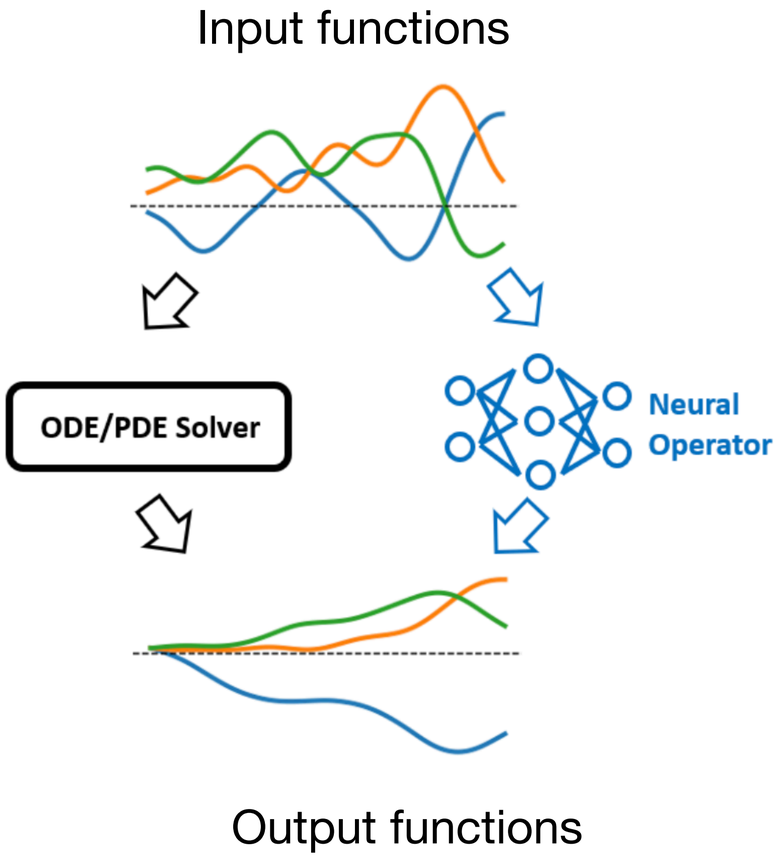
Physics-Informed Latent Neural Operator for Real-time Predictions of Complex Physical Systems

Sharmila Karumuri, Lori Graham-Brady, Somdatta Goswami
Civil and Systems Engineering
Johns Hopkins University

Outline

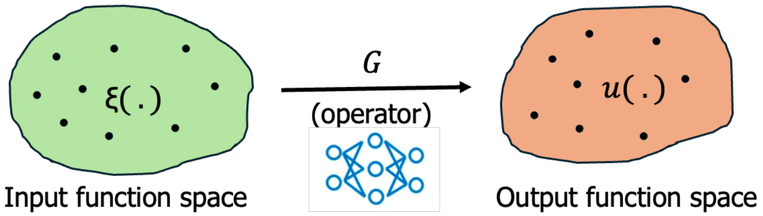
- Introduction
- Neural Operators
- Our approach
- Results
- Future work



Problem

$$\textbf{PDE:} \begin{cases} \mathscr{L}\left(u, \frac{\partial u}{\partial t}, \frac{\partial u}{\partial x}, \frac{\partial^2 u}{\partial x^2}, \ldots, t, x, \gamma(x,t)\right) = 0, & \text{in } \Omega \times (0,T], \\ u(x,0) = g(x), & \text{for } x \in \Omega, \\ \mathscr{B}(u, \frac{\partial u}{\partial x}, t, x, h) = 0, & \text{on } \partial\Omega \times (0,T], \end{cases}$$

- $\mathscr{L}$ represents the PDE operator.
- u is the solution field varying in space and time.
- γ is the random input field could be conductivity field,source field, velocity field and so on.
- Ω is the spatial domain and T is the time duration.
- $g(x)$ is the random initial condition.
- h is the random boundary condition.
- $\mathscr{B}$ denotes the boundary conditions on the boundary $\partial\Omega$.

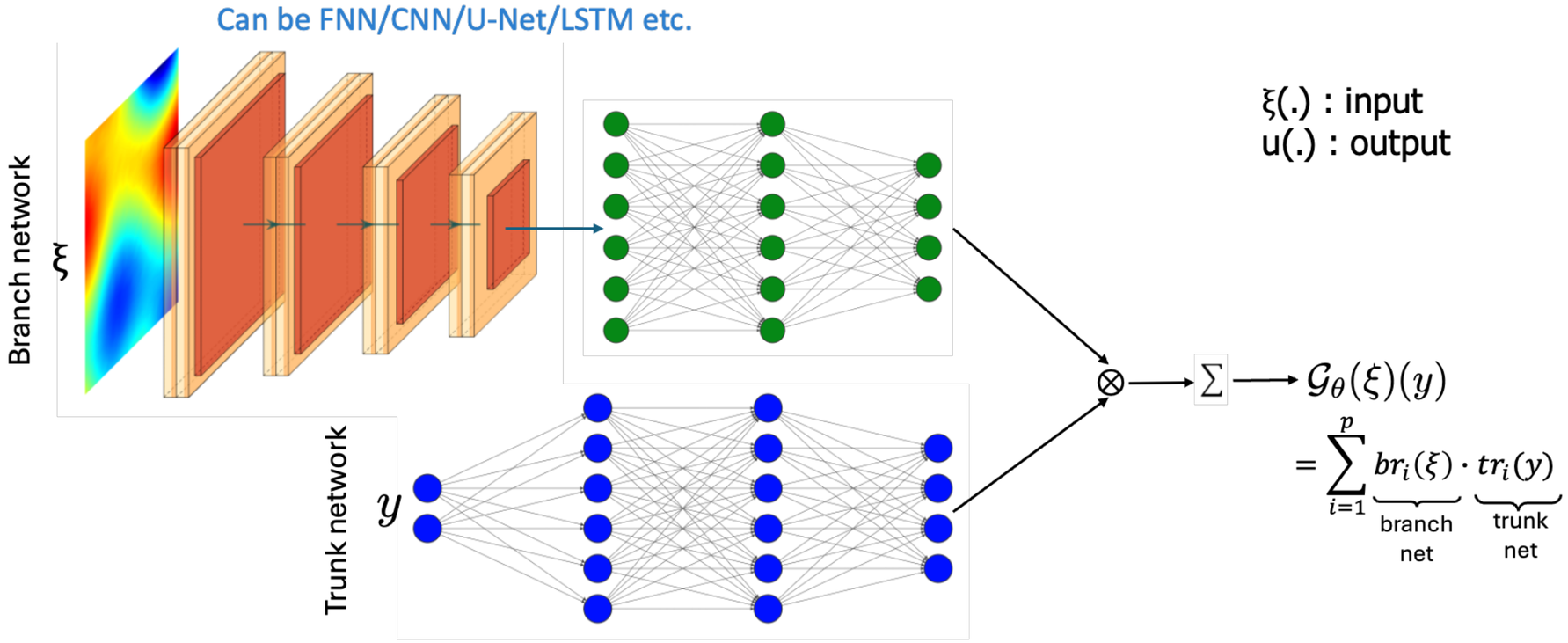


Goal: To learn the mapping between random input fields ξ (γ, g or h) and their corresponding solution fields u.

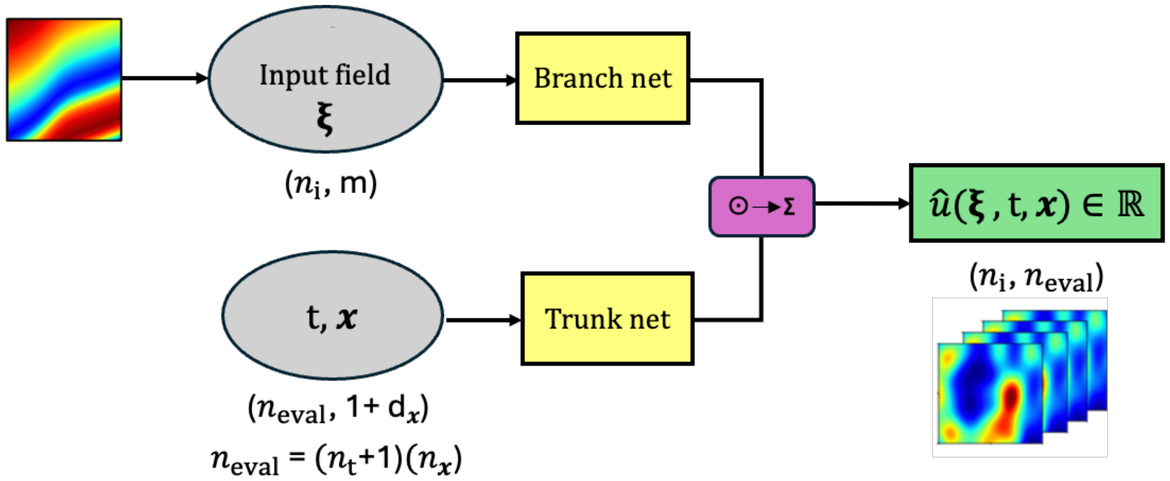
$$\mathscr{G} : \xi \rightarrow u$$

Deep Operator Networks (DeepONet)

- Generalized Universal Approximation Theorem for Operator [Chen ’95, Lu et al. ’19]
- **Branch net**: input ξ, output: $[b_1, b_2, \ldots, b_p]^T \in \mathbb{R}^p$
- **Trunk net**: input y, output: $[t_1, t_2, \ldots, t_p]^T \in \mathbb{R}^p$



Physics Informed Vanilla Neural Operator (PI-Vanilla-NO)



Architecture of the PI-Vanilla-NO

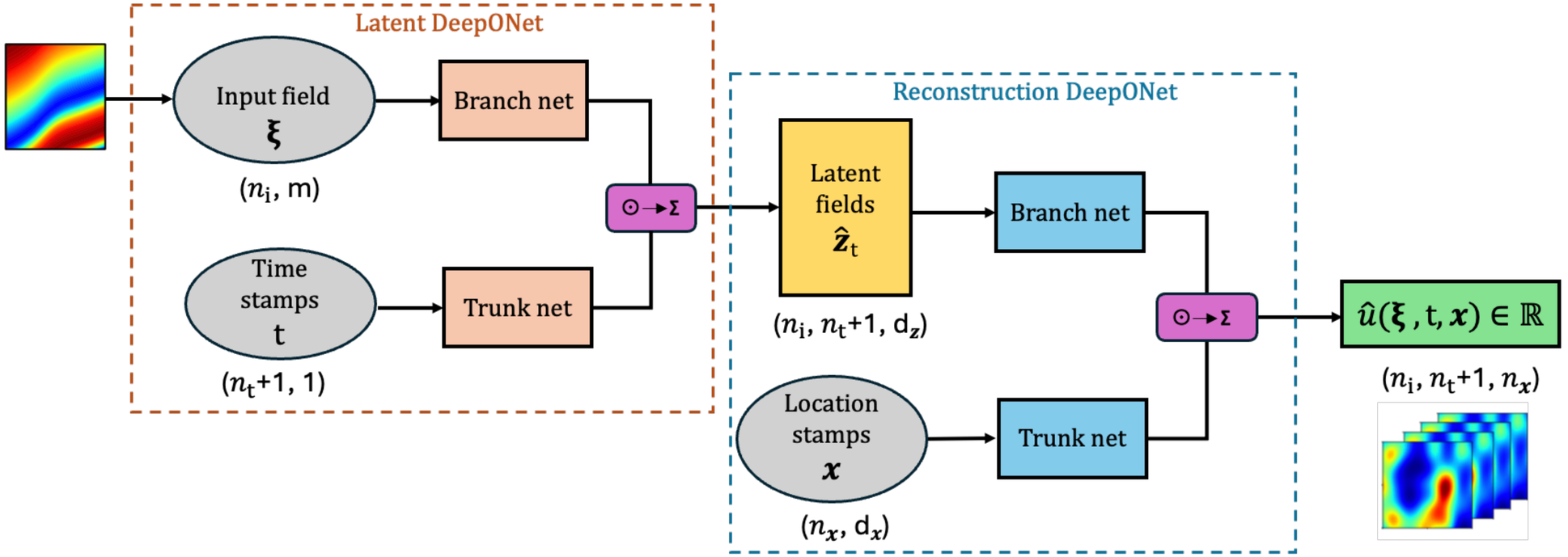
$$\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}_{\text{physics-informed}}(\boldsymbol{\theta}) + \mathcal{L}_{\text{data-driven}}(\boldsymbol{\theta})$$

$$\mathcal{L}_{\text{physics-informed}}(\boldsymbol{\theta}) = \mathcal{L}_r(\boldsymbol{\theta}) + \mathcal{L}_{bc}(\boldsymbol{\theta}) + \mathcal{L}_{ic}(\boldsymbol{\theta})$$

$$\mathcal{L}_{\text{data-driven}}(\boldsymbol{\theta}) = \frac{1}{n_{\text{train}}(n_t+1)n_{\boldsymbol{x}}} \sum_{i=1}^{n_{\text{train}}} \sum_{j=0}^{n_t} \sum_{k=1}^{n_{\boldsymbol{x}}} \left(u(\boldsymbol{\xi}^{(i)}, j\Delta t, \boldsymbol{x}^{(k)}) - \hat{u}(\boldsymbol{\xi}^{(i)}, j\Delta t, \boldsymbol{x}^{(k)}) \right)^2$$

Issue: Solving large-scale problems using standard Vanilla-NO often results in highly over-parameterized networks, long training times, and convergence issues.

Physics Informed Latent Neural Operator (PI-Latent-NO)



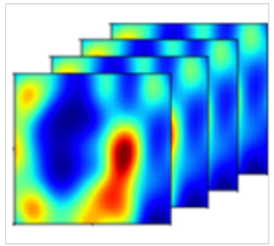
Schematic of Our Approach

$$\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}_{\text{physics-informed}}(\boldsymbol{\theta}) + \mathcal{L}_{\text{data-driven}}(\boldsymbol{\theta})$$

$$\mathcal{L}_{\text{physics-informed}}(\boldsymbol{\theta}) = \mathcal{L}_r(\boldsymbol{\theta}) + \mathcal{L}_{bc}(\boldsymbol{\theta}) + \mathcal{L}_{ic}(\boldsymbol{\theta})$$

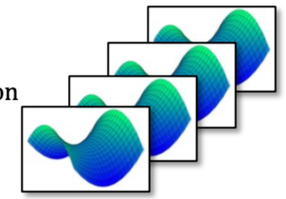
$$\mathcal{L}_{\text{data-driven}}(\boldsymbol{\theta}) = \frac{1}{n_{\text{train}}(n_t+1)n_{\boldsymbol{x}}} \sum_{i=1}^{n_{\text{train}}} \sum_{j=0}^{n_t} \sum_{k=1}^{n_{\boldsymbol{x}}} \left(u(\boldsymbol{\xi}^{(i)}, j\Delta t, \boldsymbol{x}^{(k)}) - \hat{u}(\boldsymbol{\xi}^{(i)}, j\Delta t, \boldsymbol{x}^{(k)})\right)^2$$

$$+ \frac{1}{n_{\text{train}}(n_t+1)n_{\boldsymbol{z}}} \sum_{i=1}^{n_{\text{train}}} \sum_{j=0}^{n_t} \left\| \mathbf{z}(\boldsymbol{\xi}^{(i)}, j\Delta t) - \hat{\mathbf{z}}(\boldsymbol{\xi}^{(i)}, j\Delta t) \right\|_2^2$$



Dimensionality reduction

PCA/Autoencoder



$$\mathcal{T}_{u,i} = \left[\mathbf{u}_0^{(i)}, \mathbf{u}_{1\Delta t}^{(i)}, \mathbf{u}_{2\Delta t}^{(i)}, \ldots, \mathbf{u}_{n_t\Delta t}^{(i)}\right]$$

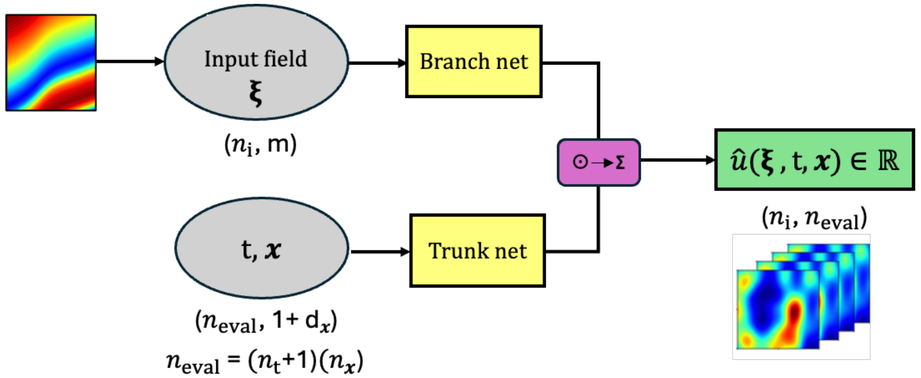
$$\mathcal{T}_{z,i} = \left[\mathbf{z}_0^{(i)}, \mathbf{z}_{1\Delta t}^{(i)}, \mathbf{z}_{2\Delta t}^{(i)}, \ldots, \mathbf{z}_{n_t\Delta t}^{(i)}\right]$$

Key Merits of Latent-NO

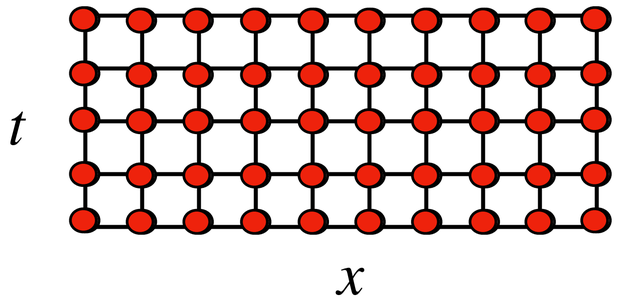
- Efficiently estimates low-dimensional latent spaces, facilitating scalable modeling of complex high-dimensional systems.
- Inherent separability in time and space in our architecture, provides substantial computational advantages.

Comparison

Vanilla - NO

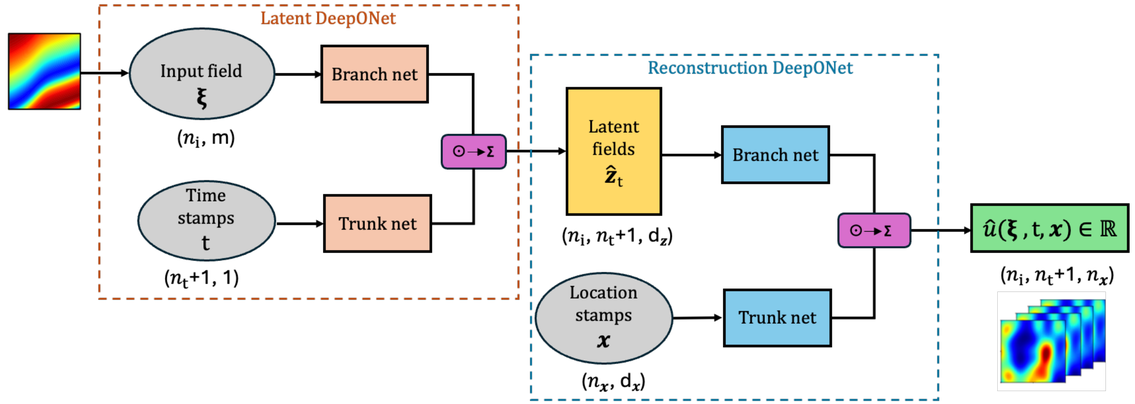


$$u = \sum_{i=1}^{p} Br_i(\xi) \cdot Tr_i(t, x)$$

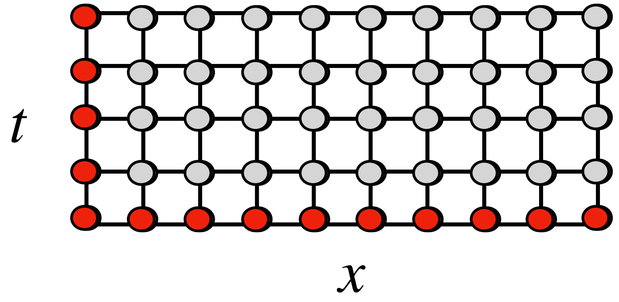


Trunk Network Evaluation

Latent - NO



$$u = \sum_{j=1}^{p} Br_j^r \left(\sum_{i=1}^{q} Br_i^l(\xi) \cdot Tr_i^l(t) \right) Tr_j^r(x)$$



Trunk Network Evaluation

Results

Example 1 - Diffusion Reaction Dynamics

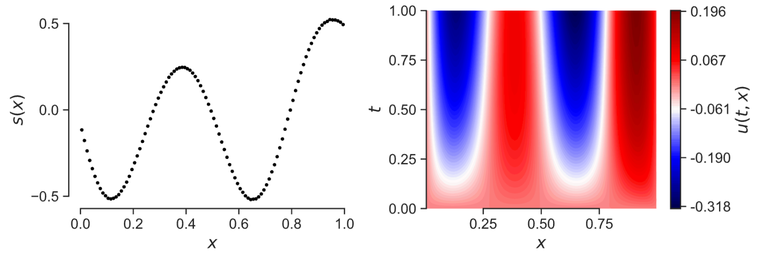
PDE

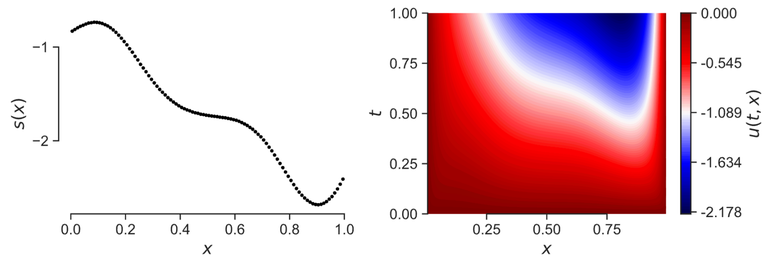
$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} + ku^2 + s(x),$$
$$D = 0.01,\ k = 0.01,$$
$$(t, x) \in (0, 1] \times (0, 1],$$
$$u(0, x) = 0,\ x \in (0, 1)$$
$$u(t, 0) = 0,\ t \in (0, 1)$$
$$u(t, 1) = 0,\ t \in (0, 1)$$
$$\mathcal{G}_{\boldsymbol{\theta}} : s(x) \to u(t, x).$$

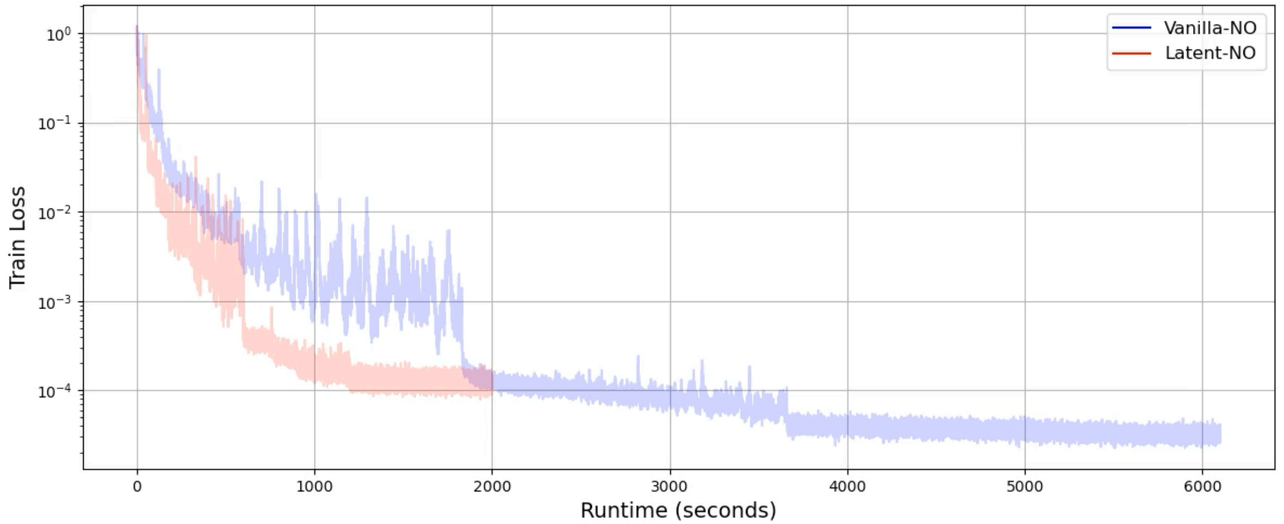
Input Function

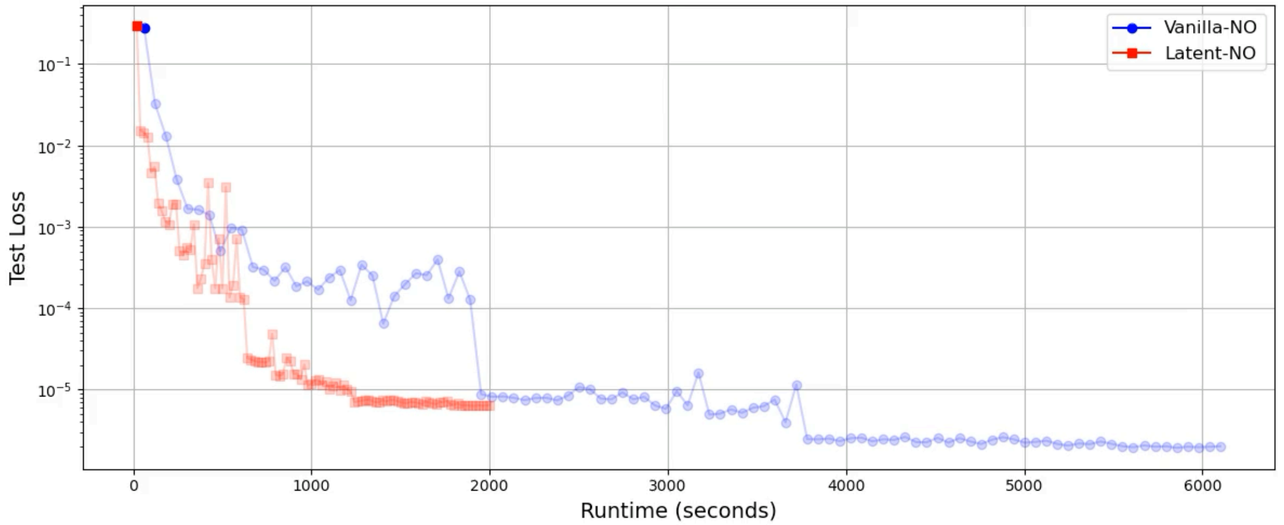
$$s(x) \sim \mathrm{GP}(0, k(x, x')),$$
$$\ell_x = 0.2,\ \sigma^2 = 1.0,$$
$$k(x, x') = \sigma^2 \exp\left\{-\frac{\|x - x'\|^2}{2\ell_x^2}\right\}$$

Samples



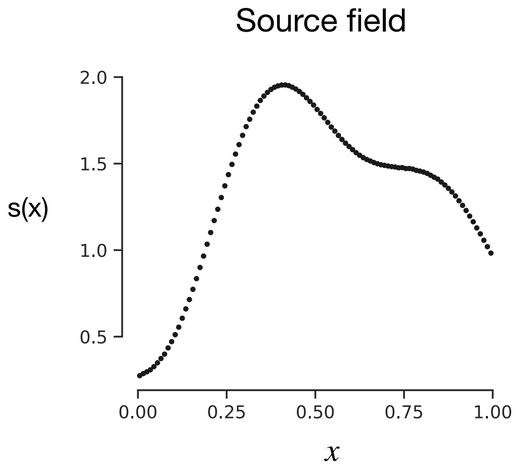


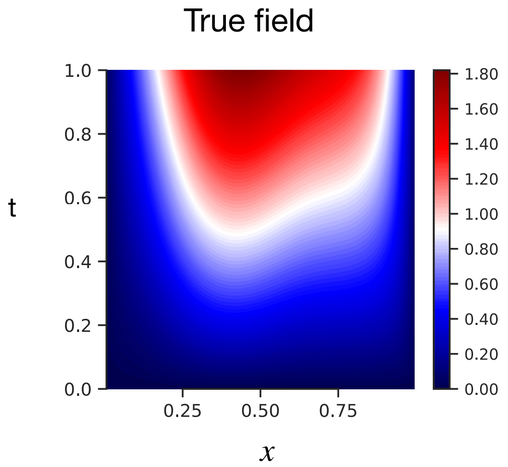


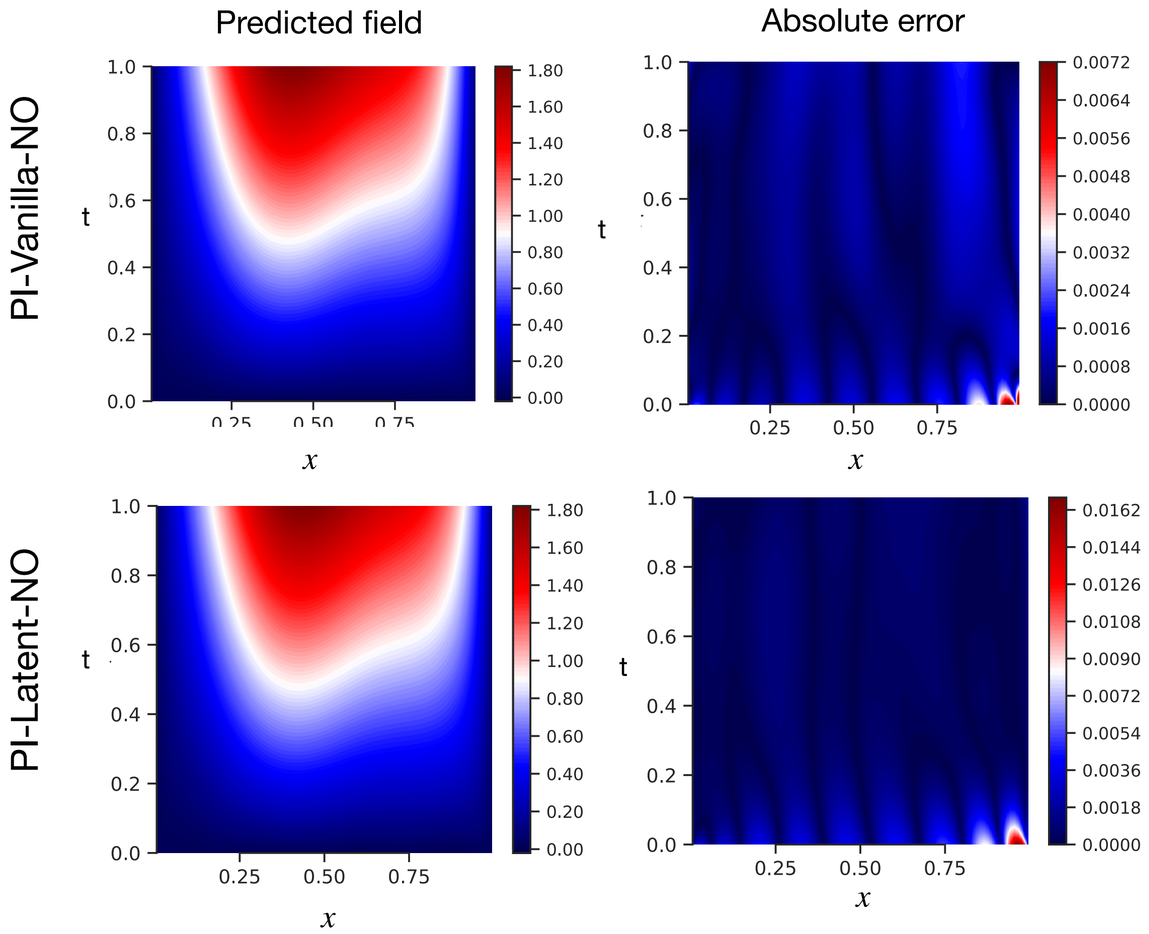


Example 1 (contd..) (Diffusion-reaction dynamics)

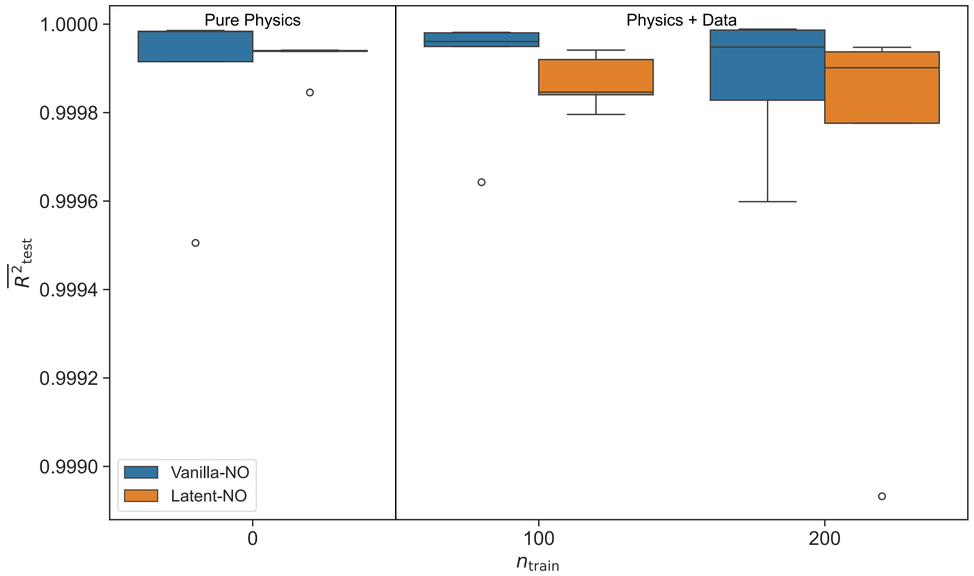
Comparison of all models for a representative test sample:

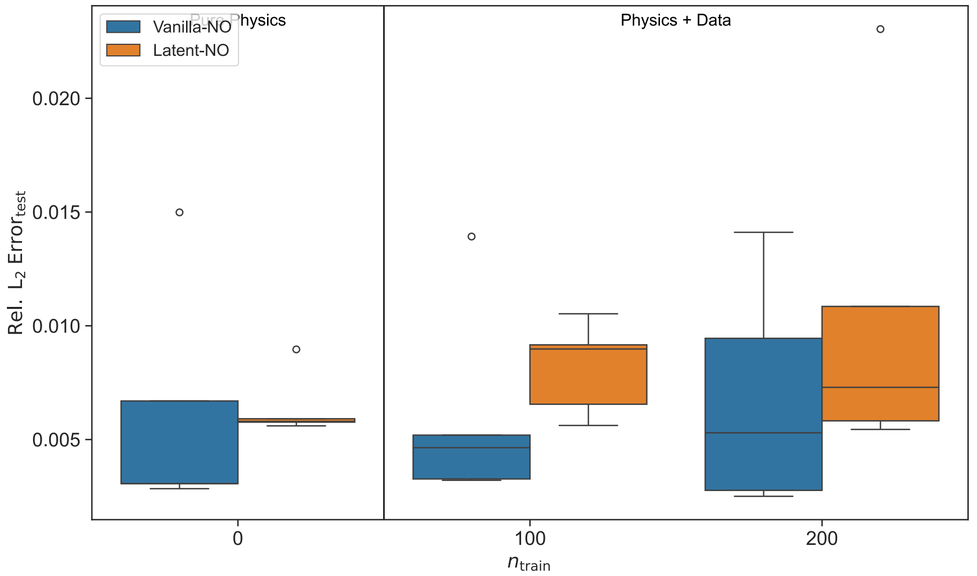


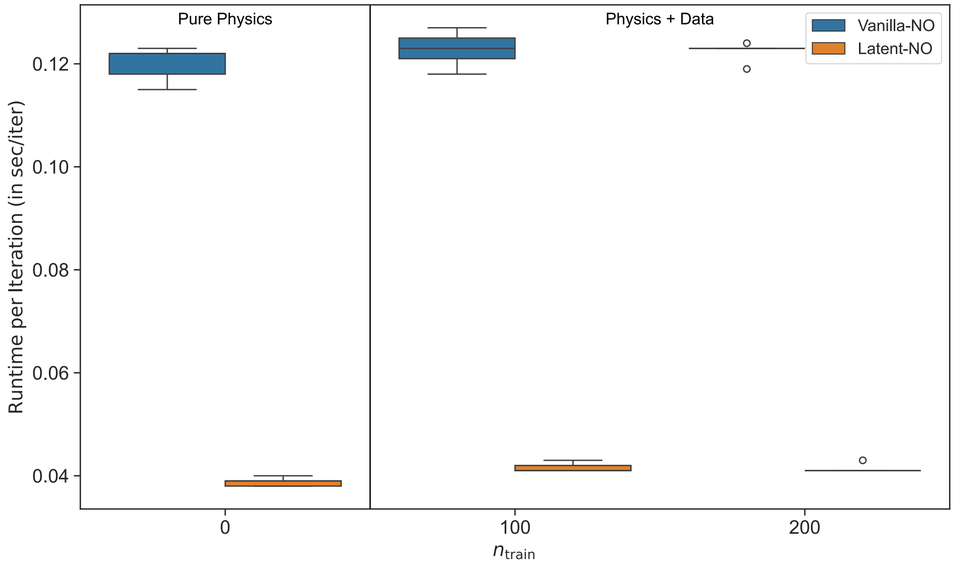




Example 1 (contd..) (Diffusion-reaction dynamics)

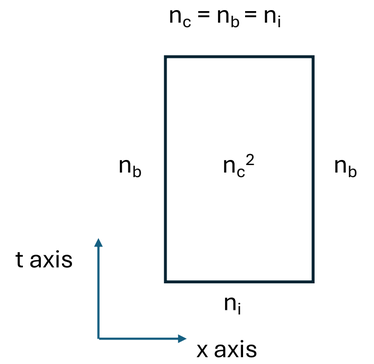


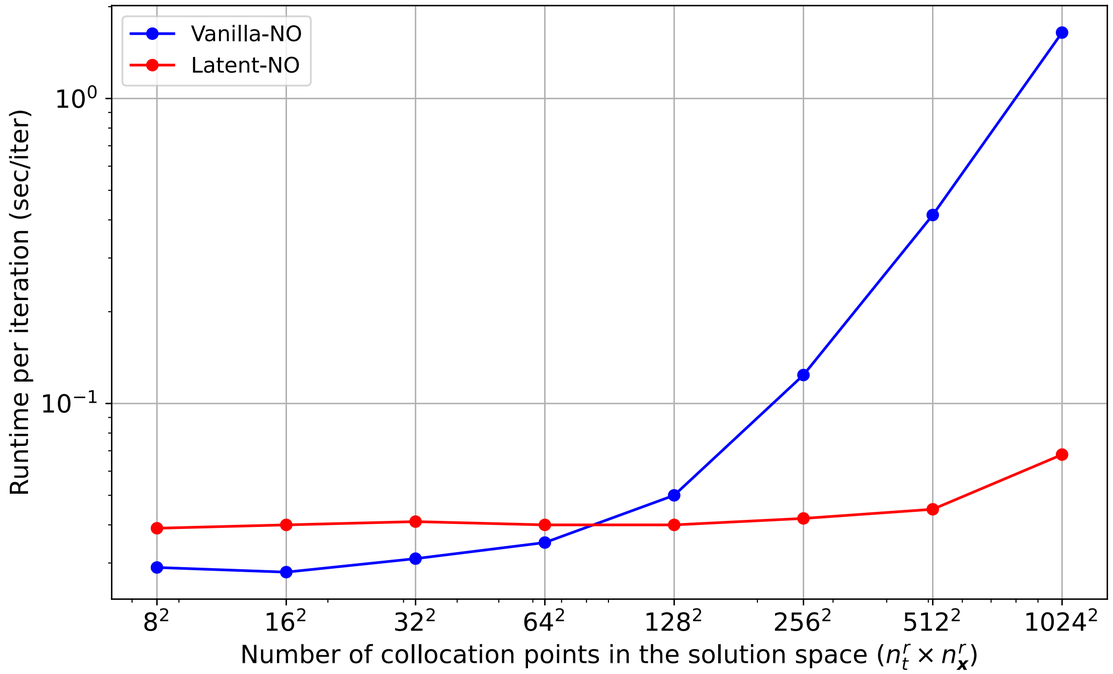


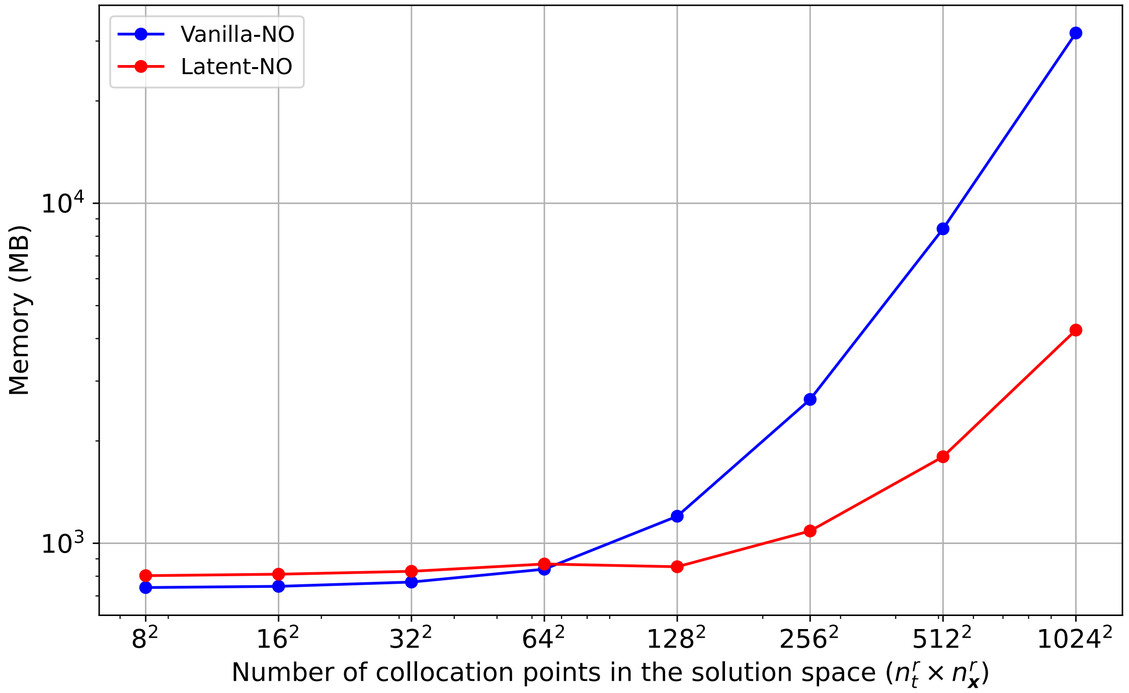


Our method achieves comparable accuracy with significantly reduced training time

Example 1 (contd..) (Diffusion-reaction dynamics)

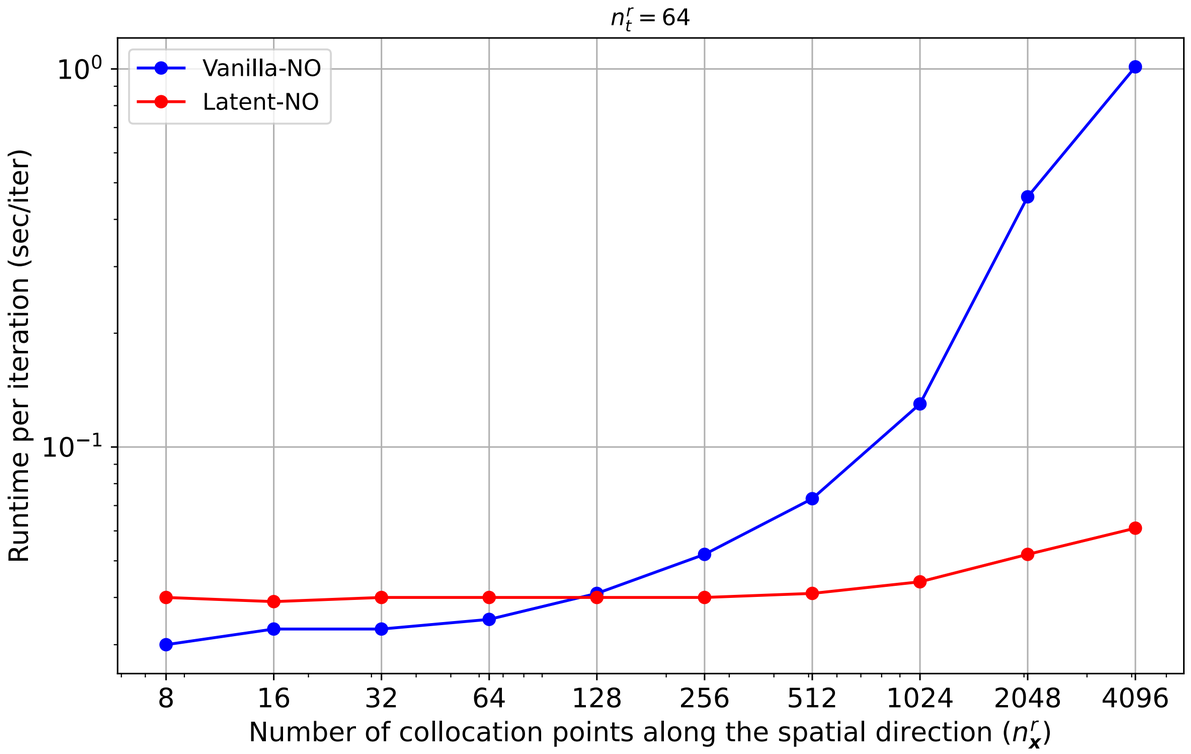


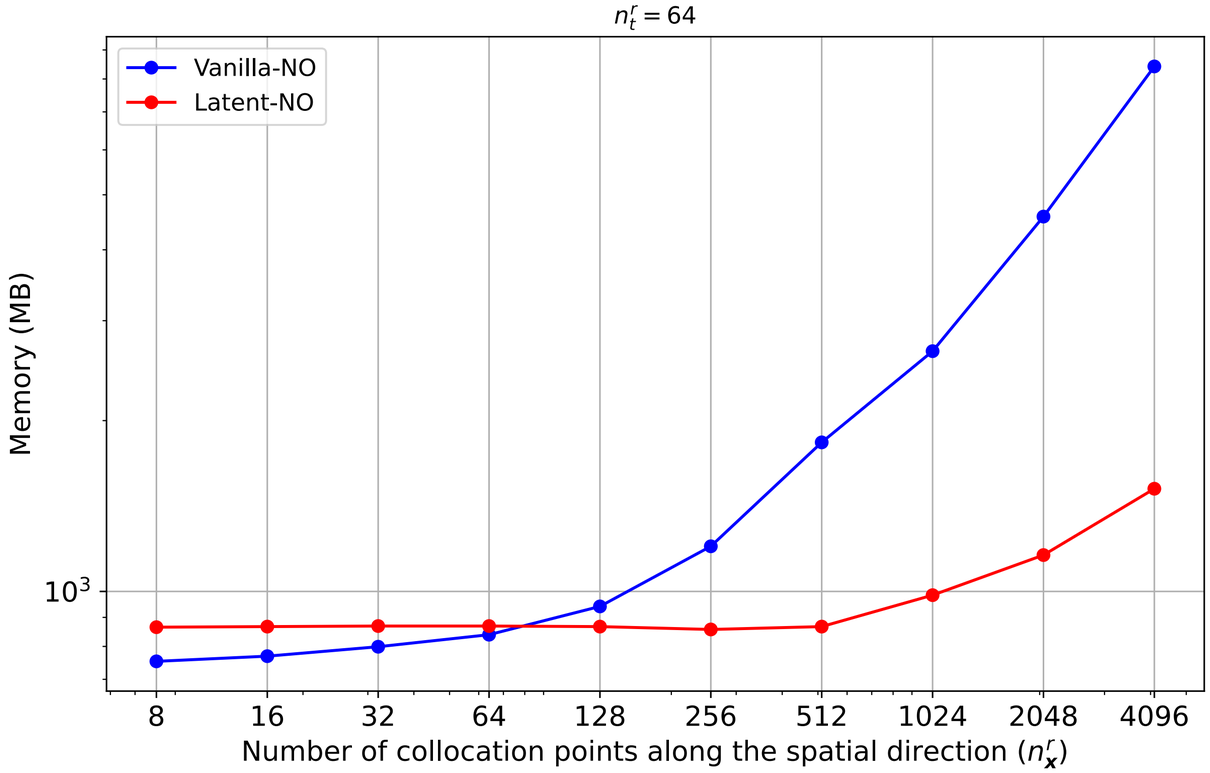




Runtime and memory is almost independent of the discretization of the solution space with our PI-Latent-NO.

Example 1 (contd..) (Diffusion-reaction dynamics)





Example 2 - Stove-Burner Simulation

PDE

$$\frac{\partial u}{\partial t} = D\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + s(x, y, r, a, \text{shape}),$$

$$t \in (0, 1], \quad (x, y) \in \Omega = [-2, 2] \times [-2, 2],$$

$$D = 1,$$

$$\text{ICs:} \quad u(0, x, y) = 0, \quad \text{for all } (x, y) \in \Omega,$$

$$\text{BCs:} \quad u(t, x, y) = 0, \quad \text{for all } (x, y) \in \partial\Omega, \ t \in (0, 1]$$

Input Function

$$s(x, y, r, a, \text{shape})$$

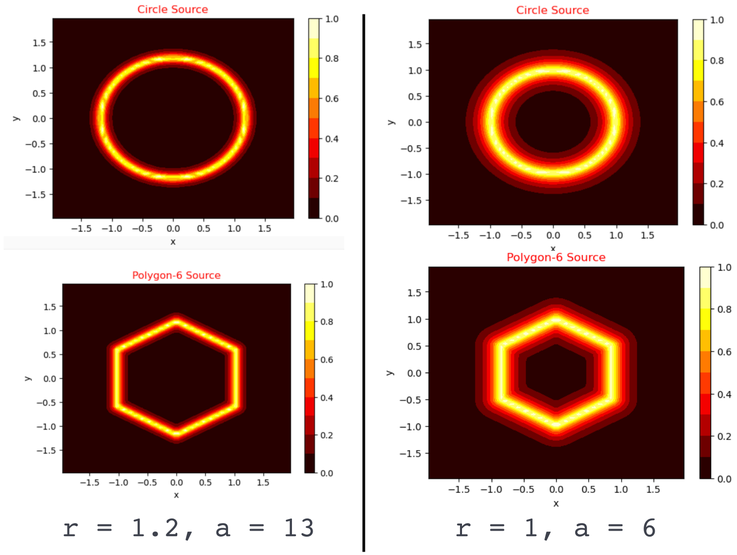
$$r \sim \mathcal{U}(0.75, 1.25) \rightarrow \text{Burner size (radius, side length, etc.)}$$

$$a \sim \mathcal{U}(5, 15) \rightarrow \text{Burner Intensity factor}$$

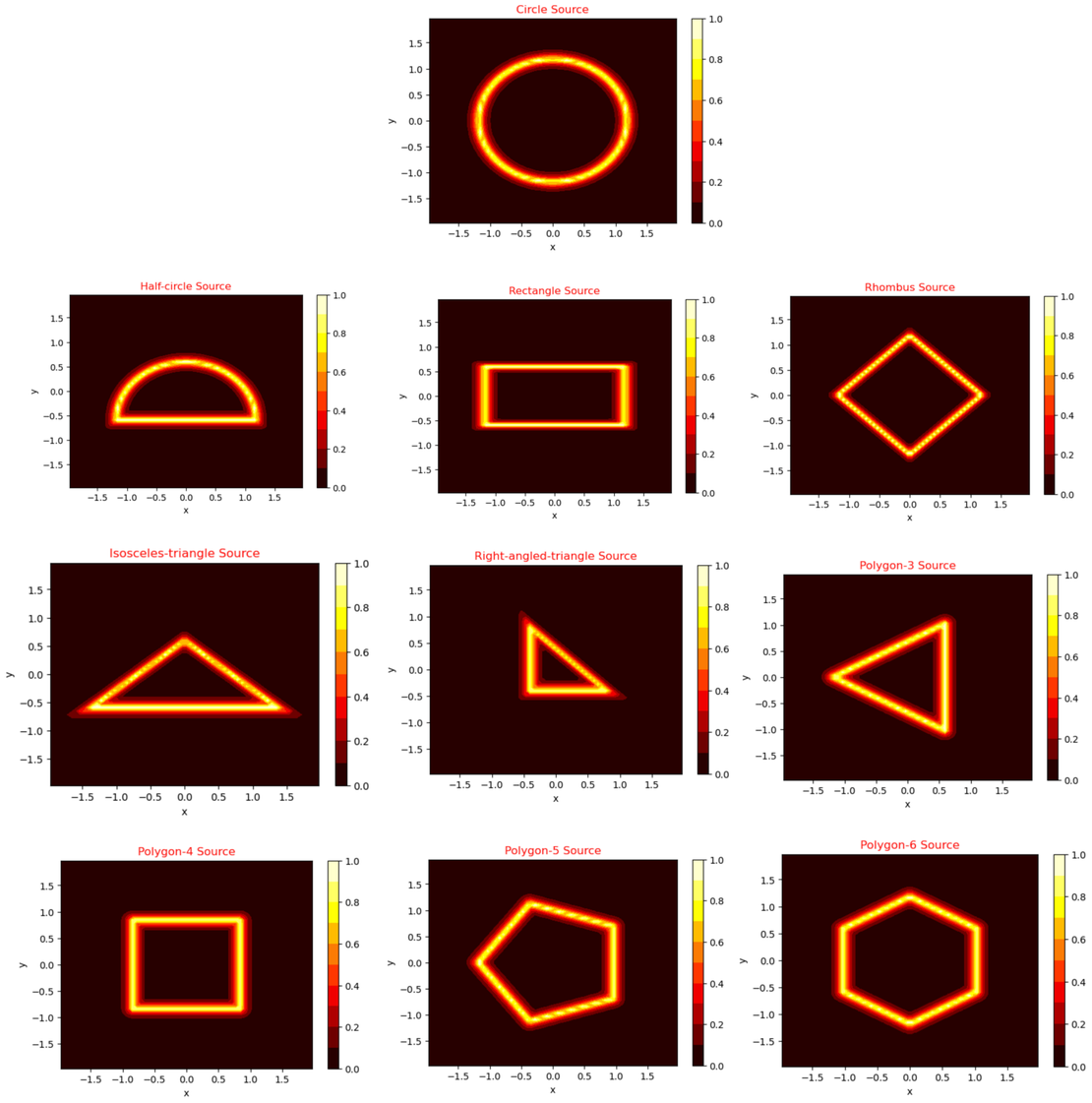
Operator

$$\mathcal{G}_{\boldsymbol{\theta}} : s(x, y, r, a, \text{shape}) \rightarrow u(t, x)$$

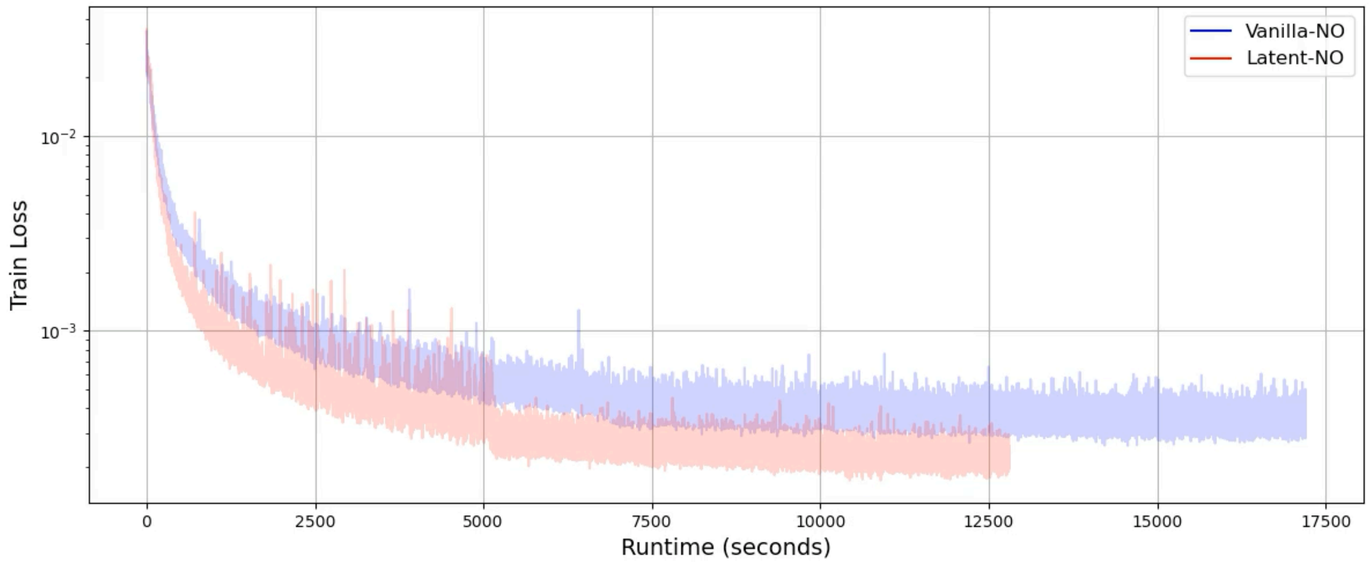
Varying Intensities

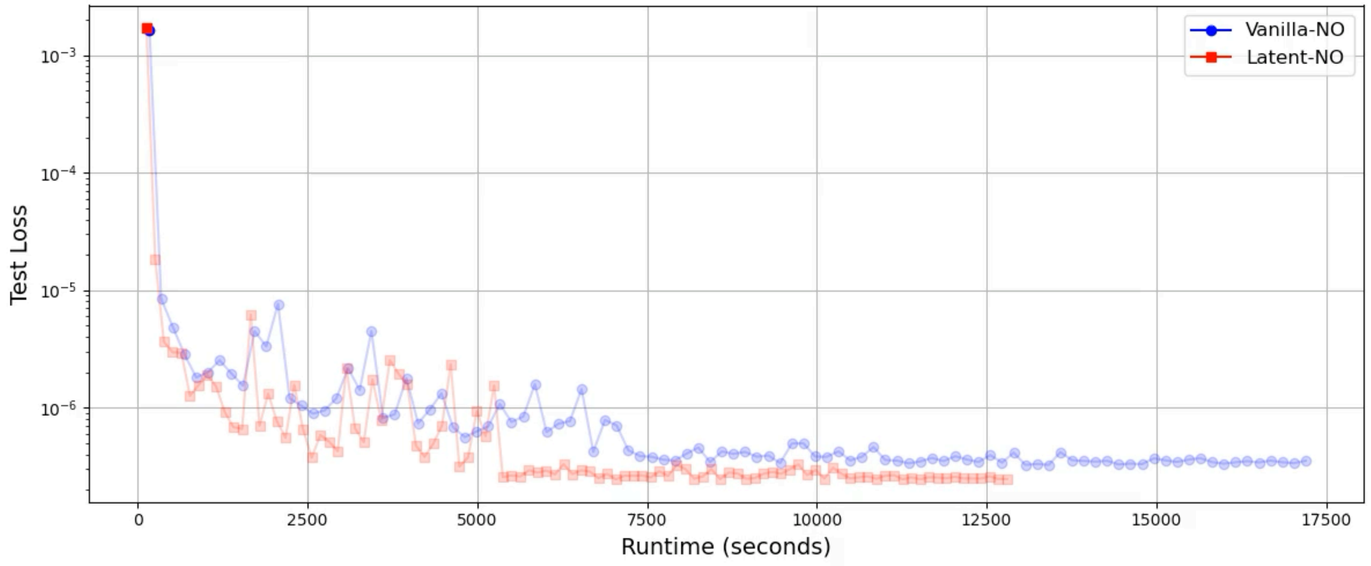


Different Burners



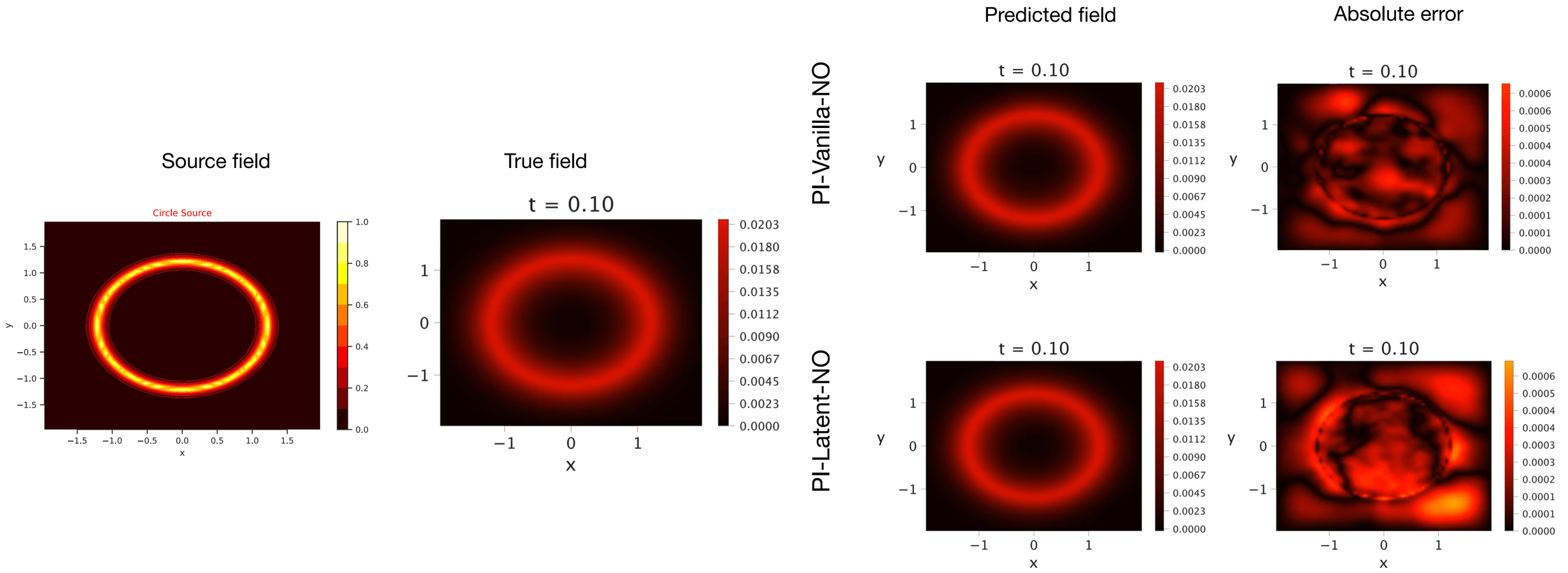
Example 2 (contd..) (Stove-Burner)





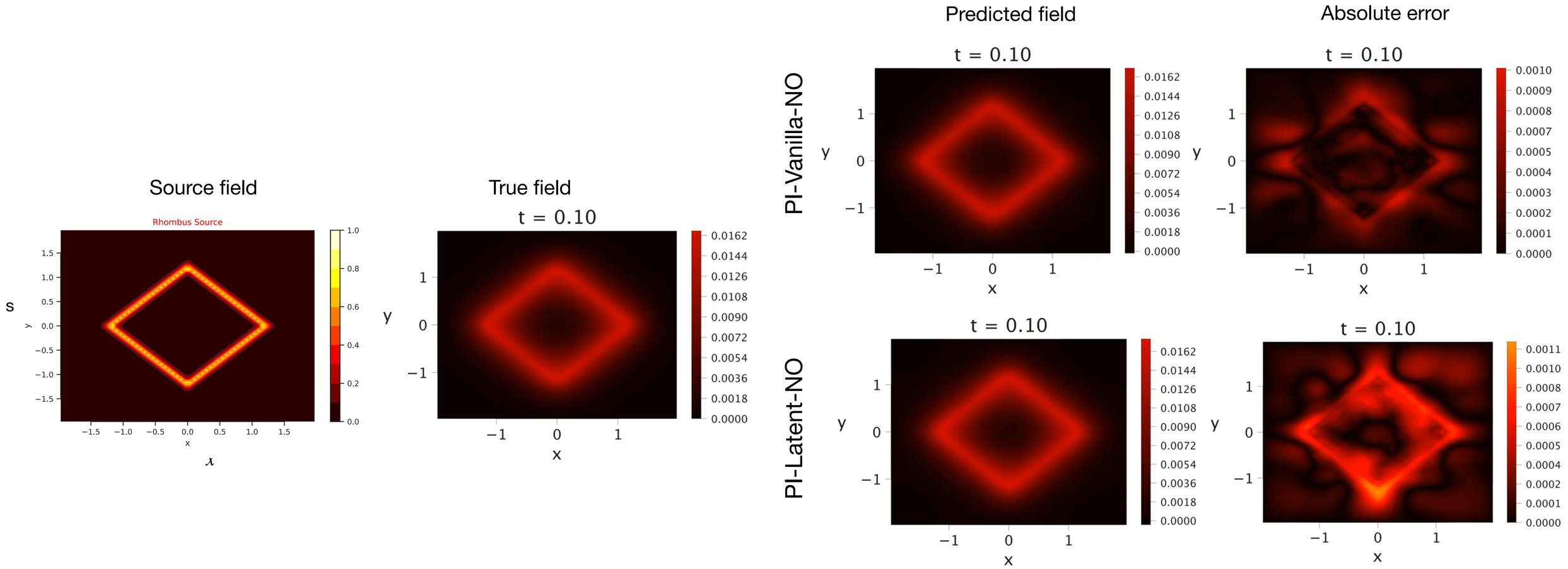
Example 2 (contd..) (Stove-Burner)

Comparison of all models for a representative test sample:

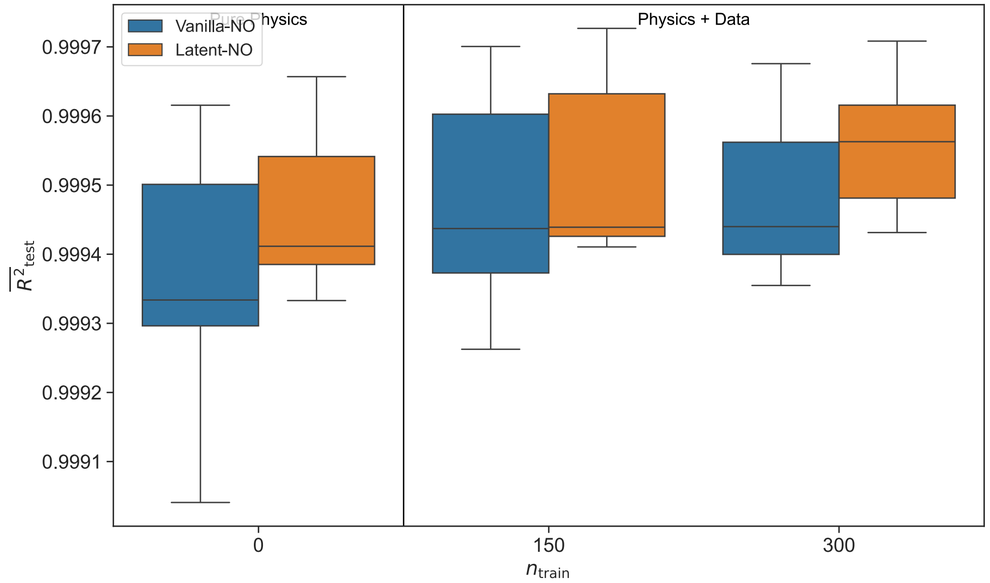


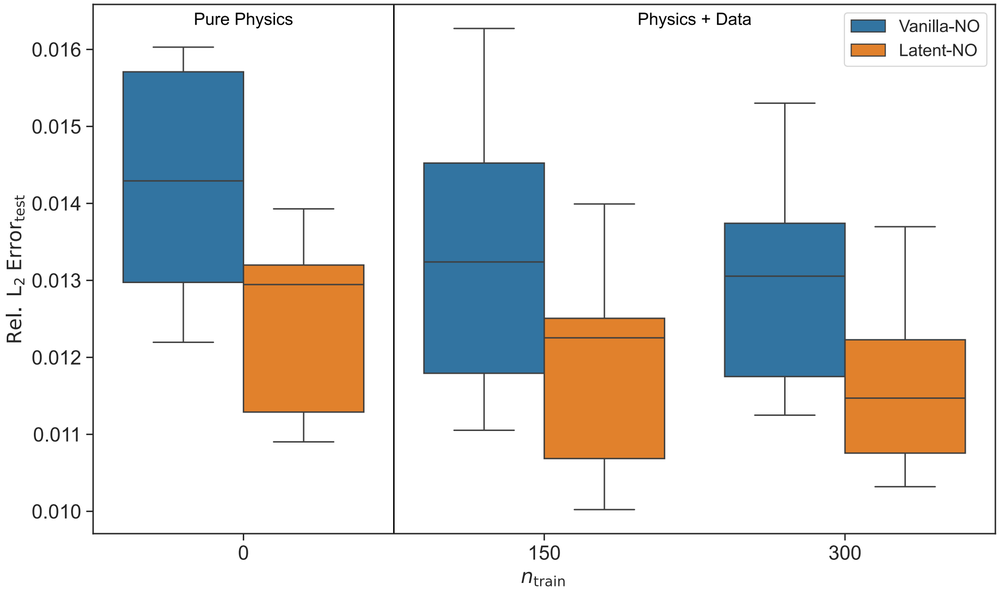
Example 2 (contd..) (Stove-Burner)

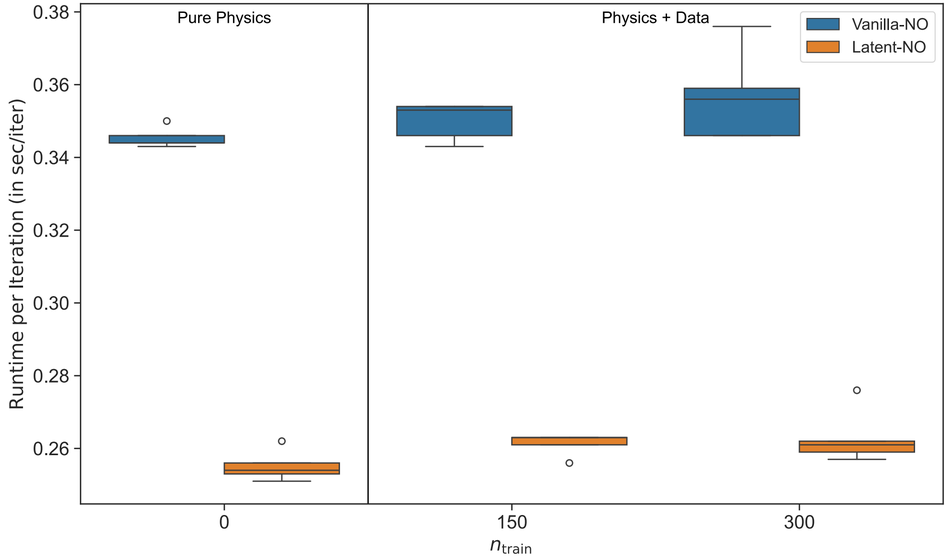
Comparison of all models for a representative test sample:



Example 2 (contd..) (Stove-Burner)

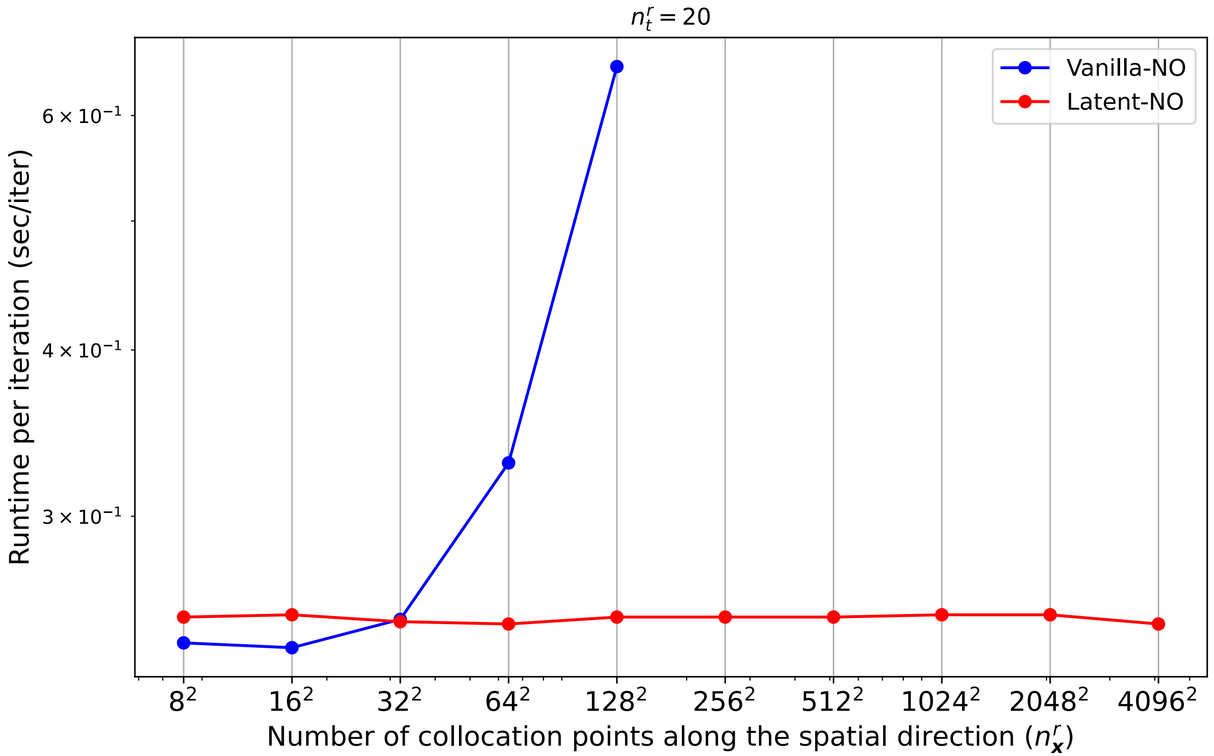


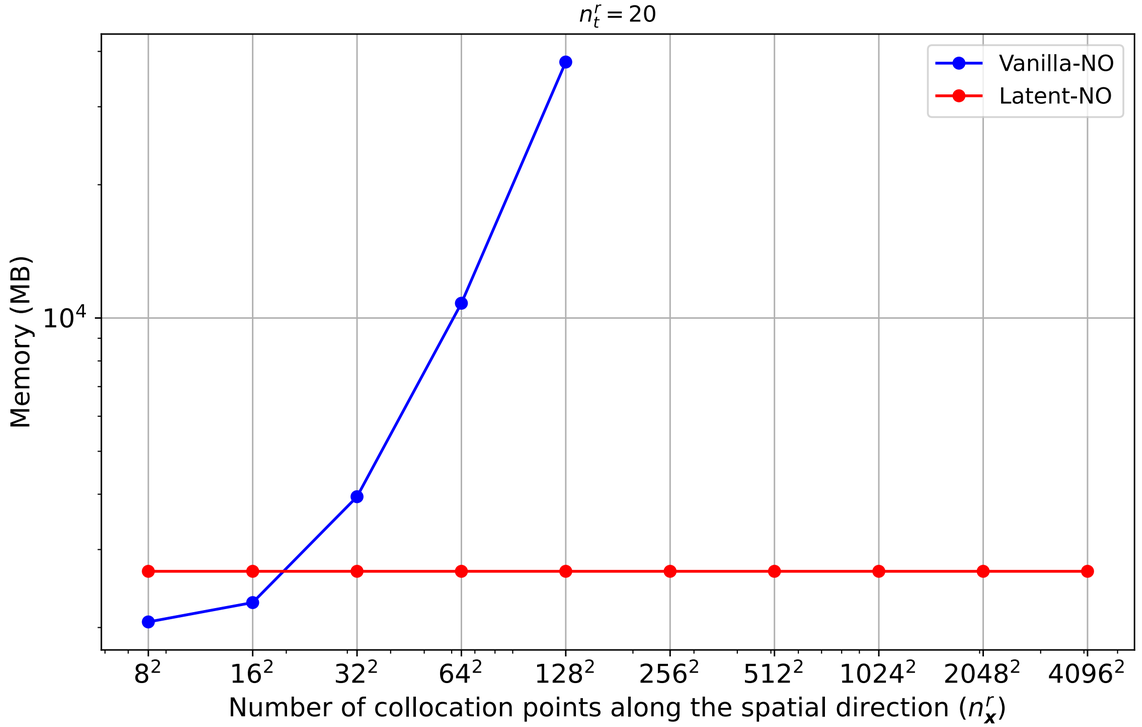




Our method achieves better accuracy with significantly reduced training time

Example 2 (contd..) (Stove-Burner)





Future work

- Extend this method to predict multi-output fields.
- Enforce separability in the trunk network of the reconstruction DeepONet to further reduce run times.
- Incorporate uncertainty estimation by making the neural network weights Bayesian.

Sharmila Karumuri, Lori Graham-Brady, Somdatta Goswami. “Physics-Informed Latent Neural Operator for Real-time Predictions of Complex Physical Systems.” *arXiv preprint arXiv: 2501.08428* (2024)
https://arxiv.org/pdf/2501.08428

Thank you

Table 3: Source Term Equations for Different Geometries

Shape	Equation
Circle	$s(x, y, a, r) = \exp\left(-\left\lvert a\left(\sqrt{x^2 + y^2} - r\right)\right\rvert\right)$
Half-circle	$s(x, y, a, r) = \exp\left(-\left\lvert a \max\left(\sqrt{x^2 + (y + \frac{r}{2})^2} - r, -\left(y + \frac{r}{2}\right)\right)\right\rvert\right)$
Isosceles Triangle	$s(x, y, a, r) = \exp\left(-\left\lvert a \max\left(\max\left(y - \frac{\sqrt{3}}{2}x - \frac{r}{2}, y + \frac{\sqrt{3}}{2}x - \frac{r}{2}\right), -y - \frac{r}{2}\right)\right\rvert\right)$
Right-angled Triangle	$s(x, y, a, r) = \exp\left(-\left\lvert a \max\left(\max\left(-\left(y + \frac{r}{3}\right), -\left(x + \frac{r}{3}\right)\right), x + y - r\right)\right\rvert\right)$
Rectangle	$s(x, y, a, r) = \exp\left(-\left\lvert a\left(\max\left(\frac{\lvert x\rvert}{r_x}, \frac{\lvert y\rvert}{r_y}\right) - 1\right)\right\rvert\right)$
Square	$s(x, y, a, r) = \exp\left(-\left\lvert a\left(\max\left(\lvert x\rvert, \lvert y\rvert\right) - \frac{r}{2}\right)\right\rvert\right)$
Rhombus	$s(x, y, a, r) = \exp\left(-\lvert a\left(\lvert x\rvert + \lvert y\rvert - r\right)\rvert\right)$

$$
\begin{aligned}
\mathcal{L}_{\text{physics-informed}}(\boldsymbol{\theta}) &= \mathcal{L}_r(\boldsymbol{\theta}) + \mathcal{L}_{bc}(\boldsymbol{\theta}) + \mathcal{L}_{ic}(\boldsymbol{\theta}) \\
&= \frac{1}{n_i n_t^r n_{\boldsymbol{x}}^r} \sum_{i=1}^{n_i} \sum_{j=1}^{n_t^r} \sum_{k=1}^{n_{\boldsymbol{x}}^r} \left(\frac{\partial \hat{u}(\boldsymbol{\xi}^{(i)}, t^{(j)}, \boldsymbol{x}^{(k)})}{\partial t} + \mathcal{N}[\hat{u}](\boldsymbol{\xi}^{(i)}, t^{(j)}, \boldsymbol{x}^{(k)}) \right)^2 \\
&+ \frac{1}{n_i n_t^{bc} n_{\boldsymbol{x}}^{bc}} \sum_{i=1}^{n_i} \sum_{j=1}^{n_t^{bc}} \sum_{k=1}^{n_{\boldsymbol{x}}^{bc}} \left(\mathcal{B}[\hat{u}](\boldsymbol{\xi}^{(i)}, t^{(j)}, \boldsymbol{x}^{(k)}) \right)^2 \\
&+ \frac{1}{n_i n_{\boldsymbol{x}}^{ic}} \sum_{i=1}^{n_i} \sum_{k=1}^{n_{\boldsymbol{x}}^{ic}} \left(\hat{u}(\boldsymbol{\xi}^{(i)}, 0, \boldsymbol{x}^{(k)}) - g(\boldsymbol{x}^{(k)}) \right)^2 .
\end{aligned}
$$

Notation

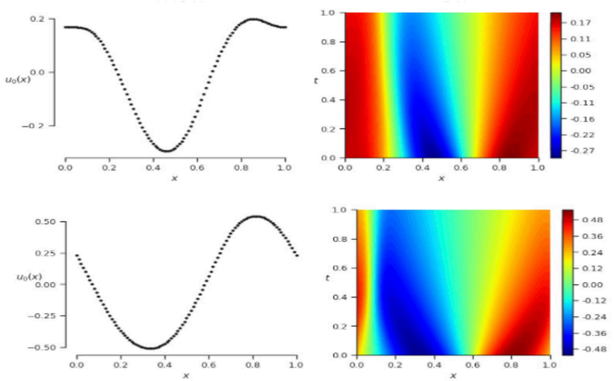
- $\boldsymbol{\xi} \in \mathbb{R}^m$ denotes the random input field.
- $\mathbf{u}_a \in \mathbb{R}^{n_x}$ denotes the solution field of the PDE at time $t = a$.
- n_x represents the dimensionality of the solution field at a given time, determined by the number of grid points in the spatial domain.
- Δt fixed discrete time interval.
- n_t corresponds to the training trajectory length.

$$\left.\begin{array}{r} \text{Input field data: } \boldsymbol{\xi}^{(i)} \\ \text{Output field data: } \left[\mathbf{u}_0^{(i)}, \mathbf{u}_{1\Delta t}^{(i)}, \mathbf{u}_{2\Delta t}^{(i)}, \ldots, \mathbf{u}_{n_t\Delta t}^{(i)}\right] \end{array}\right\} \forall i = 1 \text{ to } n_{\text{train}}$$

- $\mathbf{z}_a \in \mathbb{R}^{d_z}$ is the corresponding latent field at time $t = a$.
- d_z represents the dimensionality of the latent field at a given time.

$$\text{Latent field data from PCA / Autoencoder: } \left[\mathbf{z}_0^{(i)}, \mathbf{z}_{1\Delta t}^{(i)}, \mathbf{z}_{2\Delta t}^{(i)}, \ldots, \mathbf{z}_{n_t\Delta t}^{(i)}\right] \Bigg\} \forall i = 1 \text{ to } n_{\text{train}}$$

Example 3 - Burgers Transport dynamics (1D)

Case	Burgers' transport dynamics
PDE	$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \nu\frac{\partial^2 u}{\partial x^2} = 0,$ $\nu = 0.01,$ $(t, x) \in (0, 1] \times (0, 1],$ $u(0, x) = g(x),\ x \in (0, 1)$ $u(t, 0) = u(t, 1)$ $\frac{\partial u}{\partial x}(t, 0) = \frac{\partial u}{\partial x}(t, 1)$ $\mathcal{G}_{\boldsymbol{\theta}} : g(x) \to u(t, x).$
Input Function	$g(x) \sim \mathcal{N}\left(0, 25^2\left(-\Delta + 5^2 I\right)^{-4}\right),$
Samples	

Model	n_train	Mean Squared Error Test	Rel. L2 Error Test	R2 score Test	Training Time (in sec)	Runtime per Iteration (in sec/iter)
Vanilla-NO	0	1.3e-03 ± 2.3e-04	0.141 ± 0.011	0.974 ± 0.004	6698 ± 27	0.13 ± 0.00
Latent-NO	0	1.6e-03 ± 3.0e-04	0.154 ± 0.014	0.969 ± 0.005	2993 ± 123	0.06 ± 0.00
Vanilla-NO	100	1.2e-03 ± 2.4e-04	0.131 ± 0.013	0.977 ± 0.004	6766 ± 50	0.14 ± 0.00
Latent-NO	100	1.2e-03 ± 1.5e-04	0.131 ± 0.004	0.977 ± 0.002	3079 ± 41	0.06 ± 0.00
Vanilla-NO	200	1.0e-03 ± 9.6e-05	0.125 ± 0.005	0.980 ± 0.002	6734 ± 26	0.13 ± 0.00
Latent-NO	200	1.1e-03 ± 9.5e-05	0.122 ± 0.007	0.980 ± 0.002	3155 ± 156	0.06 ± 0.00

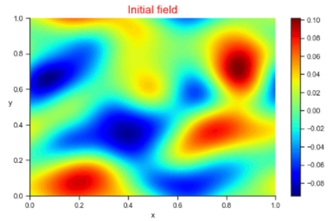
Example 4 - Burgers Transport dynamics (2D)

$$u_t + u\partial_x u + u\partial_y u = 0.01\left(u_{xx} + u_{yy}\right) \forall (x, y) \in [0, 1]^2, t \in [0, 1]$$

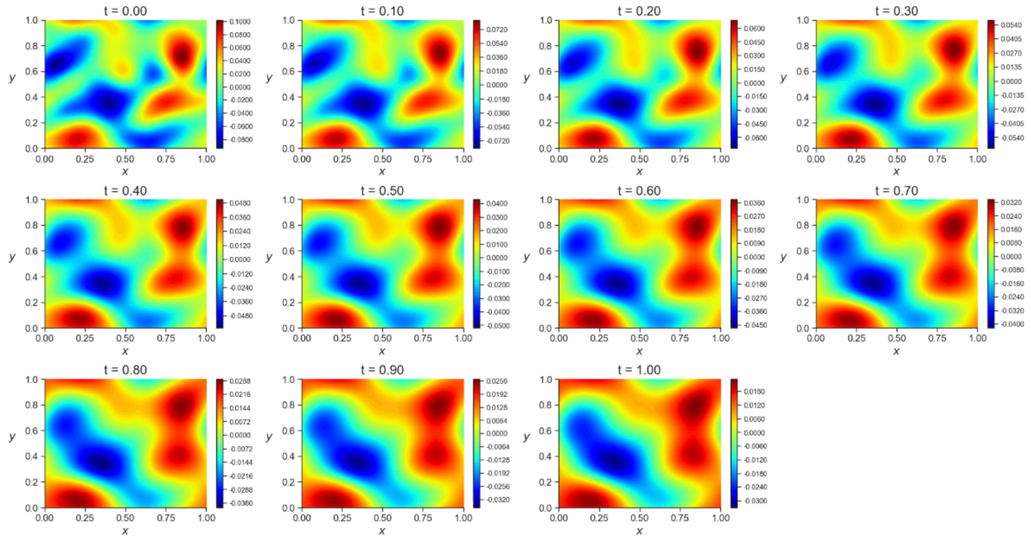
$$u(0, x, y) = u_0(x, y),$$

$$u(t, 0, y) = u(t, L, y)$$
$$\left.\frac{\partial u}{\partial x}\right|_{x=0} = \left.\frac{\partial u}{\partial x}\right|_{x=L}$$
$$u(t, x, 0) = u(t, x, L)$$
$$\left.\frac{\partial u}{\partial y}\right|_{y=0} = \left.\frac{\partial u}{\partial y}\right|_{y=L}$$

$$\mathcal{G}_\theta : u_0(x, y) \to u(t, x, y)$$



True Solution



Model	n_train	Mean Squared Error Test	Rel. L2 Error Test	R2 score Test	Training Time (in sec)	Runtime per Iteration (in sec/iter)
Vanilla-NO	0	1.0e-05 ± 3.6e-07	0.115 ± 0.003	0.987 ± 0.001	60194 ± 404	0.75 ± 0.01
Latent-NO	0	1.0e-05 ± 6.6e-07	0.114 ± 0.004	0.987 ± 0.001	51543 ± 745	0.64 ± 0.01
Vanilla-NO	150	8.0e-06 ± 2.8e-07	0.101 ± 0.003	0.989 ± 0.001	60584 ± 557	0.76 ± 0.01
Latent-NO	150	8.8e-06 ± 1.2e-07	0.107 ± 0.002	0.988 ± 0.000	52125 ± 922	0.65 ± 0.01
Vanilla-NO	300	7.5e-06 ± 3.8e-07	0.099 ± 0.002	0.990 ± 0.000	60219 ± 150	0.75 ± 0.00
Latent-NO	300	8.2e-06 ± 3.6e-07	0.103 ± 0.001	0.989 ± 0.000	52504 ± 750	0.66 ± 0.01