

# TI-DeepONet: Learnable Time Integration for Stable Long-Term Extrapolation

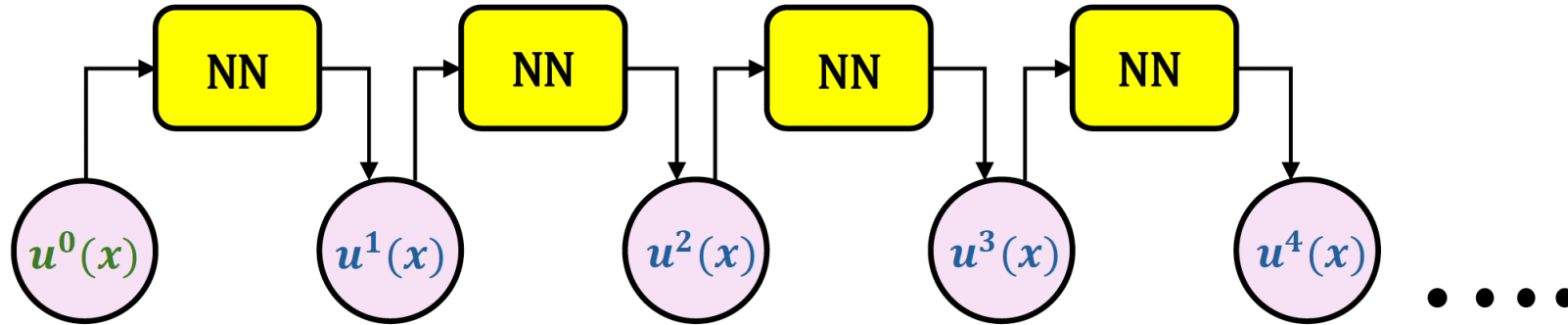
Dibyajyoti Nayak

**Advisor:** Dr. Somdatta Goswami

Department of Civil and Systems Engineering  
Johns Hopkins University

# Introduction

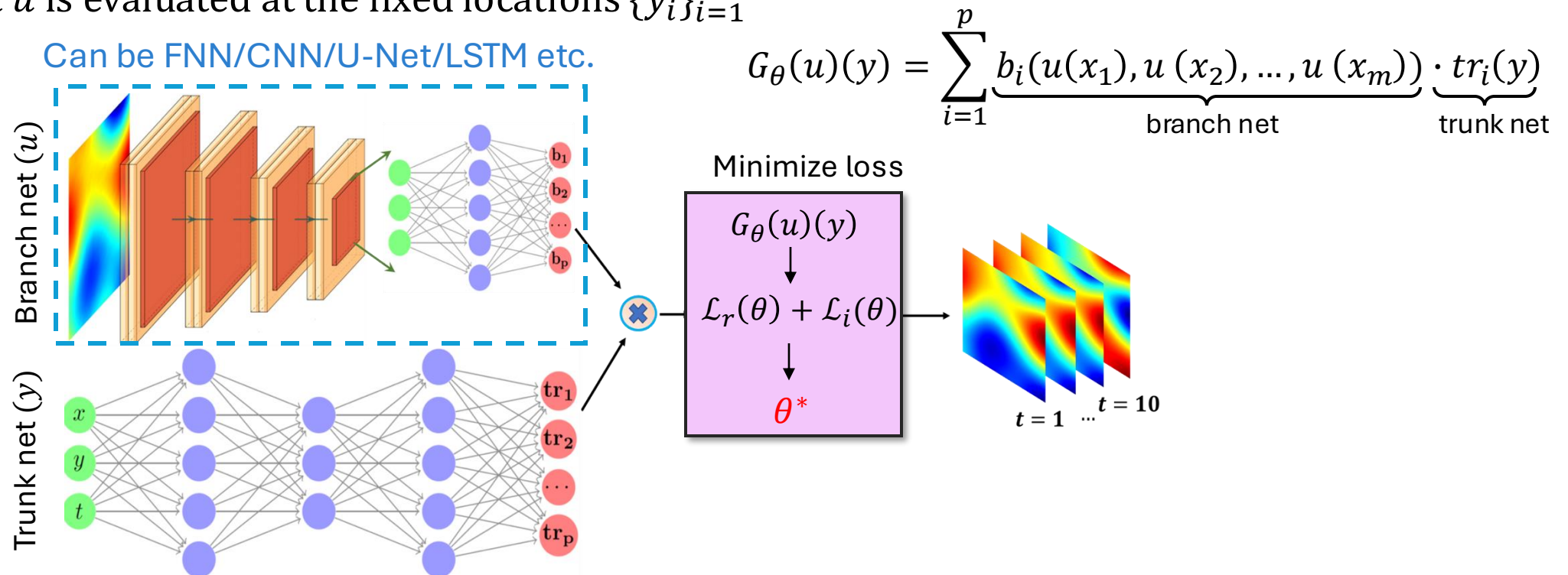
- Operator learning frameworks inherently operate in two learning paradigms:
  - **Full rollout:** Represent entire spatiotemporal output solution in terms of learnt basis functions and coefficients
  - **Autoregressive:** Recursive predictions in time beginning from an initial state



- **Full rollout:** Does not respect the underlying Markovian (causal) structure, i.e., dependency between the solution states in time
- **Autoregressive:** Error accumulation in time
- **Idea:** To formulate a hybrid approach by combining operator learning with classical numerical schemes for accurate prediction of the future states of the system

# Background – Deep Operator Network (DeepONet)

- Generalized Universal Approximation Theorem for Operator [Chen '95, Lu et al. '19]
- Branch net:** Input  $\{u(x_i)\}_{i=1}^m$ , output:  $[b_1, b_2, \dots, b_p]^T \in \mathbb{R}^p$
- Trunk net:** Input  $y$ , output:  $[t_1, t_2, \dots, t_p]^T \in \mathbb{R}^p$
- Input  $u$  is evaluated at the fixed locations  $\{y_i\}_{i=1}^m$



# One-dimensional viscous Burgers' Equation

- Consider the 1D viscous Burgers' PDE:

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2}, \quad \nu = 0.01$$

- Initial conditions:** Gaussian Random Field (GRF) with Matern' Kernel

$$u(x, t = 0) = s(x)$$

- Boundary conditions:** (1)  $u(x = 0, t) = u(x = 1, t)$ , (2)  $\frac{\partial u}{\partial x}(x = 0, t) = \frac{\partial u}{\partial x}(x = 1, t)$

- $N_s = 2500$  samples of ICs,  $N_t = 101$  timesteps,  $N_x = 101$  spatial grid points

- Let  $u^i(x)$  represent the solution at the  $i^{th}$  timestep

- Branch Input:**  $[u^0(x), u^1(x), u^2(x), \dots, u^{49}(x)]$

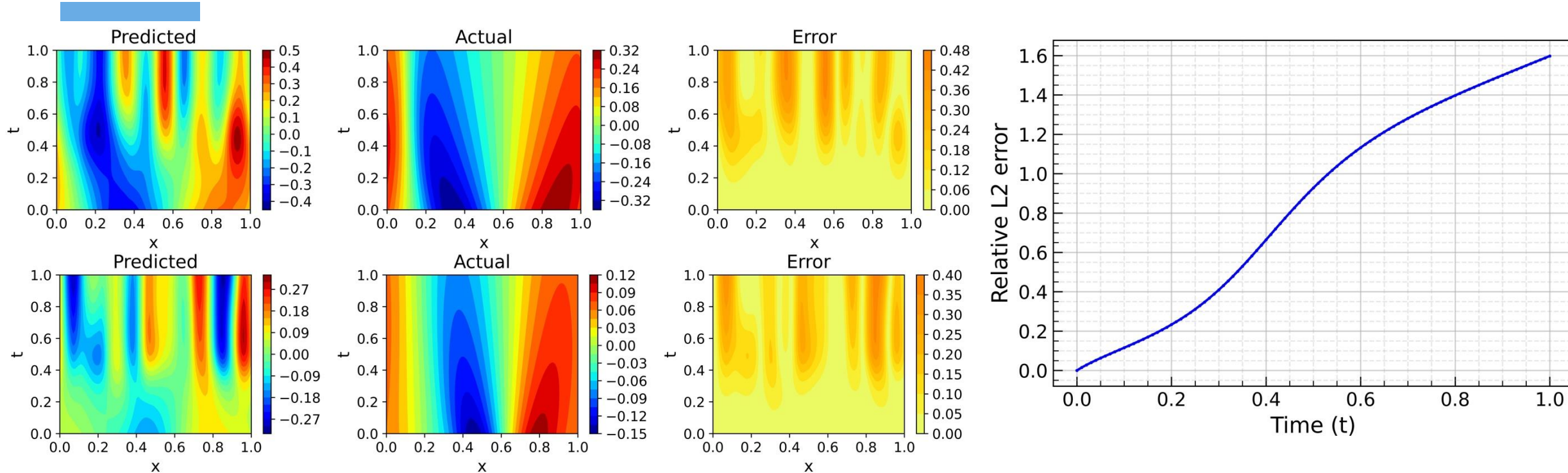
- Ground Truth Output:**  $[u^1(x), u^2(x), u^3(x), \dots, u^{50}(x)]$

- Trunk Input:**  $x \in [0, 1]$

- Goal:** learn the operator mapping  $u^i(x)$  to  $u^{i+1}(x)$ , i.e.,

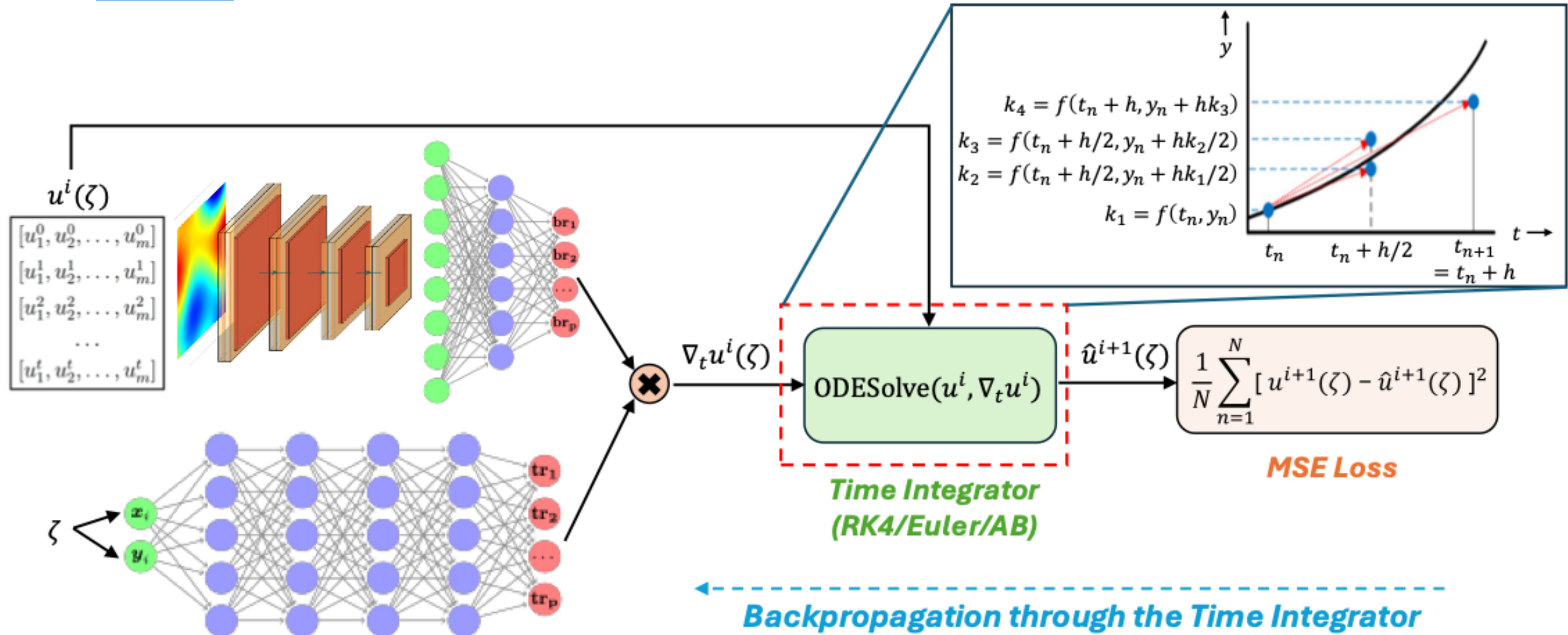
$$\mathcal{G} \sim \mathcal{G}_\theta: u(t = i, x) \rightarrow u(t = i + 1, x)$$

# Autoregressive Errors

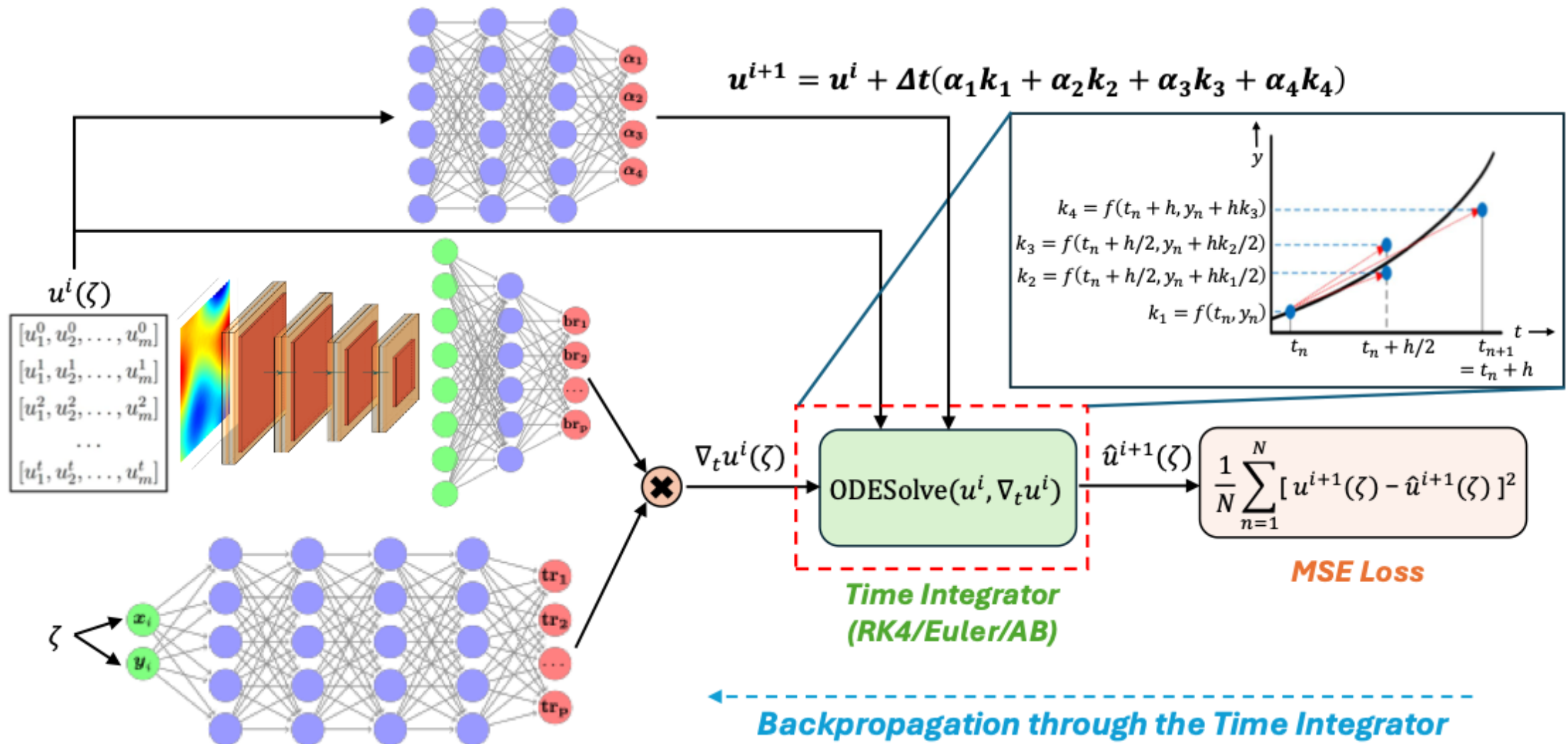


- Network trained with an Adam optimizer (Learning Rate = 0.001) until convergence
- Prediction done in a recursive manner starting from  $u^0(x) \Rightarrow$  predicted  $u^{i+1}(x)$  becomes input  $u^i(x)$  for next timestep prediction
- **Relative  $L_2$  error between predictions and ground truth grows quickly as we progress in time. This is also evident in the predicted contours.**

# Proposed Architecture: (1) TI-DeepONet



# Proposed Architecture: (2) TI(L)-DeepONet



# TI(L)-DeepONet (cont.)

- Instead of using fixed coefficients for the RK4 slopes, i.e.,  $\frac{1}{6}, \frac{2}{6}, \frac{2}{6}, \frac{1}{6}$  for  $k_1, k_2, k_3, k_4$ , we use adaptive coefficients,  $\alpha = [\alpha_1, \alpha_2, \alpha_3, \alpha_4]$ , which is a function of the current state  $u_i$ .
- The adaptive RK4 update becomes:

$$u^{i+1} = u^i + \Delta t(\alpha_1 k_1 + \alpha_2 k_2 + \alpha_3 k_3 + \alpha_4 k_4)$$

- The RK4 slope coefficients are learnt through an auxiliary neural network:  $\alpha = NN(u^i; \theta)$
- We compare the extrapolation performance of the different frameworks for the following PDE systems:
  - 1D viscous Burgers' Equation
  - 1D Korteweg-de Vries (KdV) Equation
  - 1D Kuramoto-Sivashinsky (KS) Equation
  - 2D viscous Burgers' Equation
  - 2D Allen-Cahn Equation
- We use a relative  $L_2$  error metric for evaluating the prediction errors:

$$\varepsilon_{L_2} = \frac{\|u_{pred} - u_{true}\|_2}{\|u_{true}\|_2}$$

# One-dimensional viscous Burgers' Equation

- The 1D viscous Burgers' PDE is defined as:

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2}, \quad (t, x) \in [0, 1] \times [0, 1]$$

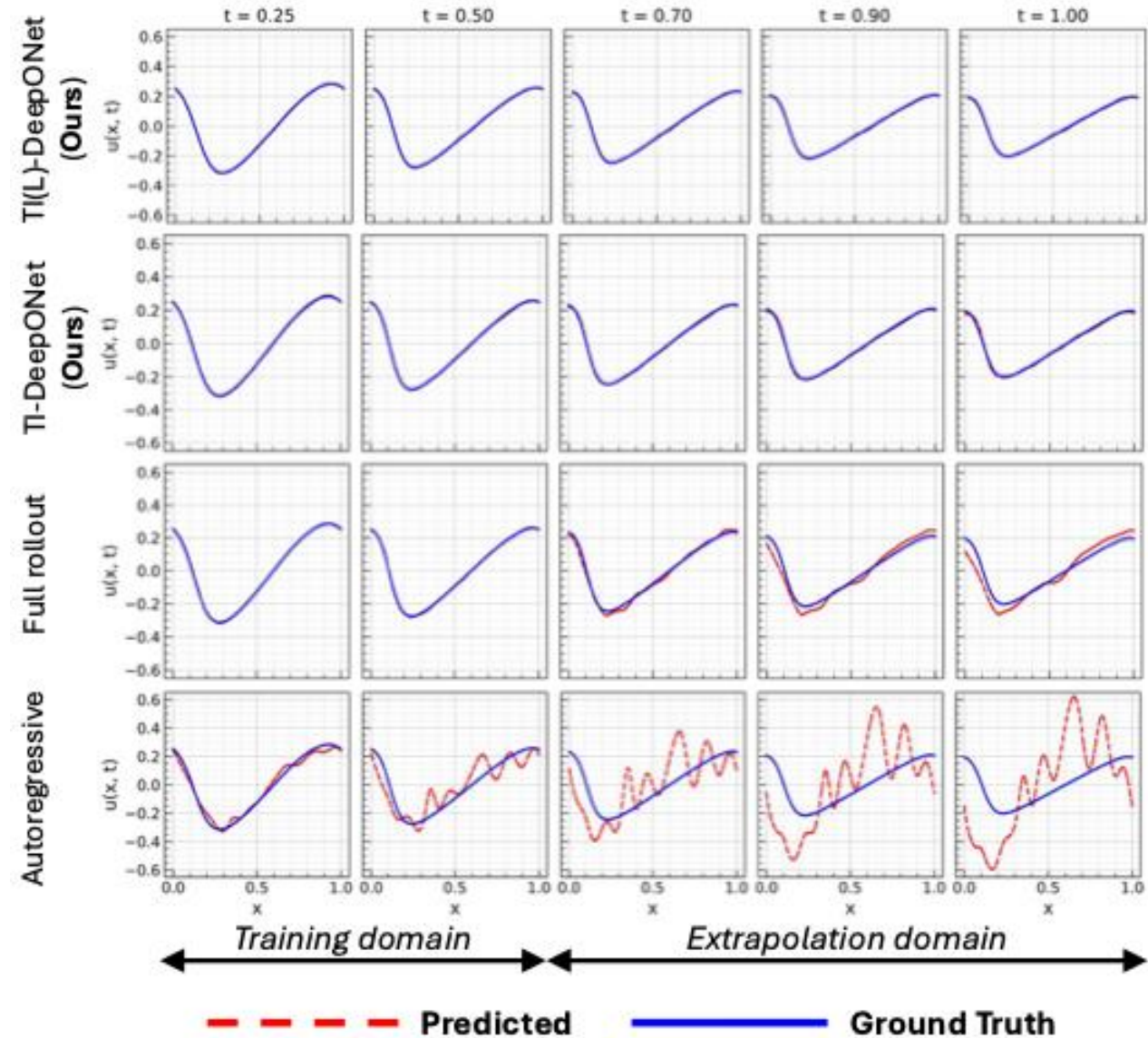
- Coefficient of viscosity:  $\nu = 0.01$
- **Initial conditions:** GRF with Matern' Kernel,  $u(x, t = 0) = s(x)$
- **Boundary conditions:** Periodic,

$$(1) u(x = 0, t) = u(x = 1, t) \quad (2) \frac{\partial u}{\partial x}(x = 0, t) = \frac{\partial u}{\partial x}(x = 1, t)$$

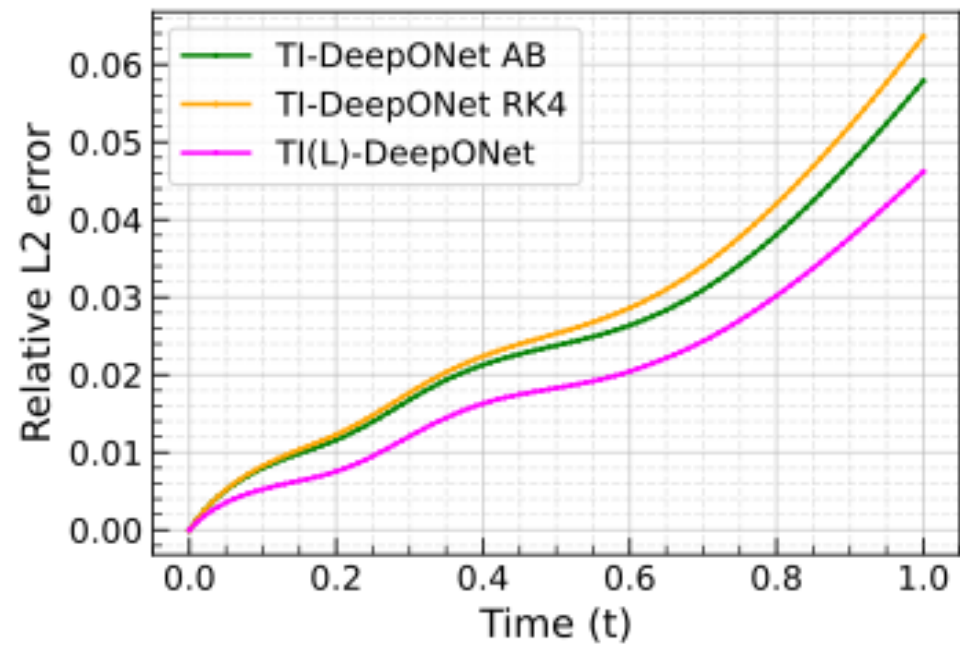
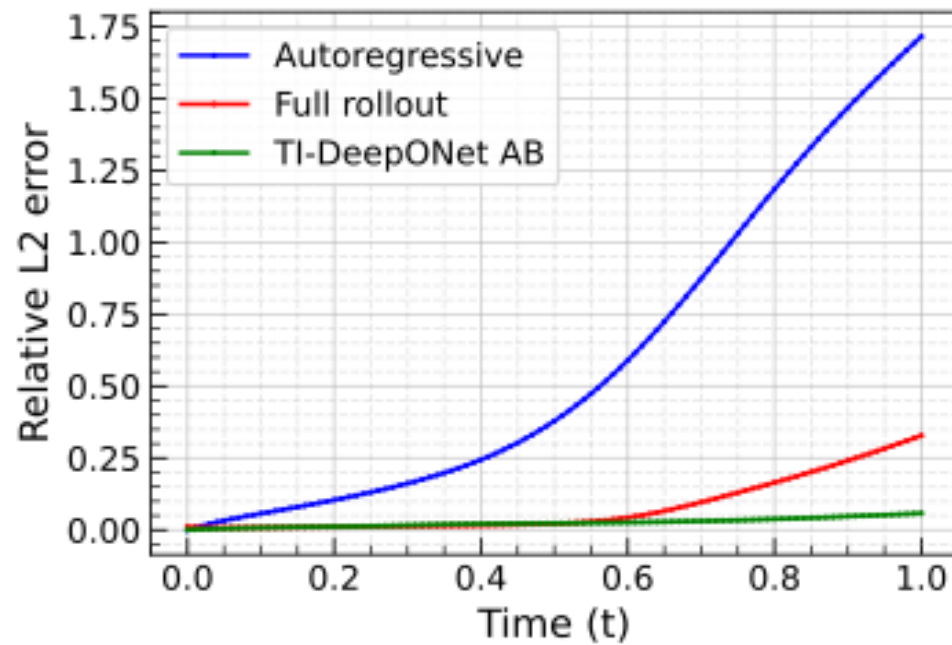
- $N_s = 2500$  samples of ICs,  $N_t = 101$  timesteps,  $N_x = 101$  spatial grid points
- Training time domain:  $t \in [0, 0.5]$
- Extrapolation time domain:  $t \in [0.5, 1.0]$
- **Characteristics of the PDE:**
  - Nonlinear parabolic with convective ( $uu_x$ ) and dissipative ( $\nu u_{xx}$ ) terms
  - Simplified form of the Navier-Stokes Equations that govern flow physics

# Results (1/2)

- Prediction of the solution profiles by the four different frameworks: (1) **TI(L)-DeepONet**, (2) **TI-DeepONet**, (3) **Full rollout**, and (4) **Autoregressive**
- **Autoregressive** accumulates errors early on and quickly deviates from the actual profile
- **Full rollout** performs well in training domain but starts incurring errors upon entering the extrapolation domain
- **Both TI-DeepONet and TI(L)-DeepONet ensure stable and accurate extrapolation of future solution states**
- **TI(L)-DeepONet** slightly performs better due to its adaptivity to the local solution



## Results (2/2)



- **Autoregressive:** Rapid error accumulation and resembles exponential error growth beyond  $t = 0.5$ .
- **Full rollout:** Performs well until  $t = 0.5$  and then starts incurring errors.
- **TI-DeepONet + AB2/AM3 inference:** Stable and controlled error growth especially in extrapolation domain
- **Within the time integrator frameworks:** TI(L)-DeepONet performs best followed by TI-DeepONet AB and then TI-DeepONet RK4

# One-dimensional Korteweg De-Vries (KdV) Equation

- The 1D KdV PDE is defined as:

$$\frac{\partial u}{\partial t} - \eta u \frac{\partial u}{\partial x} + \gamma \frac{\partial^3 u}{\partial x^3} = 0, \quad (t, x) \in [0, 5] \times [0, 10]$$

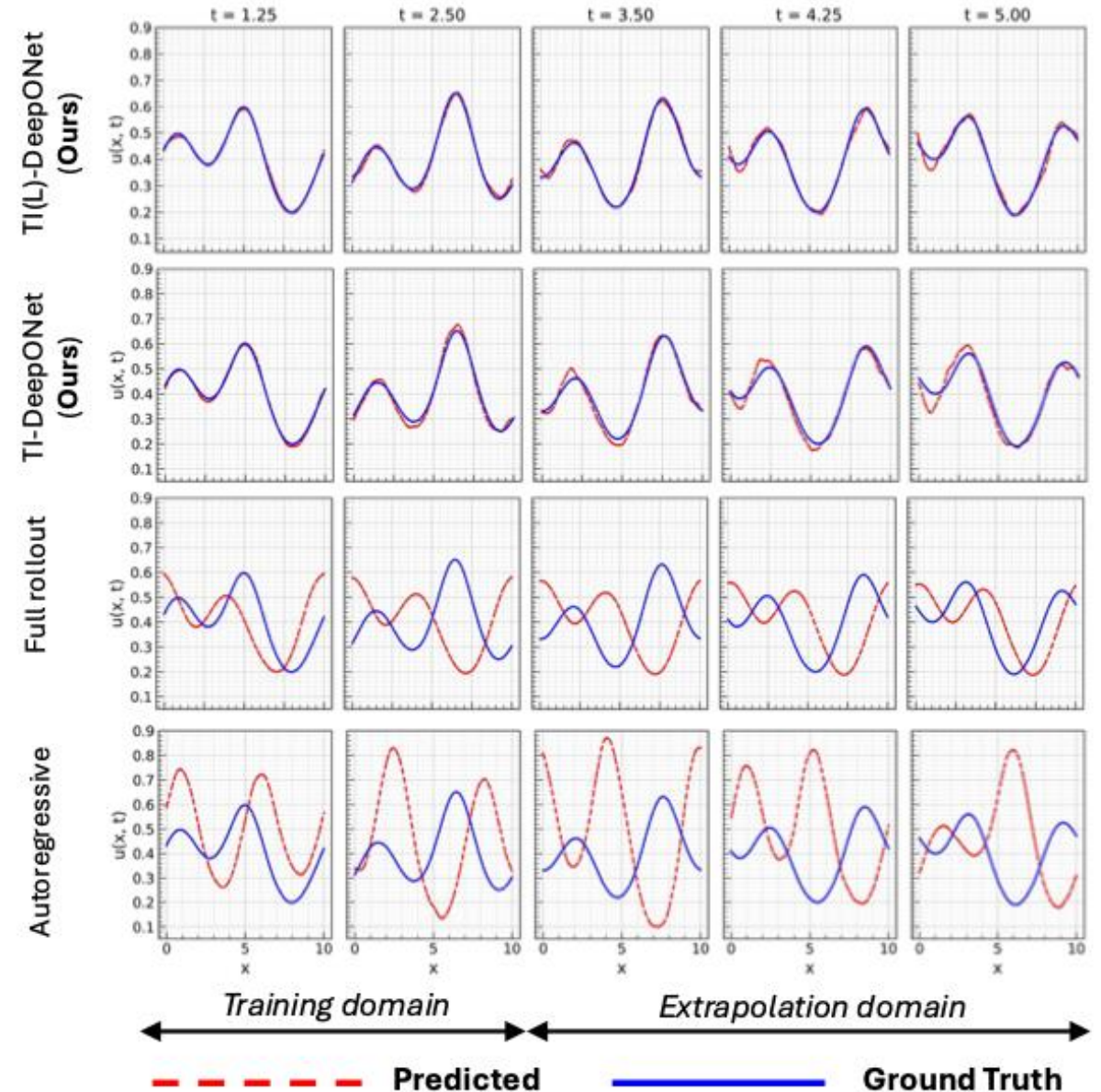
- **Initial conditions:** Sum of two solitons (nonlinear, self-stabilizing local wave packets)

$$u_i(x, 0) = 2k_i^2 \operatorname{sech}^2 \left( k_i \left( \left( x + \frac{P}{2} - Pd_i \right) \% P - \frac{P}{2} \right) \right)^2$$

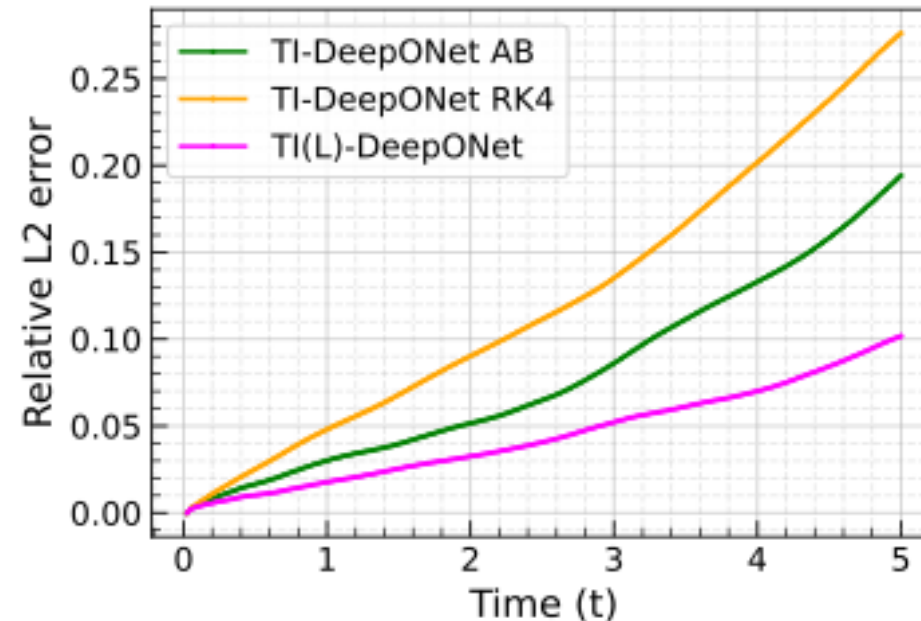
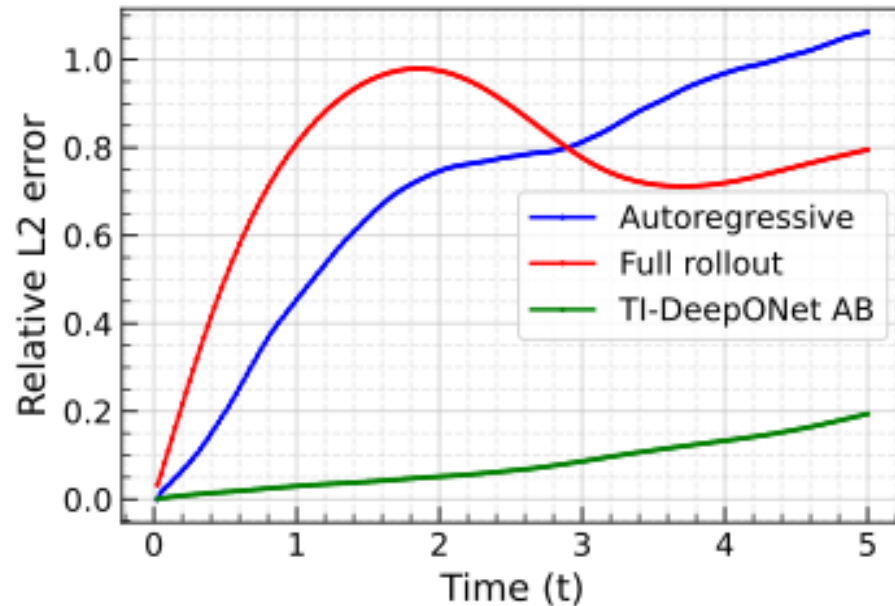
- P: Period of wave,  $k \in [0.5, 1.0]$ ,  $d \in [0, 1]$  – Coefficients that determine the height of the peak and location of the soliton,  $i \in \{0, 1\}$  – Index for the soliton
- $N_s = 1000$  samples of ICs,  $N_t = 201$  timesteps,  $N_x = 100$  spatial grid points
- Training time domain:  $t \in [0, 2.5]$
- Extrapolation time domain:  $t \in [2.5, 5.0]$
- **Characteristics of the PDE:**
  - Nonlinear hyperbolic steepening term ( $\eta u u_x$ ) and dispersive ( $u_{xxx}$ ) dynamics
  - Models wave propagation on shallow water surfaces. Admits soliton solutions.

# Results (1/2)

- Prediction of the solution profiles by the four different frameworks: (1) **TI(L)-DeepONet**, (2) **TI-DeepONet**, (3) **Full rollout**, and (4) **Autoregressive**
- Both **autoregressive** and **full rollout DeepONet** models fail to capture the dispersive, periodic dynamics of the solution
- **Both TI-DeepONet and TI(L)-DeepONet are able to capture the underlying nonlinear dispersive dynamics and thereby translate well to the extrapolation domain**
- **TI(L)-DeepONet** slightly performs better due to its adaptivity to the local solution, but the difference in this case is minimal with **TI-DeepONet** performing better in some cases



## Results (2/2)



- **Autoregressive:** Rapid and monotonically increasing error
- **Full rollout:** Non-monotonic error trend. Error increases until  $t = 1.75$  and then slightly falls only to increase again later
- **TI-DeepONet AB:** Stable and controlled error growth in the extrapolation domain
- **Within the different time integrator frameworks:** TI(L)-DeepONet and TI-DeepONet AB exhibit similar performance followed by TI-DeepONet with RK4 inference

# One-dimensional Kuramoto-Sivashinsky (KS) Equation

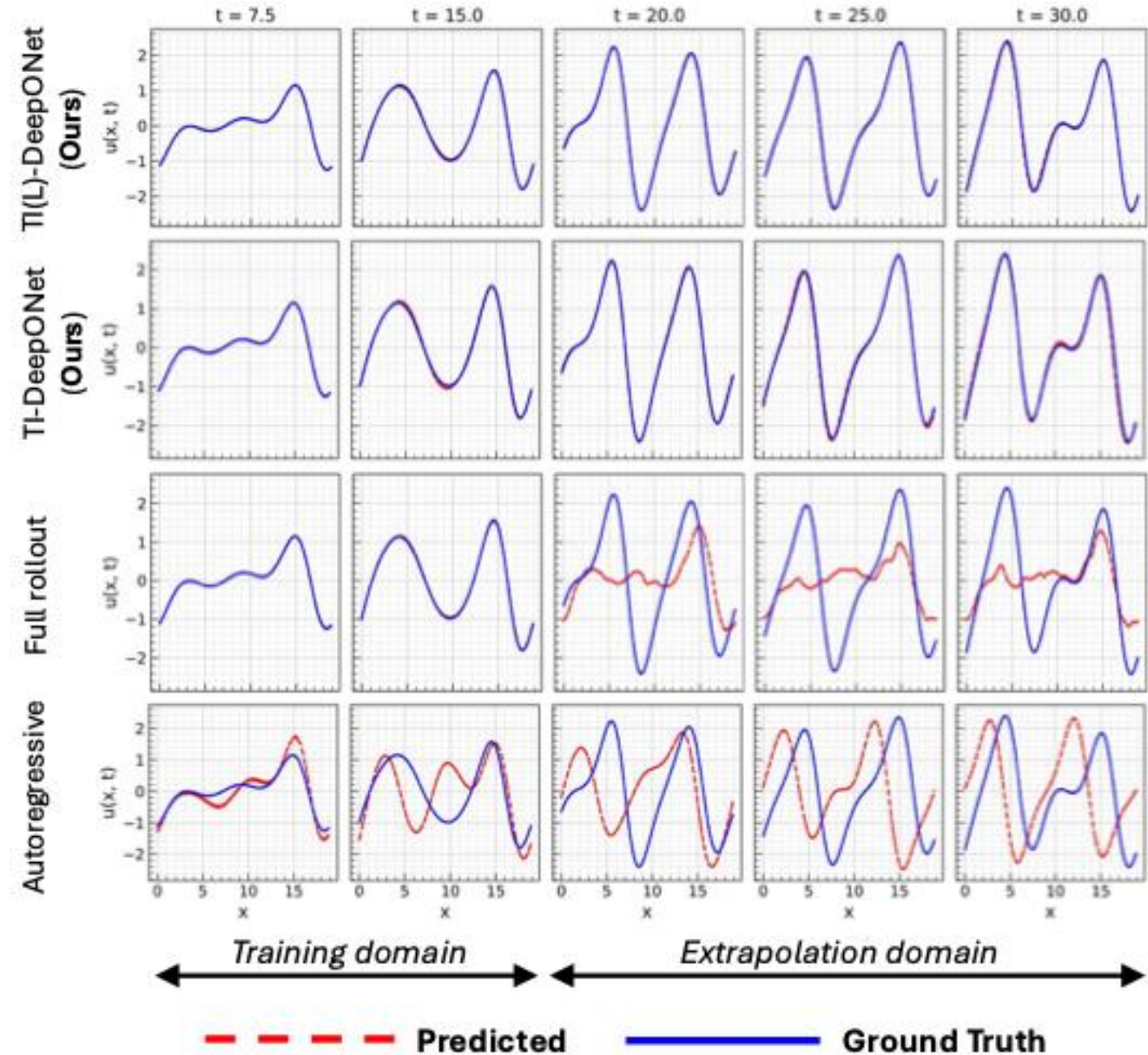
- The 1D KS PDE is defined as:

$$\frac{\partial u}{\partial t} = -u \frac{\partial u}{\partial x} - \frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial x^4}, \quad (t, x) \in [0, \infty) \times [0, 6\pi]$$

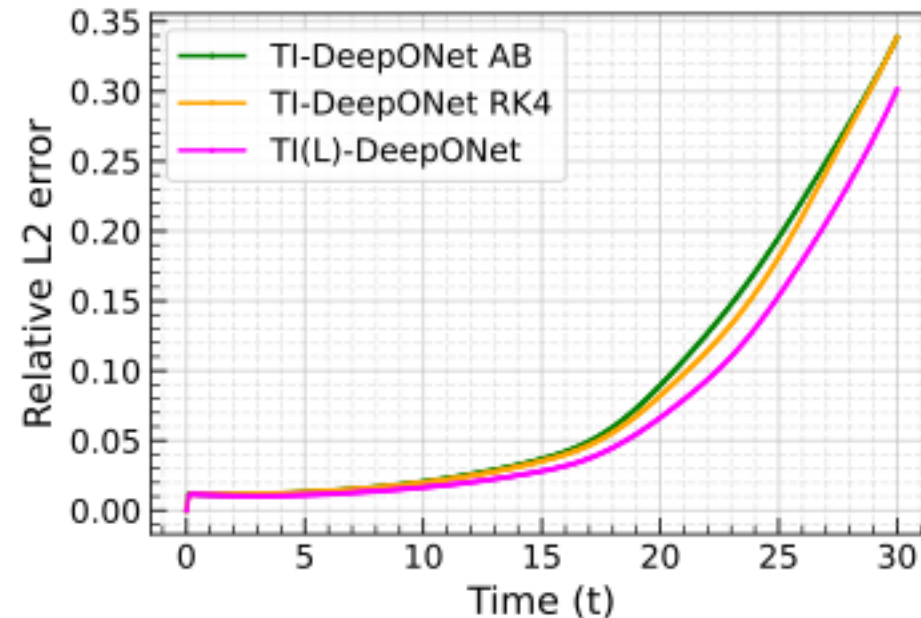
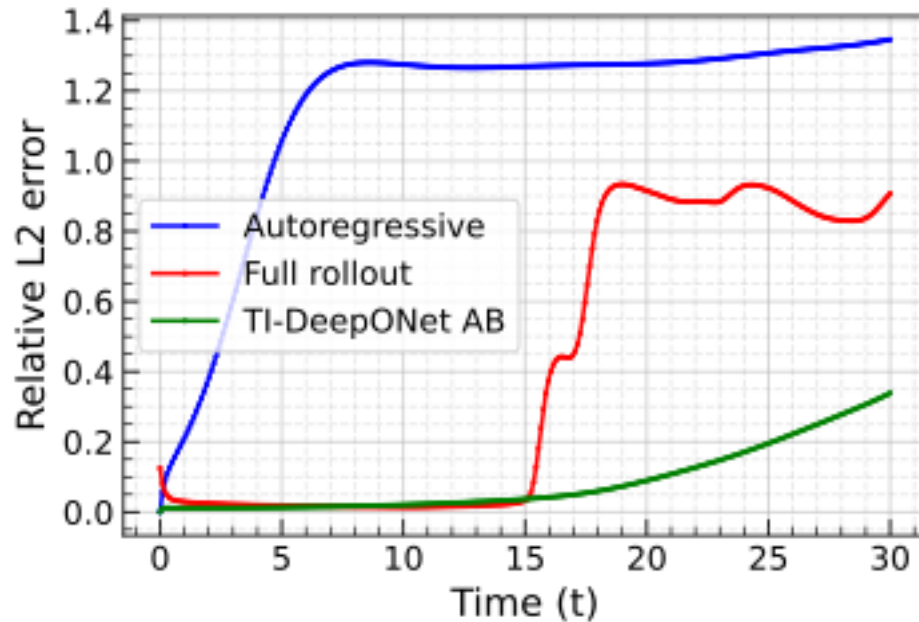
- **Initial condition:**  $u(x, t = 0) = u_0(x)$
- **Boundary conditions:** Periodic
- $N_s = 3000$  samples of ICs,  $N_t = 300$  timesteps,  $N_x = 128$  spatial grid points
- Training time domain:  $t \in [0, 15]$
- Extrapolation time domain:  $t \in [15, 30]$
- **Characteristics of the PDE:**
  - Nonlinear hyperbolic convection ( $uu_x$ ), destabilizing anti-diffusion ( $-u_{xx}$ ) due to negative viscosity, and stabilizing hyper-diffusion ( $u_{xxxx}$ ) term leading to dissipation at small scales and self-organization into a chaotic state
  - Overall, nonlinear, dispersive-diffusive PDE with chaotic dynamics
  - Models instabilities in a laminar flame front and is a prototypical toy model for turbulence

# Results (1/2)

- Prediction of the solution profiles by the four different frameworks: (1) **TI(L)-DeepONet**, (2) **TI-DeepONet**, (3) **Full rollout**, and (4) **Autoregressive**
- **Autoregressive** accumulates errors early on and quickly deviates from the actual profile
- **Full rollout** performs extremely well in the training domain but evidently starts incurring errors upon entering the extrapolation domain with a clear separation between the two temporal domains observed
- **Both TI-DeepONet and TI(L)-DeepONet ensure stable and accurate extrapolation of future chaotic states**
- **TI(L)-DeepONet** slightly performs better due to its adaptivity to the local solution and adapts well to the stiff dynamics



## Results (2/2)



- **Autoregressive:** Rapid error accumulation early on and error grows quickly and eventually saturates.
- **Full rollout:** Performs very well until  $t = 15$  and then quickly starts incurring errors for  $t \in [15, 30]$ .
- **TI-DeepONet + AB2/AM3 inference:** Stable and (relatively) controlled error growth especially in the extrapolation domain leading to a decent prediction of the chaotic solution states.
- **Within the time integrator frameworks:** TI(L)-DeepONet performs best followed by TI-DeepONet RK4 and then TI-DeepONet AB

# Two-dimensional viscous Burgers' Equation

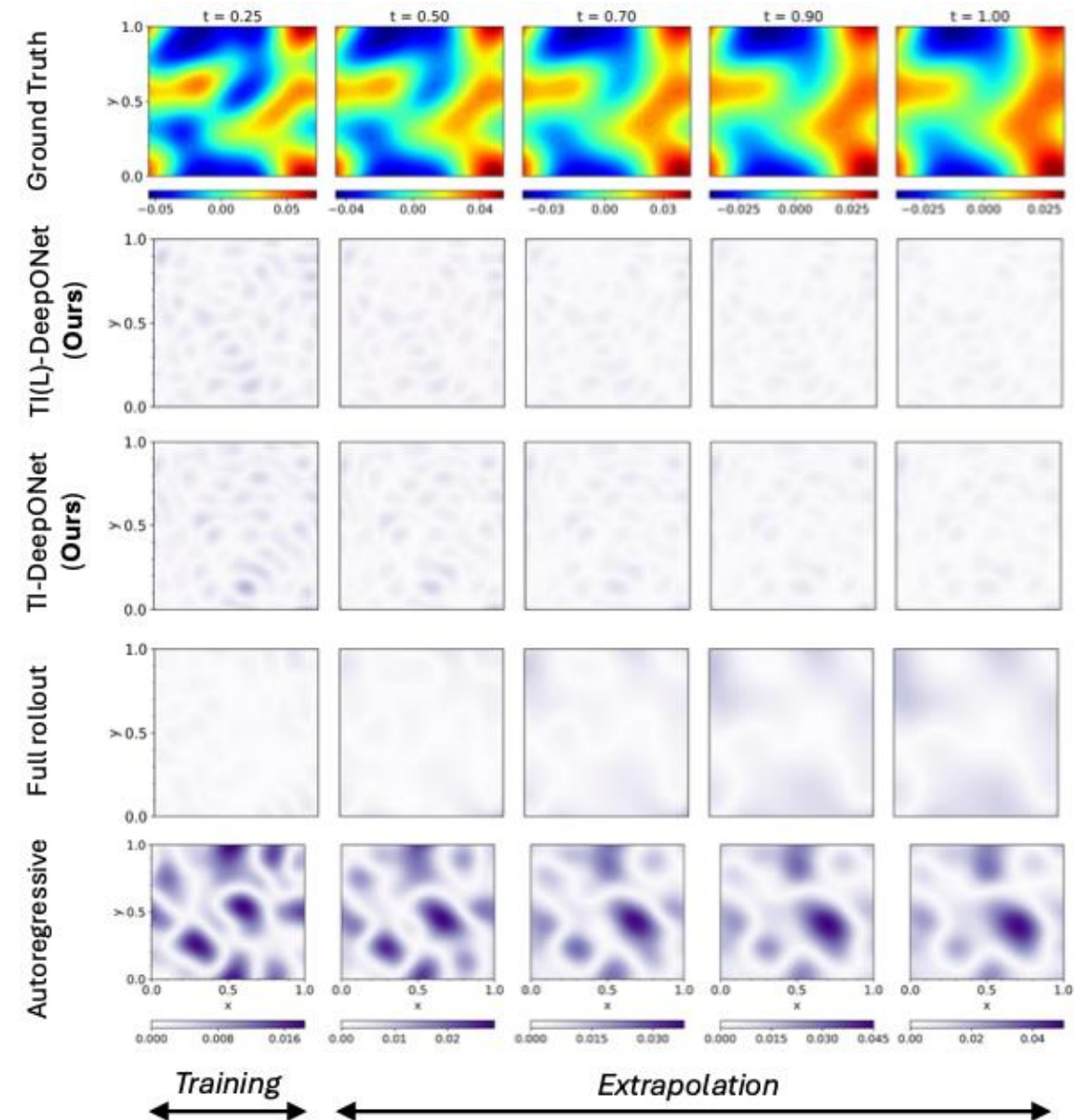
- The 2D viscous Burgers' PDE is defined as:

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + u \frac{\partial u}{\partial y} = \nu \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right), \quad (t, x, y) \in [0, 1] \times [0, 1]^2$$

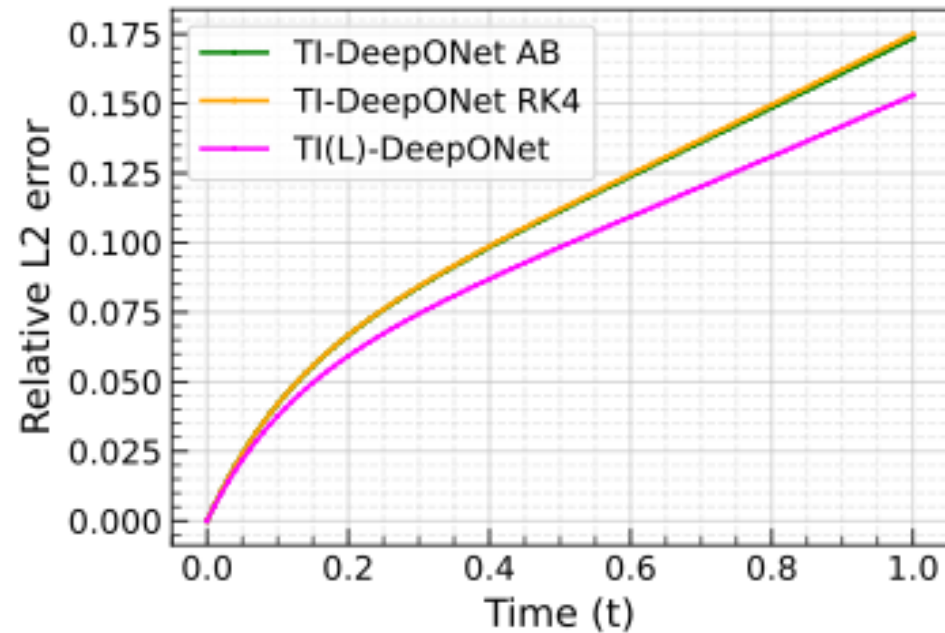
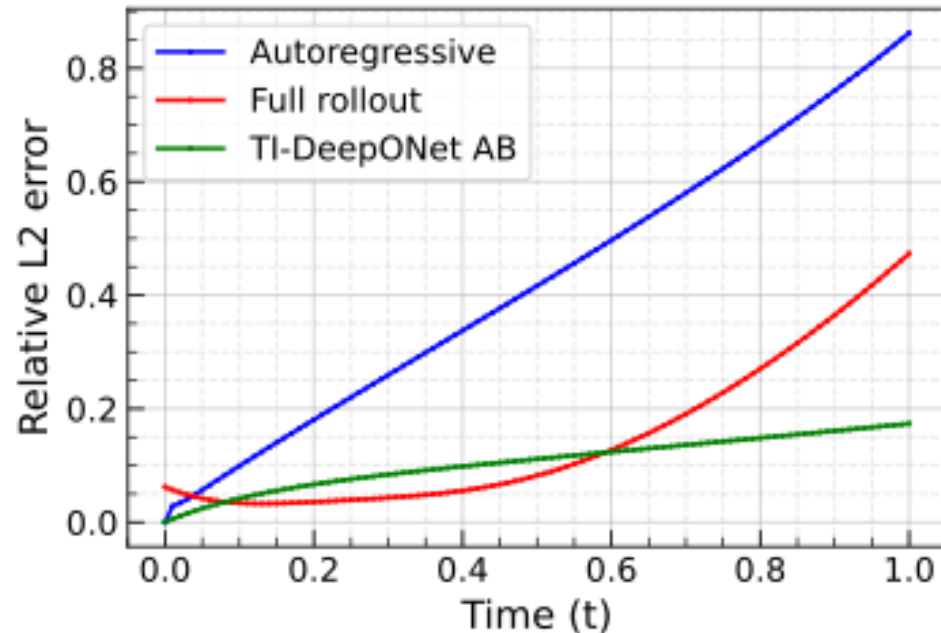
- Coefficient of viscosity:  $\nu = 0.01$
- **Initial conditions:** 2D GRF -  $u(x, y, t = 0) = s(x, y)$
- **Boundary conditions:** Periodic at both boundaries of the unit-square domain
- $N_s = 1000$  samples of ICs,  $N_t = 101$  timesteps,  $N_x = N_y = 32$  spatial grid points
- Training time domain:  $t \in [0, 0.33]$
- Extrapolation time domain:  $t \in [0.33, 1]$
- **Characteristics of the PDE:**
  - Multi-dimensional nonlinear parabolic convective-diffusive system
  - Extension of the 1D viscous Burgers' Equation with scalar output field in two-dimensions

# Results (1/2)

- Prediction of the solution profiles by the four different frameworks: (1) **TI(L)-DeepONet**, (2) **TI-DeepONet**, (3) **Full rollout**, and (4) **Autoregressive**
- **Autoregressive** accumulates errors early on and quickly deviates from the actual profile at the first few timesteps itself.
- **Full rollout** performs well up to  $t = 0.5$  (slightly extending beyond the training domain) and then starts incurring errors in the longer time-steps. Notably, it is slightly better than the TI-based frameworks in the training domain also.
- **Both TI-DeepONet and TI(L)-DeepONet incur relatively lower errors in the extrapolation regime with TI(L)-DeepONet performing slightly better due to its adaptivity to the local solution.**



## Results (2/2)



- **Autoregressive:** Rapid error accumulation with a monotonically increasing trend
- **Full rollout:** Initially performs better but its fixed basis limits temporal generalization and errors surpass TI-DeepONet beyond  $t = 0.5$ .
- **Overall, TI(L)-DeepONet yields the best performance with a final error at the last timestep to be approximately 15%.**

# Two-dimensional Allen-Cahn Equation

- The 2D Allen-Cahn PDE is defined as:

$$\frac{\partial u}{\partial t} = \epsilon^2 \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) - (u^3 - u)$$

- Interfacial width or diffusion length:  $\epsilon = 0.05$

- Initial conditions: GRF with Matern' Kernel

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

- **Boundary conditions:** Periodic at both boundaries of the unit-square domain

- $N_s = 1000$  samples of ICs,  $N_t = 101$  timesteps,  $N_x = N_y = 32$  spatial grid points

- Training time domain:  $t \in [0, 0.33]$

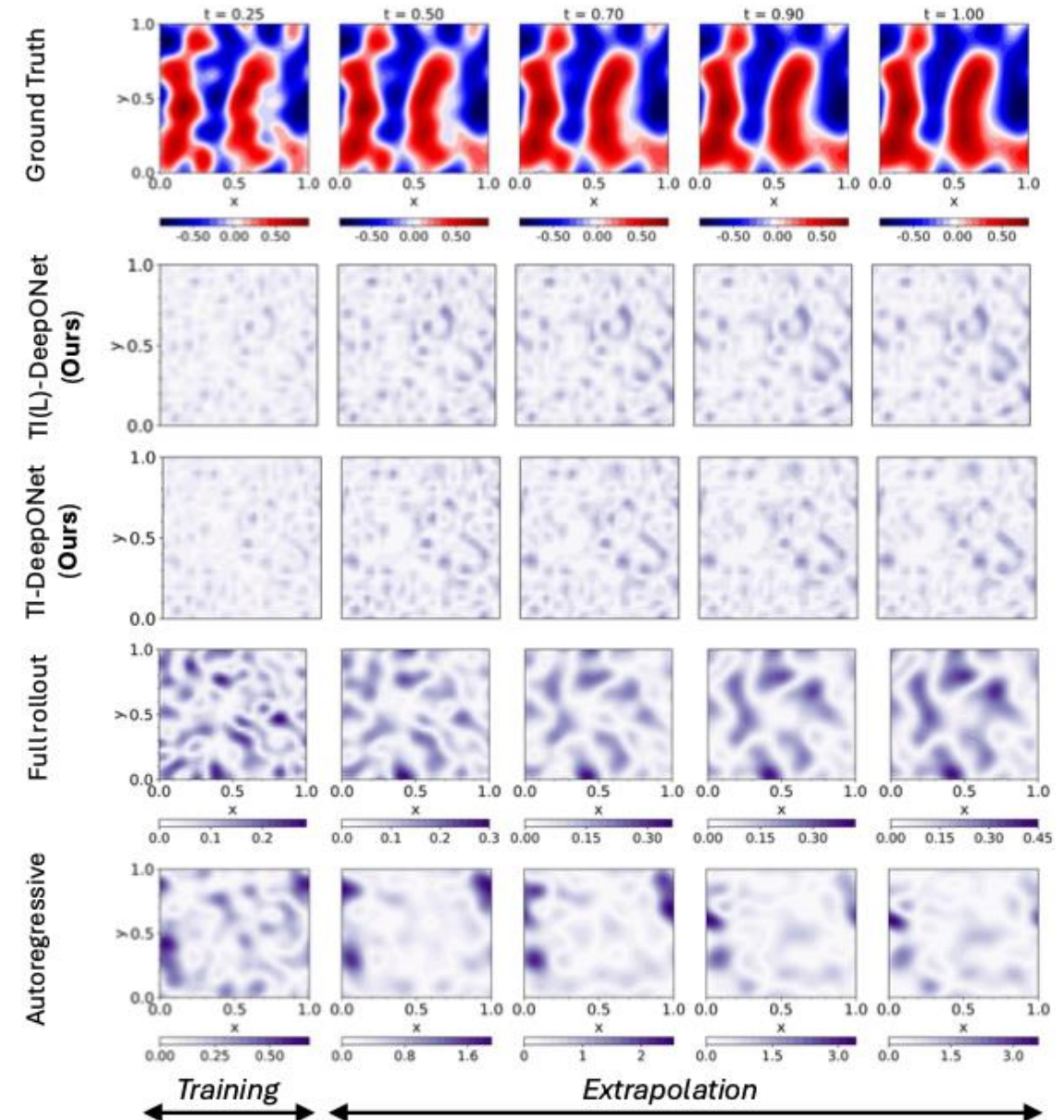
- Extrapolation time domain:  $t \in [0.33, 1]$

- **Characteristics of the PDE:**

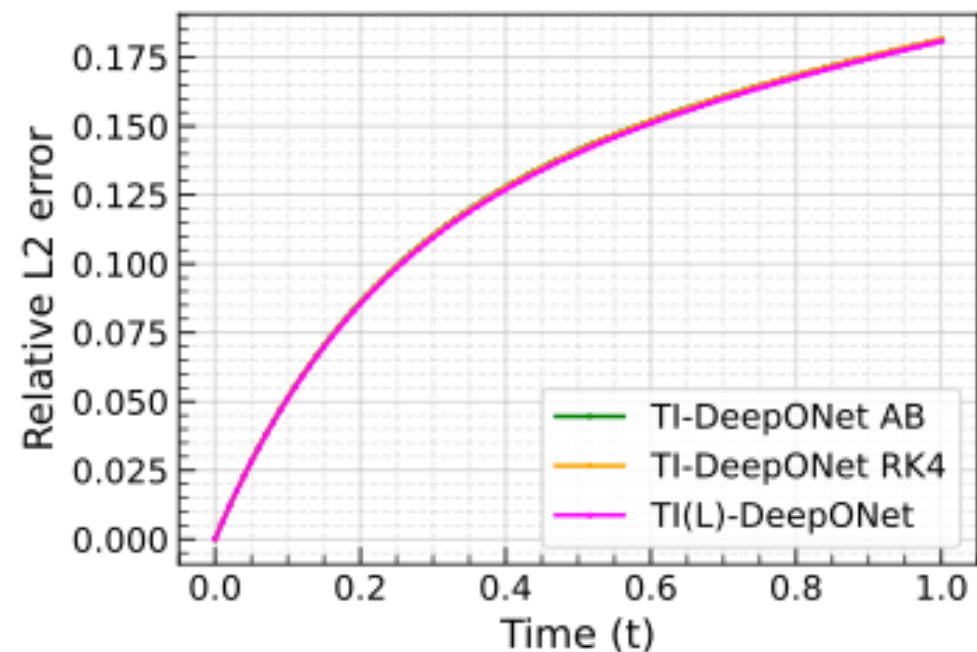
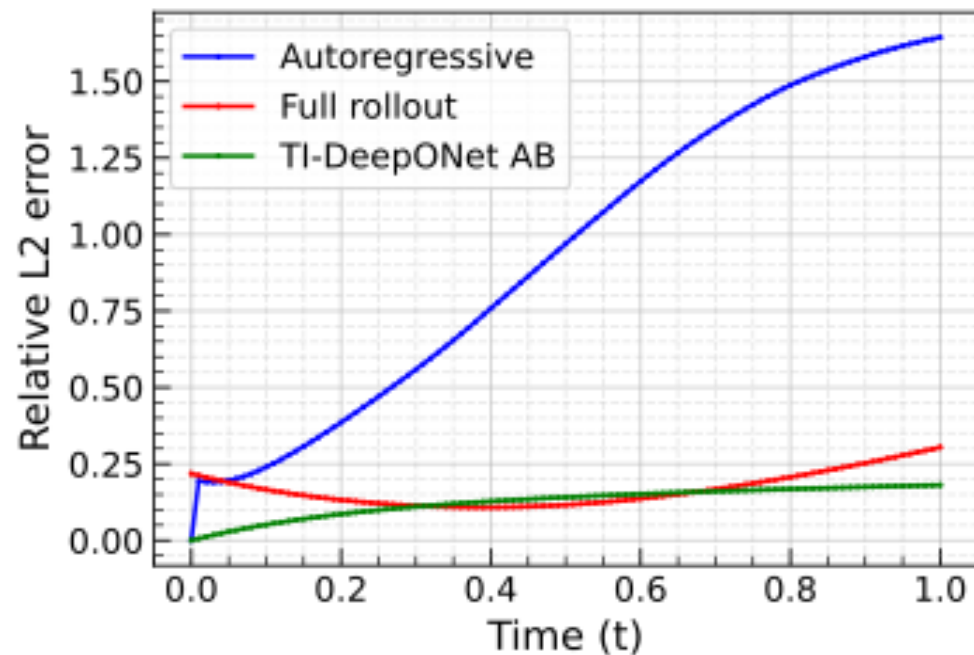
- Nonlinear parabolic PDE with bistable states
- Interplay between a diffusion term ( $\epsilon^2 \nabla^2 u$ ) and a reaction term ( $-(u^3 - u)$ )
- Prototype model for phase separation and interface dynamics;  $L^2$  gradient flow of the Ginzburg-Landau free energy functional

# Results (1/2)

- Prediction of the solution profiles by the four different frameworks: (1) **TI(L)-DeepONet**, (2) **TI-DeepONet**, (3) **Full rollout**, and (4) **Autoregressive**
- **Autoregressive** accumulates errors early on and quickly deviates from the actual profile at the first few timesteps itself. The errors are orders of magnitude higher than the others.
- **Full rollout** performs decent as opposed to autoregressive but not as good as the TI-based frameworks. The errors are higher in the initial few timesteps.
- **Both TI-DeepONet and TI(L)-DeepONet incur relatively lower errors in both the training and extrapolation regime. Thus, they are able to capture the phase separation/interface dynamics. The performance difference between them is minimal.**



## Results (2/2)



- **Autoregressive:** Error accumulation is visibly high, and the error compounding happens at the beginning timesteps itself.
- **Full Rollout:** Incurs larger errors for the initial timesteps. The error shows an initial decrease with it slightly performing better than TI-DeepONet in  $t \in [0.35, 0.65]$ . Beyond that, TI-DeepONet edges ahead.
- **Within the time-integrator frameworks:** All of them exhibited nearly same performance with TI(L)-DeepONet showing minimal improvement over the others.

# Summary

Problem	$t_{train}^*$	$\Delta t_e^*$	Method	Relative $L_2$ error			
				$t+10\Delta t_e$	$t+20\Delta t_e$	$t+40\Delta t_e$	$T^*$
Burgers' (1D)	0.5	0.01	TI(L)-DON [Ours]	<b>0.0204</b>	<b>0.0243</b>	<b>0.0377</b>	<b>0.0462</b>
			TI-DON AB [Ours]	0.0264	0.0310	0.0473	0.0579
			DON Full Rollout	0.0433	0.0965	0.2413	0.3281
			DON Autoregressive	0.5898	0.8742	1.4682	1.7154
KdV (1D)	2.5	0.05	TI(L)-DON [Ours]	<b>0.0522</b>	<b>0.0612</b>	<b>0.0701</b>	<b>0.1017</b>
			TI-DON AB [Ours]	0.0861	0.1114	0.1330	0.1941
			DON Full Rollout	0.7769	0.7163	0.7197	0.7951
			DON Autoregressive	0.8127	0.8985	0.9691	1.0626
KS (1D)	15	0.3	TI(L)-DON [Ours]	<b>0.0445</b>	<b>0.079</b>	<b>0.2056</b>	<b>0.3013</b>
			TI-DON AB [Ours]	0.0589	0.1066	0.2481	0.3366
			DON Full Rollout	0.8298	0.8917	0.8482	0.9073
			DON Autoregressive	1.2744	1.2804	1.3204	1.3463
Burgers' (2D)	0.33	0.01	TI(L)-DON [Ours]	<b>0.1093</b>	<b>0.1202</b>	<b>0.1419</b>	<b>0.1531</b>
			TI-DON AB [Ours]	0.1238	0.1361	0.1609	0.1736
			DON Full Rollout	0.1275	0.1907	0.3649	0.4733
			DON Autoregressive	0.4969	0.5801	0.7604	0.8617
Allen-Cahn (2D)	0.33	0.01	TI(L)-DON [Ours]	0.1510	<b>0.1599</b>	<b>0.1745</b>	<b>0.1808</b>
			TI-DON AB [Ours]	0.1519	0.1607	0.1751	0.1813
			DON Full Rollout	<b>0.1365</b>	0.1675	0.2527	0.3040
			DON Autoregressive	1.1734	1.3499	1.5805	1.6437

\* $t_{train} = t$ : Beginning time of extrapolation;  $\Delta t_e$ : Evaluation timestep;  $T$ : Final prediction time.

# Conclusions

- Introduced **TI-DeepONet** and its adaptive variant **TI(L)-DeepONet**, which integrates classical numerical time integration with an operator learning framework (DeepONet).
- TI-DeepONet reduces relative  $L_2$  extrapolation errors by 81% compared to autoregressive DeepONet and 70% compared to full-rollout methods, while maintaining stable predictions for temporal domains extending up to twice/thrice the training interval.
- The learnable coefficients in TI(L)-DeepONet further refine accuracy by adaptively weighting intermediate integration slopes. This is especially beneficial for stiff, chaotic PDE dynamics.
- **Limitations:** Higher computational cost during training and inference due to multiple forward/backward passes through the ODE Solver



# Thank You! Questions?