



TENG: Time-Evolving Natural Gradient for Solving PDEs With Deep Neural Nets Toward Machine Precision

Zhuo Chen¹² Jacob McCarran^{1*} Esteban Vizcaino^{1*} Marin Soljačić¹² Di Luo¹²³

¹Massachusetts Institute of Technology

²NSF AI Institute for Artificial Intelligence and Fundamental Interactions

³Harvard University

July 1st, 2024



Zhuo Chen
Ph.D. Candidate at MIT
Junior Investigator at IAIFI



Prof. Marin Soljačić
Professor at MIT
Senior Investigator at IAIFI



Jacob McCarran
Undergraduate Student at MIT



Dr. Di Luo
IAIFI Fellow at MIT and Harvard

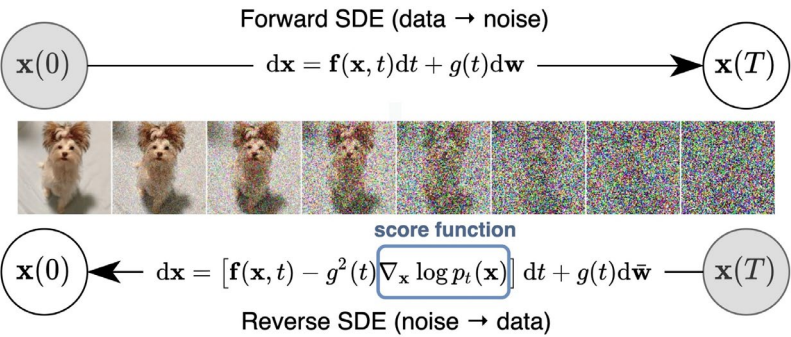
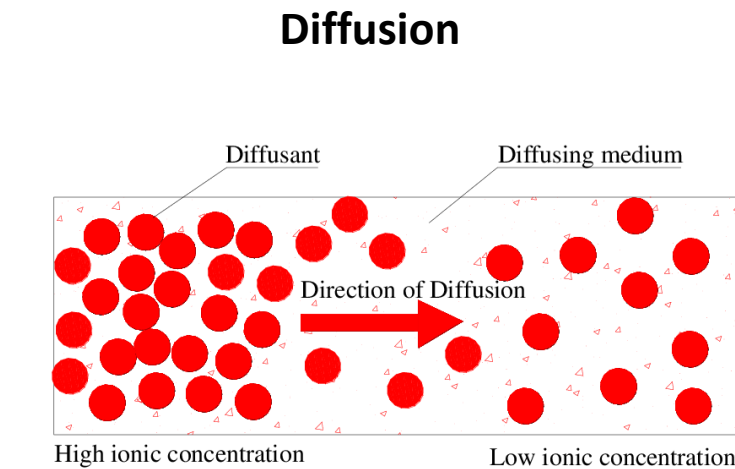
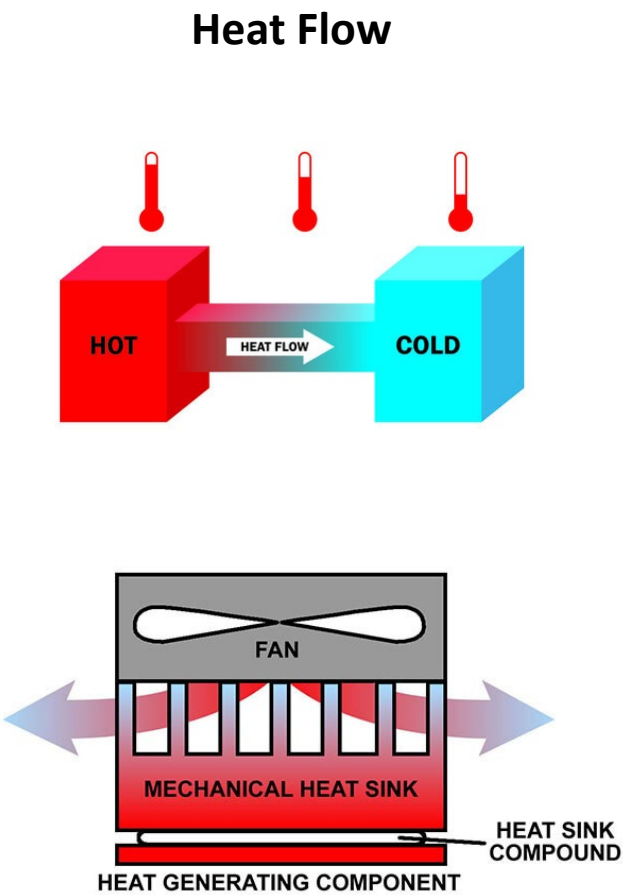


Esteban Vizcaino
Undergraduate Student at MIT

Research Opportunities Available in AI+Science
Contact <diluo@mit.edu>

PDEs Are Important

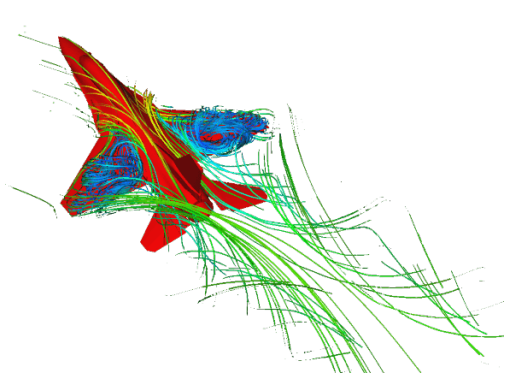
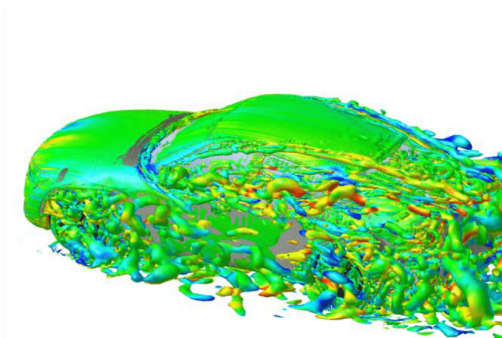
Heat/Diffusion Equation: $\frac{\partial u}{\partial t} = \nabla^2 u$



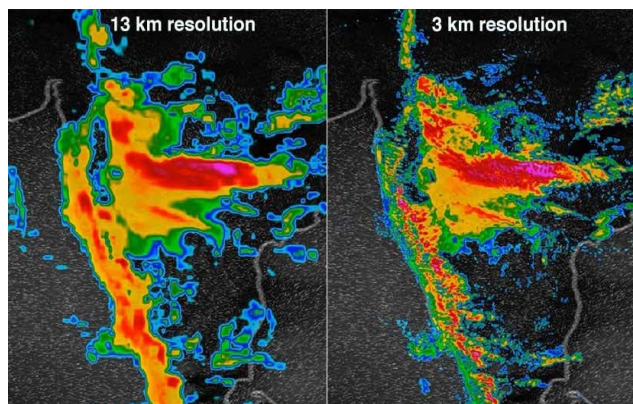
PDEs Are Important

Navier–Stokes Equation: $\rho \frac{D\vec{u}}{Dt} = -\nabla p + \nabla \cdot \left\{ \mu \left[\nabla \vec{u} + (\nabla \vec{u})^T - \frac{2}{3} (\nabla \cdot \vec{u}) \vec{I} \right] \right\} + \nabla [\zeta (\nabla \cdot \vec{u})] + \rho \vec{f}$

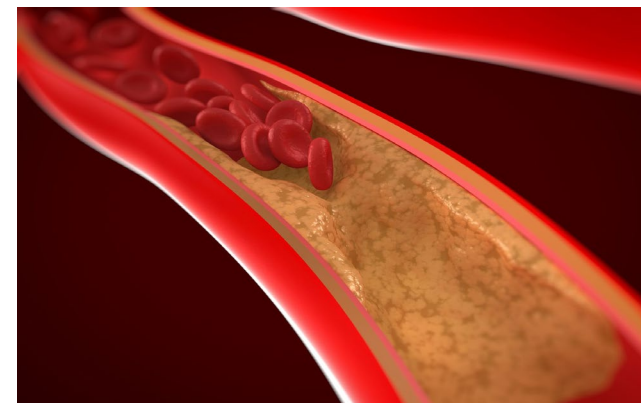
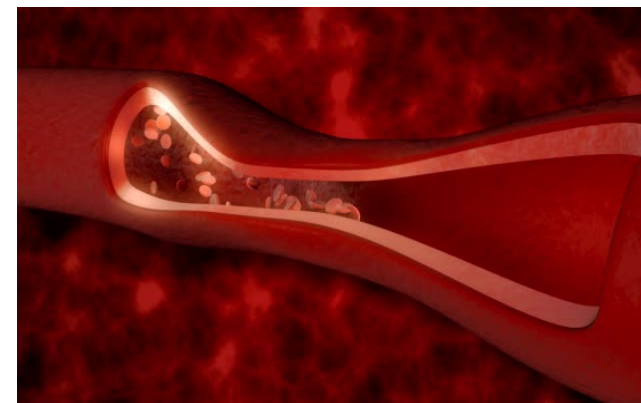
Aerodynamics



Weather Forecasting



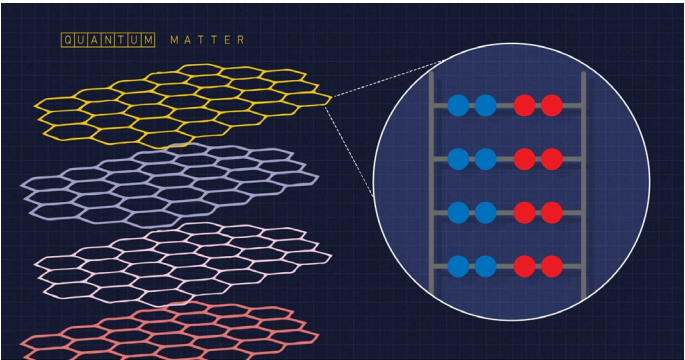
Blood Flow Modeling



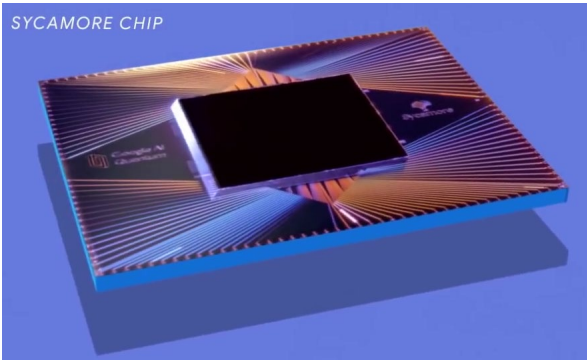
PDEs Are Important

Schrödinger’s Equation: $\frac{\partial u}{\partial t} = i[\nabla^2 u - V(x)u]$

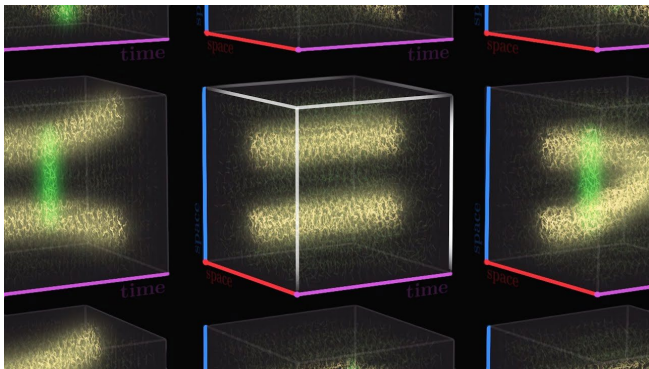
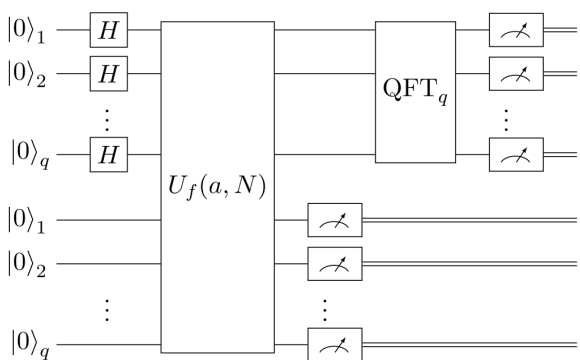
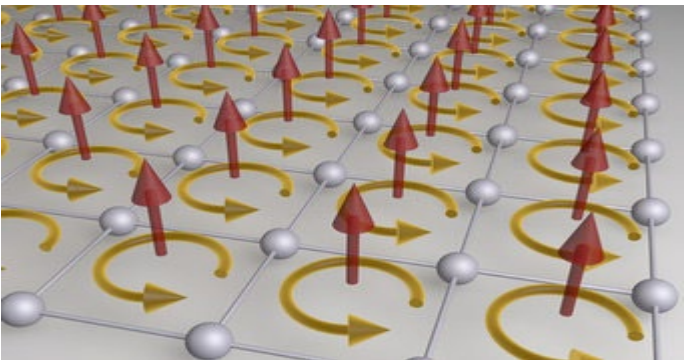
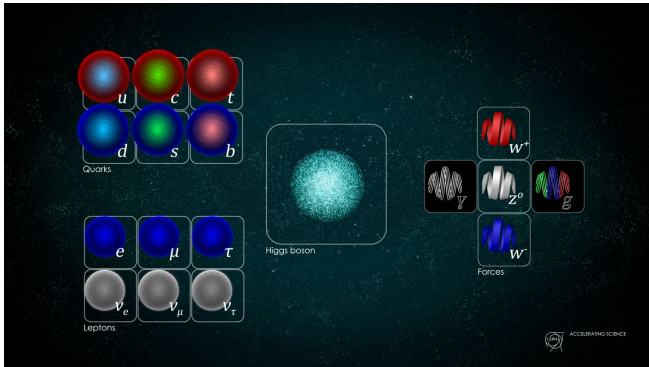
Quantum Matter



Quantum Computation



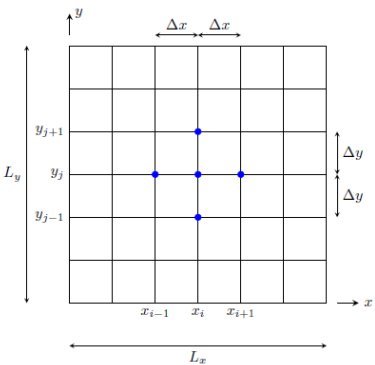
Quantum Field Theory



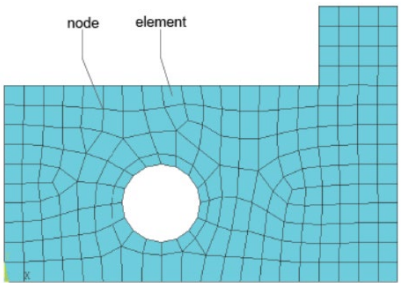
PDEs Can Be Challenging

Conventional Methods

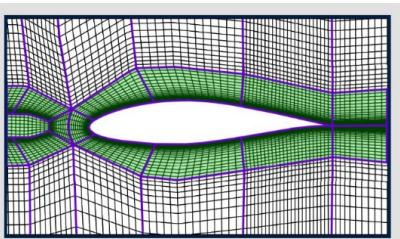
Finite Difference



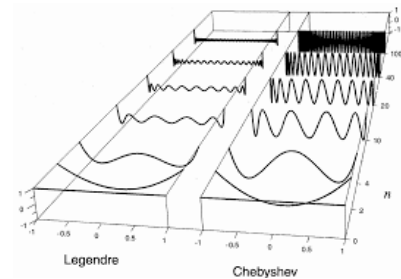
Finite Element



Finite Volume

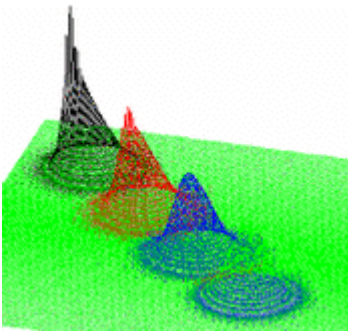


Spectral Method

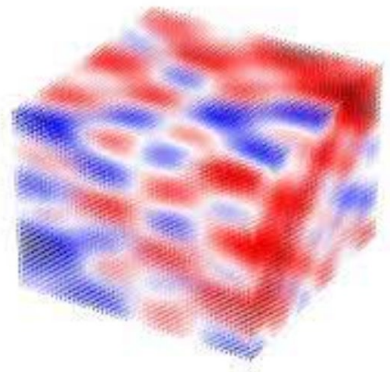


Challenges

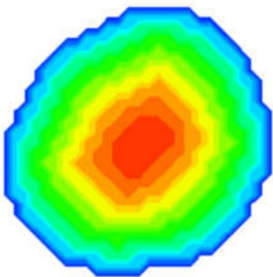
Non-Linear



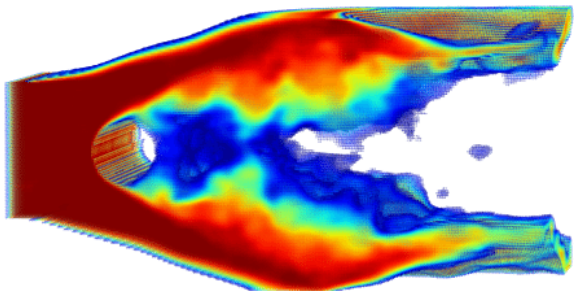
High Dimension



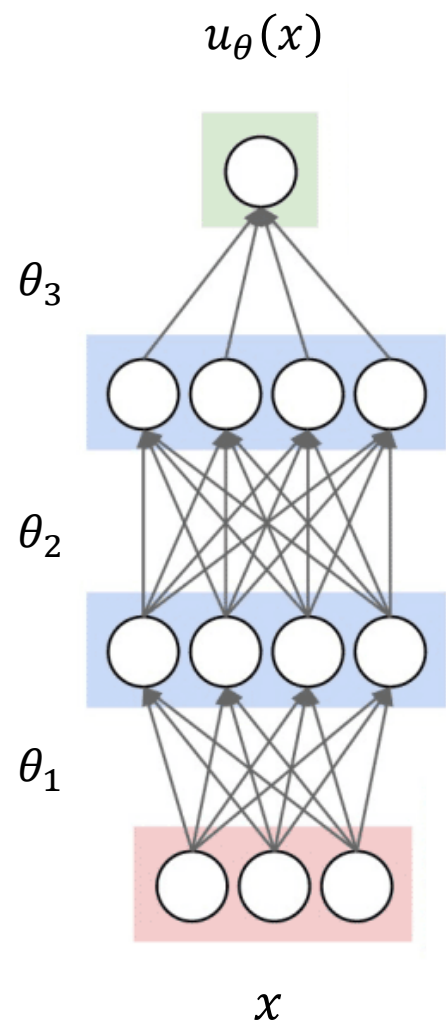
Irregular Boundary



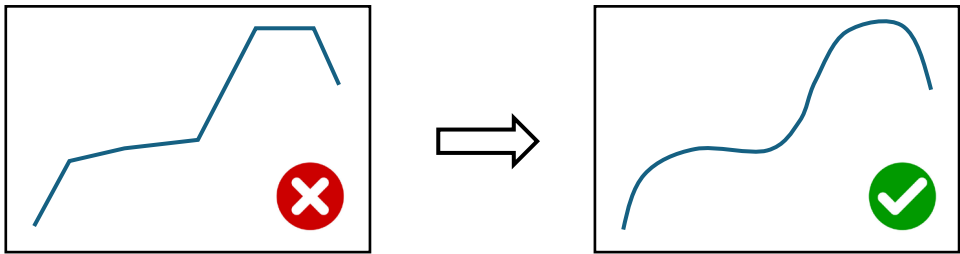
Multiscale



Neural Network Offers New Possibilities



No discretization: continuous solution $u_\theta(x)$



Efficient representation: universal approximation theorem

Universal approximation theorem — Let $C(X, \mathbb{R}^m)$ denote the set of **continuous functions** from a subset X of a Euclidean $\mathbb{R}^n$ space to a Euclidean space $\mathbb{R}^m$. Let $\sigma \in C(\mathbb{R}, \mathbb{R})$. Note that $(\sigma \circ x)_i = \sigma(x_i)$, so $\sigma \circ x$ denotes σ applied to each component of x .

Then σ is not **polynomial if and only if** for every $n \in \mathbb{N}$, $m \in \mathbb{N}$, **compact** $K \subseteq \mathbb{R}^n$, $f \in C(K, \mathbb{R}^m)$, $\varepsilon > 0$ there exist $k \in \mathbb{N}$, $A \in \mathbb{R}^{k \times n}$, $b \in \mathbb{R}^k$, $C \in \mathbb{R}^{m \times k}$ such that

$$\sup_{x \in K} \|f(x) - g(x)\| < \varepsilon$$

where $g(x) = C \cdot (\sigma \circ (A \cdot x + b))$

Automatic differentiation: efficient evaluation of ∇ , ∇^2 , etc.

Forward Laplacian: A New Computational Framework for
Neural Network-based Variational Monte Carlo

Ruichen Li^{1,2§†}, Haotian Ye^{1†}, Du Jiang^{1,2§}, Xuelan Wen², Chuwei Wang¹,
Zhe Li², Xiang Li², Di He^{1,a}, Ji Chen^{1,b}, Weiluo Ren^{2,c}, Liwei Wang^{1,d}

Physics-Informed Neural Network (PINN)

Journal of Computational Physics 378 (2019) 686–707



Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Solve $\mathcal{L}u = 0$ inside interior
under $\mathcal{B}u = 0$ on the boundary

Minimize

$$L = L_{\text{interior}} + L_{\text{boundary}} \\ = \int (\mathcal{L}u)^2 + \int (\mathcal{B}u)^2$$

Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations

M. Raissi^a, P. Perdikaris^{b,*}, G.E. Karniadakis^a

^a Division of Applied Mathematics, Brown University, Providence, RI, 02912, USA

^b Department of Mechanical Engineering and Applied Mechanics, University of Pennsylvania, Philadelphia, PA, 19104, USA

Quantum Dynamics

Solving the quantum many-body problem with artificial neural networks

Giuseppe Carleo^{1*} and Matthias Troyer^{1,2}

Neural-Network Approach to Dissipative Quantum Many-Body Dynamics

Michael J. Hartmann^{1,2} and Giuseppe Carleo³

Quantum Many-Body Dynamics in Two Dimensions with Artificial Neural Networks

Markus Schmitt¹ and Markus Heyl²

Global optimization of quantum dynamics with AlphaZero deep exploration

Mogens Dalggaard¹, Felix Motzoi¹, Jens Jakob Sørensen¹ and Jacob Sherson^{1*}

Autoregressive Neural Network for Simulating Open Quantum Systems via a Probabilistic Formulation

Di Luo,^{1,2,*} Zhuo Chen,^{1,*} Juan Carrasquilla,^{3,4} and Bryan K. Clark^{1,2,5}

Spacetime Neural Network for High Dimensional Quantum Dynamics

Jiangran Wang¹ Zhuo Chen² Di Luo^{2,3} Zhizhen Zhao¹ Vera Mikyoung Hur⁴ Bryan K. Clark^{2,3}

Gauge-invariant and anyonic-symmetric autoregressive neural network for quantum lattice models

Di Luo^{1,2,3,4,*} Zhuo Chen^{1,3,4,*} Kaiwen Hu,^{1,5,6} Zhizhen Zhao,⁷ Vera Mikyoung Hur^{1,6} and Bryan K. Clark^{1,2}

Simulating 2+1D Lattice Quantum Electrodynamics at Finite Density with Neural Flow Wavefunctions

Zhuo Chen,^{1,2,*} Di Luo,^{1,2,3,†} Kaiwen Hu,⁴ and Bryan K. Clark⁵

Neural Operators

FOURIER NEURAL OPERATOR FOR PARAMETRIC PARTIAL DIFFERENTIAL EQUATIONS

Zongyi Li
zongyili@caltech.edu

Nikola Kovachki
nkovachki@caltech.edu

Kamyar Azizzadenesheli
kamyar@purdue.edu

Burigede Liu
bgl@caltech.edu

Kaushik Bhattacharya
bhatta@caltech.edu

Andrew Stuart
astuart@caltech.edu

Anima Anandkumar
anima@caltech.edu

- **Data driven**
- **Reverse engineer/learn PDE operator from data**

Initial Value Problem: Global-In-Time vs Local-In-Time

Setup

$$\text{Solve } \frac{\partial u(x,t)}{\partial t} = \mathcal{L}u(x,t) \text{ with } u(x,0) = u_0(x)$$

Global-in-time:

Solve $u_\theta(x, t)$ **for all** $x \in \mathcal{X} \subseteq \mathbb{R}^d$ **and** $t \in \mathcal{T} \subseteq \mathbb{R}$

Minimize:

$$L = L_{\text{interier}} + L_{\text{initial}}$$

- $L_{\text{interier}} = \int \left(\mathcal{L}u_\theta(x, t) - \frac{\partial u_\theta(x, t)}{\partial t} \right)^2 dx dt$
- $L_{\text{initial}} = \int \left(u_\theta(x, 0) - u_0(x) \right)^2 dx$

Local-in-time:

Evolve $u_{\theta_t}(x)$ **time-step by time-step**

In each time step:

- Solve for $u_{\theta_{t+\Delta t}}$ based on u_{θ_t}

Pro:

- Solution for all t simutanuously
- Similar to traditional machine learning objective

Con:

- Limited representation for both x and t
- Difficulty in optimization

Pro:

- Efficient representation at each t
- Application of high-order time integrators

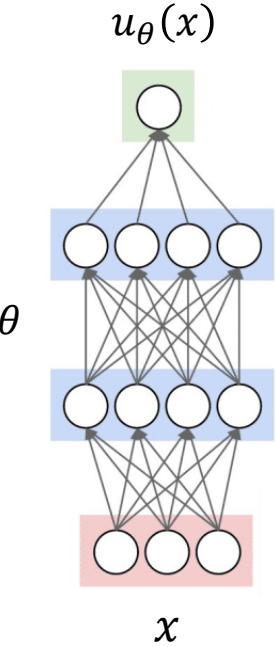
Con:

- Solution only at discrete time steps



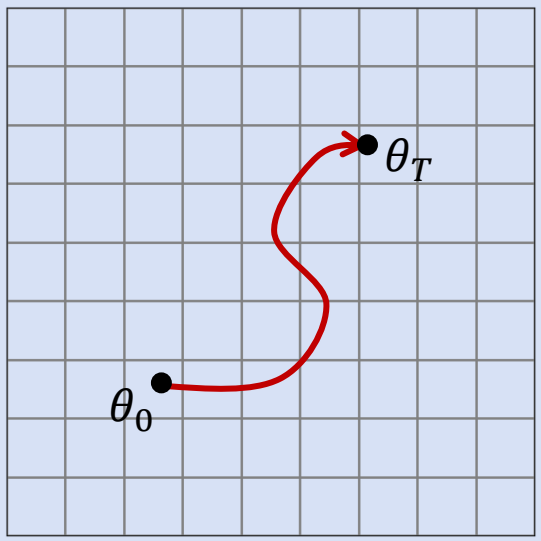
Neural Network Manifold

Neural network parameterizes a manifold in the function space $\mathcal{U} \subset \{u: \mathbb{R}^d \rightarrow \mathbb{R}\}$



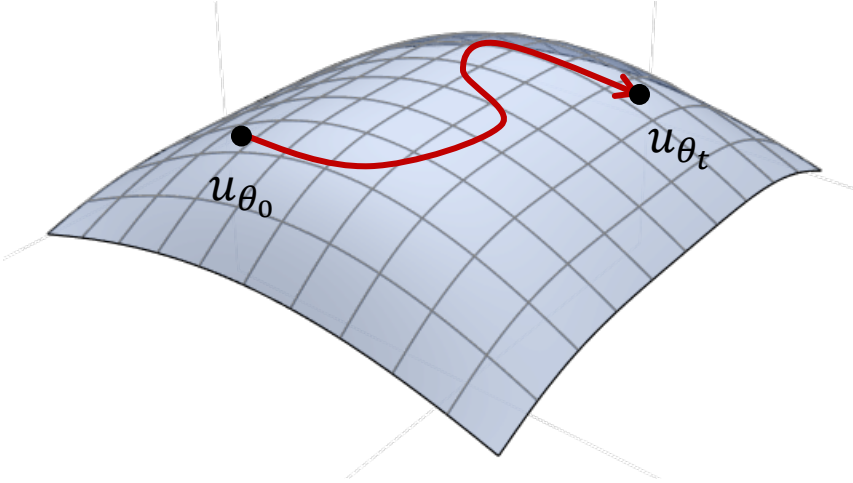
$u_\theta:$

Parameter space: $\Theta \subset \mathbb{R}^p$



Θ

Neural network manifold: $\mathcal{M}_\Theta \subset \mathcal{U}$



$\rightarrow$

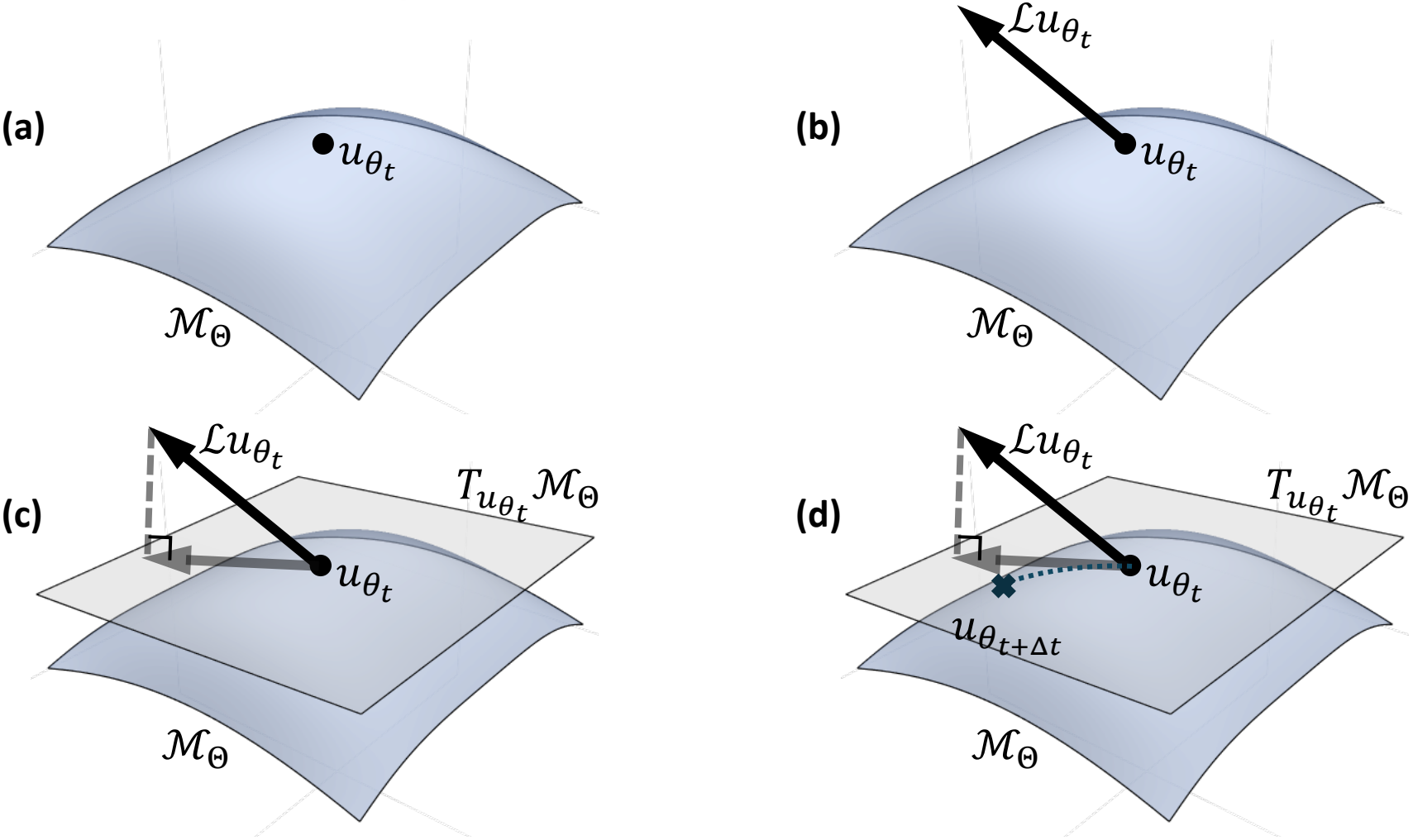
$\mathcal{M}_\Theta$

Time-Dependent Variational Principle (TDVP)

For each time step:

$$\partial_t \theta = \arg \min_{\partial_t \theta \in \mathbb{R}^{N_p}} \left\| \mathcal{L} \hat{u}_\theta(\cdot) - \sum_j J(\cdot, j) \partial_t \theta_j \right\|_{L^2(\mathcal{X})}^2$$

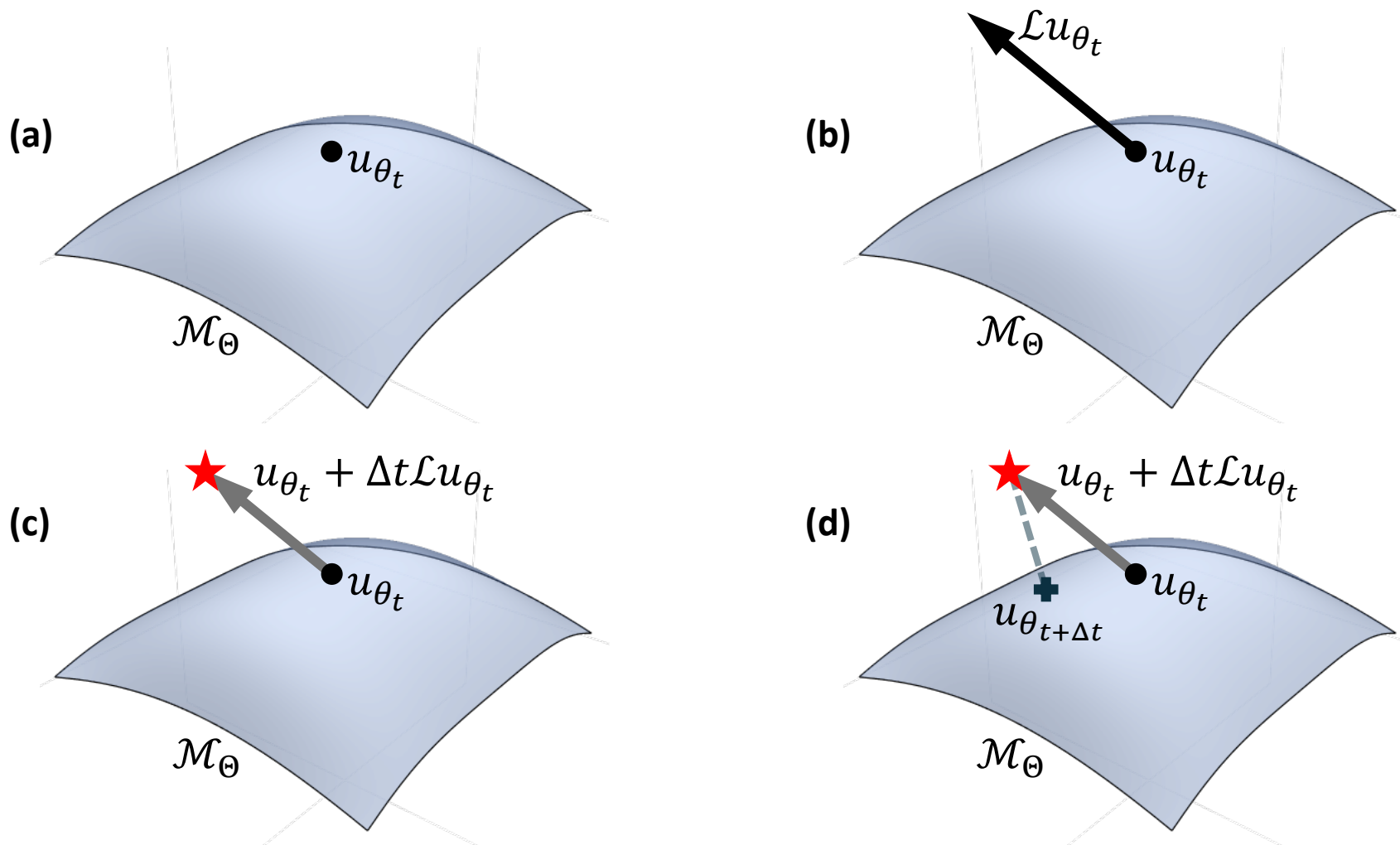
Also called neural Galerkin method



Optimization-Based Time Integration (OBTI)

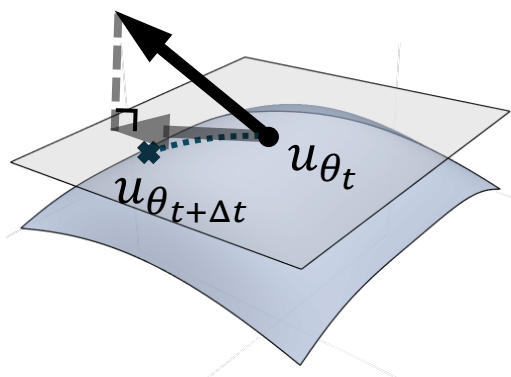
For each time step:

$$\theta_{t+\Delta t} = \arg \min_{\theta \in \Theta} L(\hat{u}_{\theta}, u_{\text{target}})$$



Limitations of TDVP and OBTI

TDVP

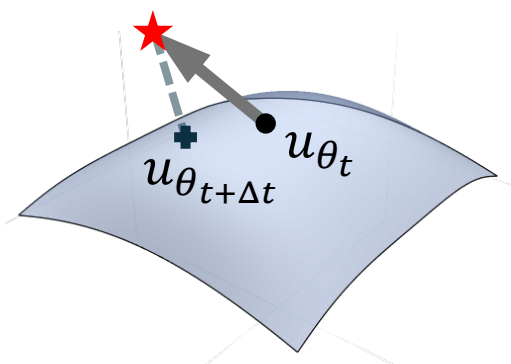


$$\partial_t \theta = \arg \min_{\partial_t \theta \in \mathbb{R}^{N_p}} \left\| \mathcal{L} \hat{u}_\theta(\cdot) - \sum_j J_{(\cdot),j} \partial_t \theta_j \right\|_{L^2(\mathcal{X})}^2$$

Limitations:

- Least square problem usually ill-conditioned/indeterminant
- Regularization affect solution
- Not reparameterization invariant under nonzero Δt

OBTI



$$\theta_{t+\Delta t} = \arg \min_{\theta \in \Theta} L(\hat{u}_\theta, u_{\text{target}})$$

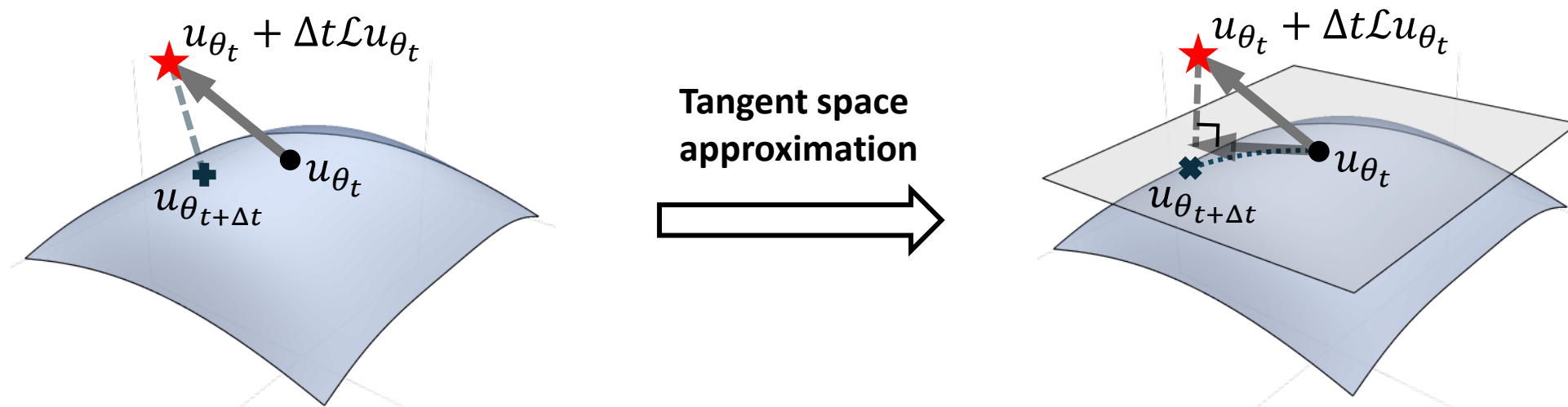
Limitations:

- Optimization is difficult
- No efficient higher-order time integration schemes

How can we resolve the limitations?

Our Method: Time-Evolving Natural Gradient (TENG)

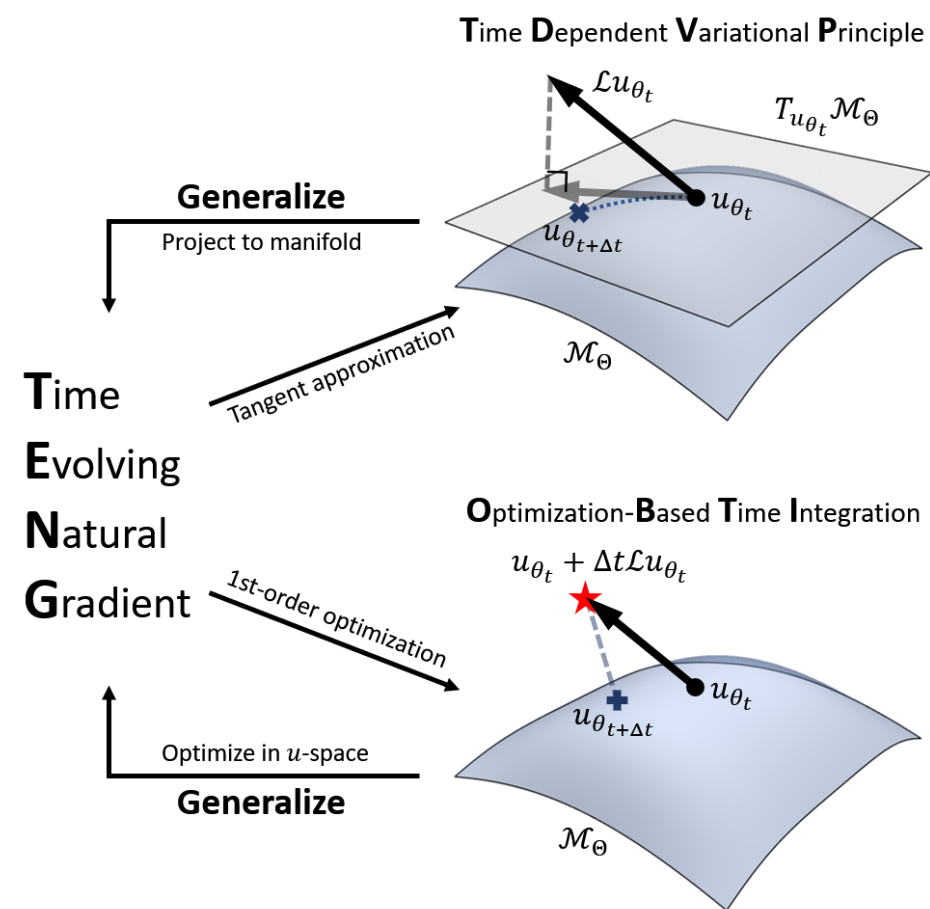
Observation:



TDVP gives the tangent space approximation of the (perfect) OBTI solution

Integrate and generalize TDVP and OBTI!

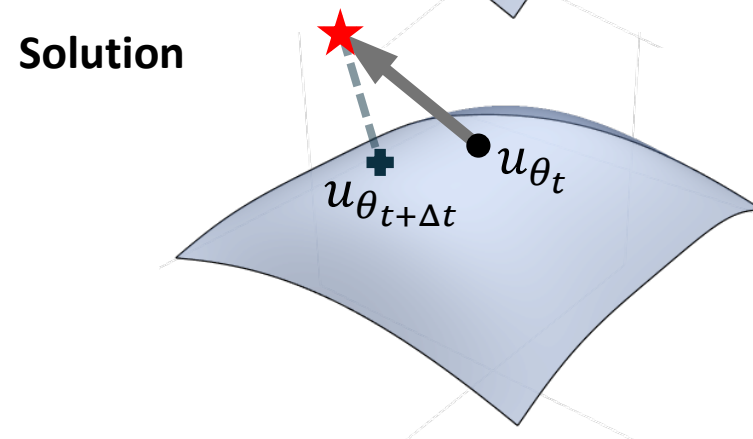
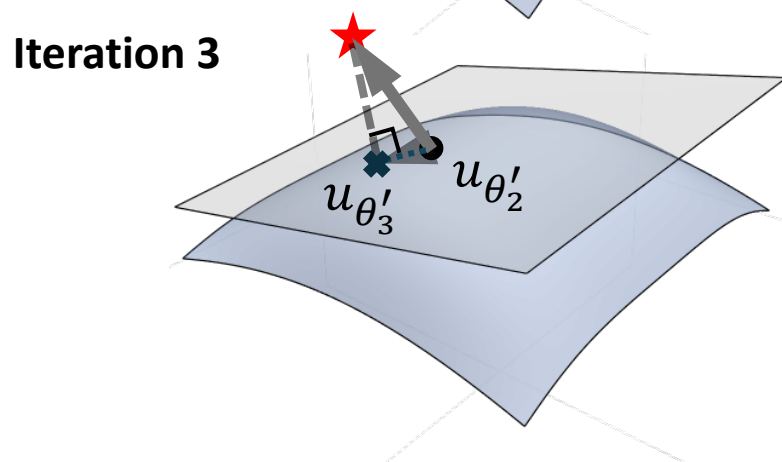
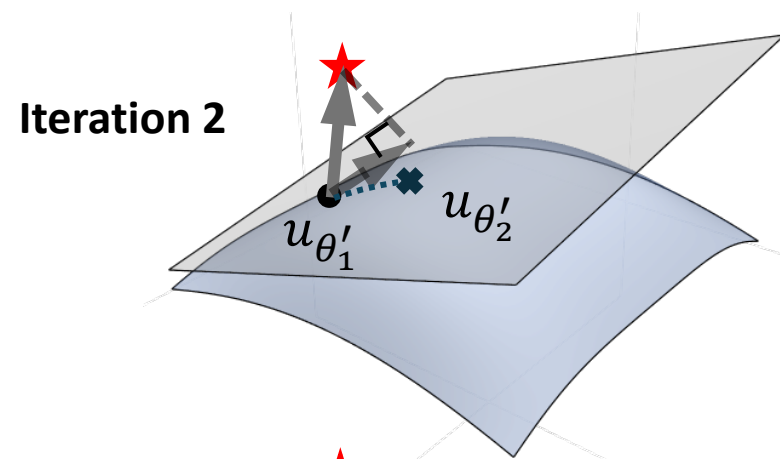
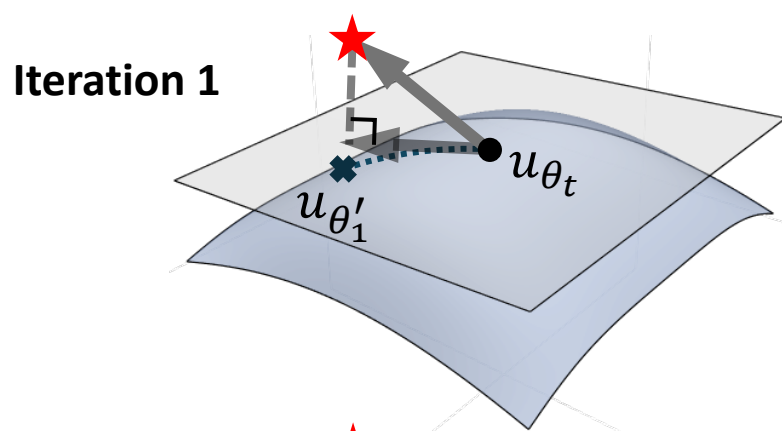
Our Method: Time-Evolving Natural Gradient (TENG)



TENG generalizes TDVP and OBTI, and resolves the limitations

Our Method: Time-Evolving Natural Gradient (TENG)

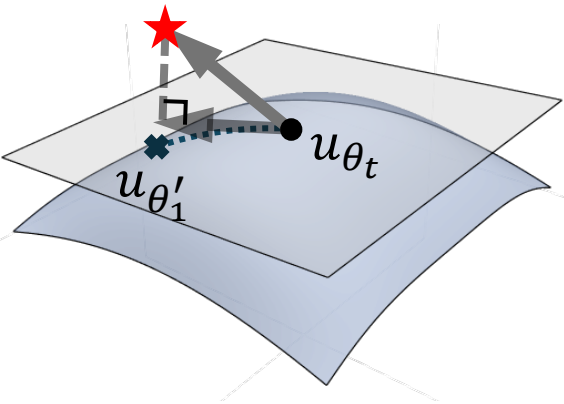
TENG solves $\theta_{t+\Delta t} = \arg \min_{\theta \in \Theta} L(\hat{u}_\theta, u_{\text{target}})$ via a **multi-step** tangent space approximation at each time step



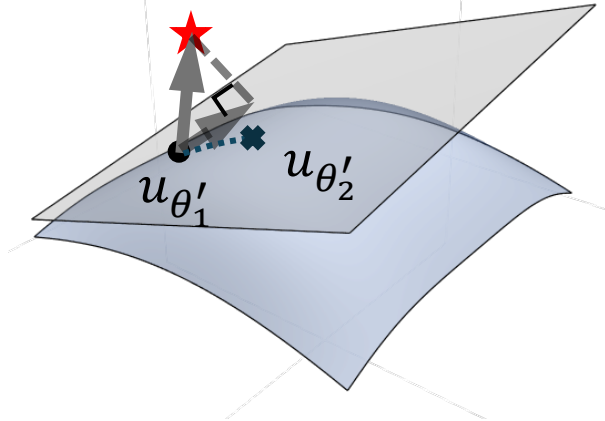
Our Method: Time-Evolving Natural Gradient (TENG)

TENG solves $\theta_{t+\Delta t} = \arg \min_{\theta \in \Theta} L(\hat{u}_\theta, u_{\text{target}})$ via a **multi-step** tangent space approximation at each time step

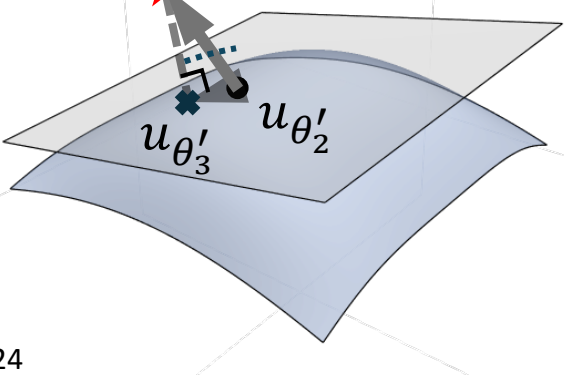
Iteration 1



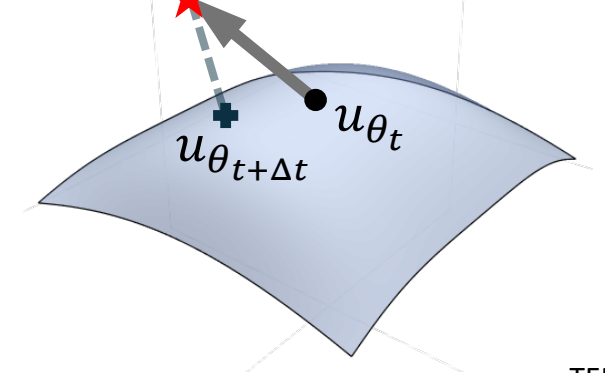
Iteration 2



Iteration 3



Solution



Subroutine TENG_stepper

```
Input:  $\theta_{\text{init}}, u_{\text{target}}$   
 $n \leftarrow 0, \theta \leftarrow \theta_{\text{init}}$   
while  $n < N_{\text{it}}$  do  
   $\Delta u(x) \leftarrow -\alpha_n \frac{\partial L(\hat{u}_\theta, u_{\text{target}})}{\partial \hat{u}_\theta}(x)$   
   $J_{(x),j} \leftarrow \frac{\partial \hat{u}_\theta(x)}{\partial \theta_j}$   
   $\Delta \theta \leftarrow \text{least\_square}(J_{(x),j}, \Delta u(x))$   
   $\theta \leftarrow \theta + \Delta \theta$   
   $n \leftarrow n + 1$   
end while  
Output:  $\theta$ 
```


 $\text{least_square}(J_{(x),j}, \Delta u(x))$
 $:= \Delta \theta = \arg \min_{\Delta \theta \in \mathbb{R}^{N_p}} \left\| \Delta u(\cdot) - \sum_j J_{(\cdot),j} \Delta \theta_j \right\|_{L^2(\mathcal{X})}^2$

Further Enhancing the Efficiency

$$\text{least_square}(J_{(x),j}, \Delta u(x)) = \Delta \theta = \arg \min_{\Delta \theta \in \mathbb{R}^{N_p}} \left\| \Delta u(\cdot) - \sum_j J_{(\cdot),j} \Delta \theta_j \right\|_{L^2(\mathcal{X})}^2$$

Cost of solving the least square solution is

$$C \sim N_s N_p^2$$

..... and it can be low-rank/ill-conditioned

Sparse update!

Randomly subsample parameters for each least square

First proposed under the context of TDVP

Randomized Sparse Neural Galerkin Schemes for Solving Evolution Equations with Deep Networks

Jules Berman Courant Institute for Mathematical Sciences New York University New York, NY 10012 jmb1174@nyu.edu	Benjamin Peherstorfer Courant Institute for Mathematical Sciences New York University New York, NY 10012 pehersto@cims.nyu.edu
--	---

First-Order Euler Time-Integration Schemes

Algorithm TENG_Euler: A 1st-order integration scheme

Input: $\theta_{t=0}, \Delta t, T$

$t \leftarrow 0$

while $t < T$ **do**

$u_{\text{target}}(x) \leftarrow \hat{u}_{\theta_t}(x) + \Delta t \mathcal{L} \hat{u}_{\theta_t}(x)$

$\theta_{t+\Delta t} \leftarrow \text{TENG_stepper}(\theta_t, u_{\text{target}})$

$t \leftarrow t + \Delta t$

end while

Output: $\theta_{t=T}$

Second-Order Time-Integration Schemes

Algorithm TENG_Heun: A 2nd-order integration scheme

Input: $\theta_{t=0}, \Delta t, T$

$t \leftarrow 0$

while $t < T$ **do**

$u_{\text{temp}}(x) \leftarrow \hat{u}_{\theta_t}(x) + \Delta t \mathcal{L} \hat{u}_{\theta_t}(x)$

$\theta_{\text{temp}} \leftarrow \text{TENG_stepper}(\theta_t, u_{\text{temp}})$

$u_{\text{target}}(x) \leftarrow \hat{u}_{\theta_t}(x) + \frac{\Delta t}{2} (\mathcal{L} \hat{u}_{\theta_t}(x) + \mathcal{L} u_{\theta_{\text{temp}}}(x))$

$\theta_{t+\Delta t} \leftarrow \text{TENG_stepper}(\theta_{\text{temp}}, u_{\text{target}})$

$t \leftarrow t + \Delta t$

end while

Output: $\theta_{t=T}$

Connection to Natural Gradient

Hilbert natural gradient formulation:

$$\Delta\theta_i = -\alpha \sum_j G^{-1}(\theta)_{i,j} \frac{\partial L(\hat{u}_\theta, u_{\text{target}})}{\partial \theta_j}(x)$$

Hilbert gram matrix

$$(J^T J)_{i,j} = \int \frac{\partial \hat{u}_\theta(x)}{\partial \theta_i} \frac{\partial \hat{u}_\theta(x)}{\partial \theta_j} dx \equiv G_{i,j}(\theta)$$

Mathematical equivalent to least squares, but numerically less stable

Error Bound

Theorem 4.2. *The error $\varepsilon_p(t)$ is bounded by $\varepsilon_p^{\text{EE}}(t) + \varepsilon_p^{\text{TE}}(t)$, where $\varepsilon_p^{\text{EE}}(t)$ is an order $\mathcal{O}(\Delta t)$ quantity, and $\varepsilon_p^{\text{TE}}(t) = \left\| \sum_{n=0}^{t/\Delta t - 1} \mathcal{G}^n r(\cdot, t - n\Delta t) \right\|_{L^p(\mathcal{X})}$.*

Proof. Denote $D(\cdot, t) = u(\cdot, t) - \hat{u}_\theta(\cdot, t) = (u(\cdot, t) - u^{\text{Eu}}(\cdot, t)) + (u^{\text{Eu}}(\cdot, t) - \hat{u}_\theta(\cdot, t)) \equiv D^{\text{EE}}(\cdot, t) + D^{\text{TE}}(\cdot, t)$, where $u(\cdot, t)$ is the exact solution, $u^{\text{Eu}}(\cdot, t)$ is the solution from Euler method, $\hat{u}_\theta(\cdot, t)$ is the TENG-Euler solution at time t . By definition, $\varepsilon_p(t) = \|D(\cdot, t)\|_{L^p(\mathcal{X})}$, $\|u(\cdot, t) - u^{\text{Eu}}(\cdot, t)\|_{L^p(\mathcal{X})} = \|D^{\text{EE}}(\cdot, t)\|_{L^p(\mathcal{X})} = \varepsilon^{\text{EE}}(t)$, and $\|u^{\text{Eu}}(\cdot, t) - \hat{u}_\theta(\cdot, t)\|_{L^p(\mathcal{X})} = \|D^{\text{TE}}(\cdot, t)\|_{L^p(\mathcal{X})} = \varepsilon^{\text{TE}}(t)$. It follows by the triangular inequality that $\varepsilon_p(t) \leq \varepsilon_p^{\text{EE}}(t) + \varepsilon_p^{\text{TE}}(t)$. Since the Euler method is a first-order method, $\varepsilon_p^{\text{EE}}(t)$ has an error of order $\mathcal{O}(\Delta t)$.

Denote the optimization error in time t as $r(\cdot, t)$, such that $\hat{u}_\theta(\cdot, t + \Delta t) - \mathcal{G}\hat{u}_\theta(\cdot, t) = r(\cdot, t)$. It follows that $u^{\text{Eu}}(\cdot, t + \Delta t) - D^{\text{TE}}(\cdot, t + \Delta t) - \mathcal{G}(u^{\text{Eu}}(\cdot, t) - D^{\text{TE}}(\cdot, t)) = r(\cdot, t)$, which implies that $D^{\text{TE}}(\cdot, t + \Delta t) = \mathcal{G}D^{\text{TE}}(\cdot, t) - r(\cdot, t)$ due to the cancellation of $u^{\text{Eu}}(\cdot, t) - \mathcal{G}u^{\text{Eu}}(\cdot, t)$ from the exact Euler method. By induction, $\varepsilon_p^{\text{TE}}(t) = \left\| \sum_{n=0}^{t/\Delta t - 1} \mathcal{G}^n r(\cdot, t - n\Delta t) \right\|_{L^p(\mathcal{X})}$.

□

Theoretical Results

Properties

- TENG is reparameterization invariant
- TENG allows fast convergence at each time step
- TENG is not affected by the ill-conditionedness of least square problems

Complexity analysis

- TENG: $C_{\text{lstsq}} N_{\text{it}} T / \Delta t$ (both C_{lstsq} and N_{it} can be made small)
- TDVP: $C'_{\text{lstsq}} T / \Delta t$ (large C'_{lstsq} necessary for accurate solution)
- OBTI: $N'_{\text{it}} T / \Delta$ (large N'_{it} necessary for accurate solution)

Experiments: Setup and Baseline

2D and 3D Heat Equation:

$$\frac{\partial u}{\partial t} = \frac{1}{10} \nabla^2 u$$

2D Allen-Cahn Equation:

$$\frac{\partial u}{\partial t} = \frac{1}{200} \nabla^2 u + u - u^3$$

2D Burgers' Equation:

$$\frac{\partial u}{\partial t} = \frac{1}{100} \nabla^2 u - u \left(\frac{\partial u}{\partial x_1} + \frac{\partial u}{\partial x_2} \right)$$

Reference ground truth:

- Spectral method

Baselines:

- TDVP
 - ❖ Sparse update
 - ❖ RK4 solver
- OBTI
 - ❖ Adam/LBFGS solver
- PINN
 - ❖ Adam/ENGd/BFGS solver

Randomized Sparse Neural Galerkin Schemes for
Solving Evolution Equations with Deep Networks

Jules Berman
Courant Institute for Mathematical Sciences
New York University
New York, NY 10012
jmb1174@nyu.edu

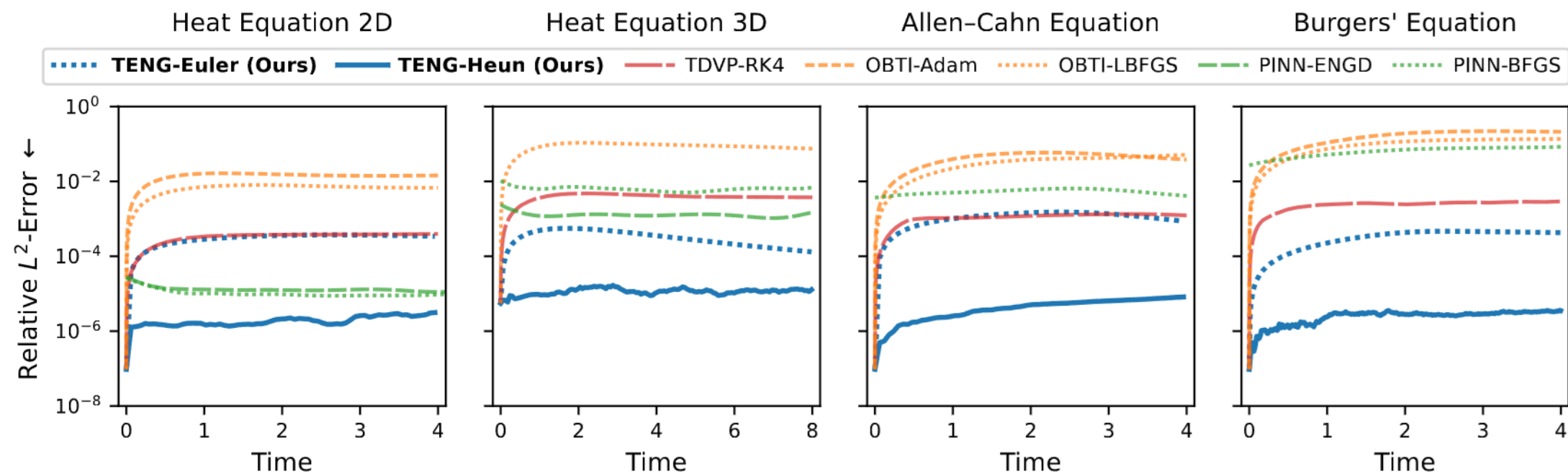
Benjamin Peherstorfer
Courant Institute for Mathematical Sciences
New York University
New York, NY 10012
pehersto@cims.nyu.edu

Achieving High Accuracy with PINNs via Energy Natural Gradient Descent

Johannes Müller¹ Marius Zeinhofer²

Experiments: Overall Results

Relative L^2 -error: $\varepsilon(t) = \frac{\|\hat{u}(\cdot, t) - u_{\text{reference}}(\cdot, t)\|_{L^2(\mathcal{X})}}{\|u_{\text{reference}}(\cdot, t)\|_{L^2(\mathcal{X})}}$



TENG-Euler is better than most algorithms
TENG-Heun achieves the best results in all cases

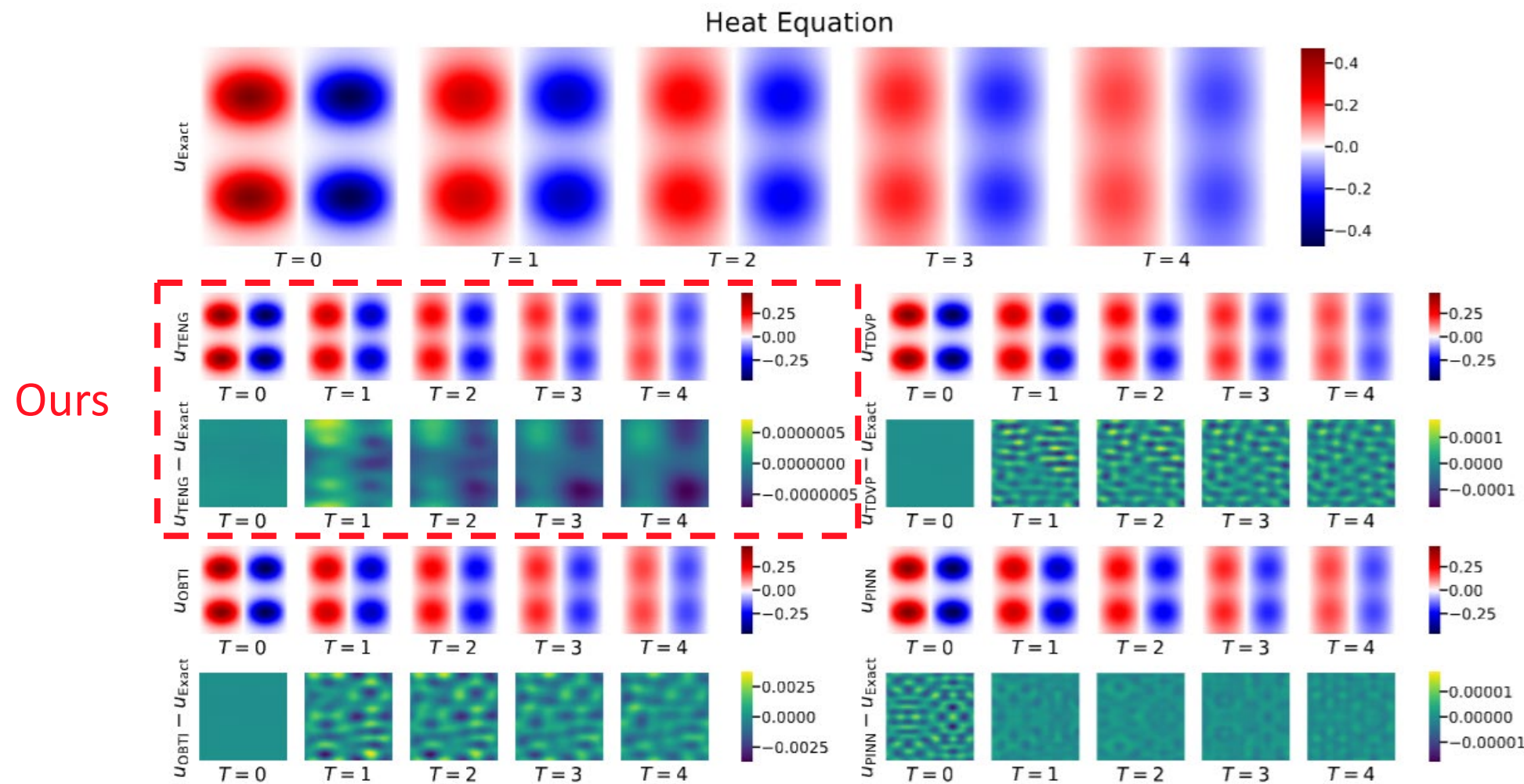
Experiments: Overall Results

Global relative L^2 -error: $\varepsilon_g = \frac{\|\hat{u}(\cdot, \cdot) - u_{\text{reference}}(\cdot, \cdot)\|_{L^2(\mathcal{X} \times \mathcal{T})}}{\|u_{\text{reference}}(\cdot, \cdot)\|_{L^2(\mathcal{X} \times \mathcal{T})}}$

Method	Global Relative L^2 -Error ↓			
	Heat (2D)	Heat (3D)	Allen–Cahn	Burgers’
TENG-Euler (Ours)	3.006×10^{-4}	3.664×10^{-4}	1.249×10^{-3}	3.598×10^{-4}
TENG-Heun (Ours)	1.588×10^{-6}	1.139×10^{-5}	6.187×10^{-6}	2.643×10^{-6}
TDVP-RK4	3.279×10^{-4}	3.841×10^{-3}	1.258×10^{-3}	2.437×10^{-3}
OBTI-Adam	1.391×10^{-2}	–	4.966×10^{-2}	1.696×10^{-1}
OBTI-LBFGS	6.586×10^{-3}	8.743×10^{-2}	4.180×10^{-2}	1.047×10^{-1}
PINN-ENGD	1.403×10^{-5}	2.846×10^{-3}	–	–
PINN-BFGS	1.150×10^{-5}	1.389×10^{-2}	5.540×10^{-3}	6.538×10^{-2}

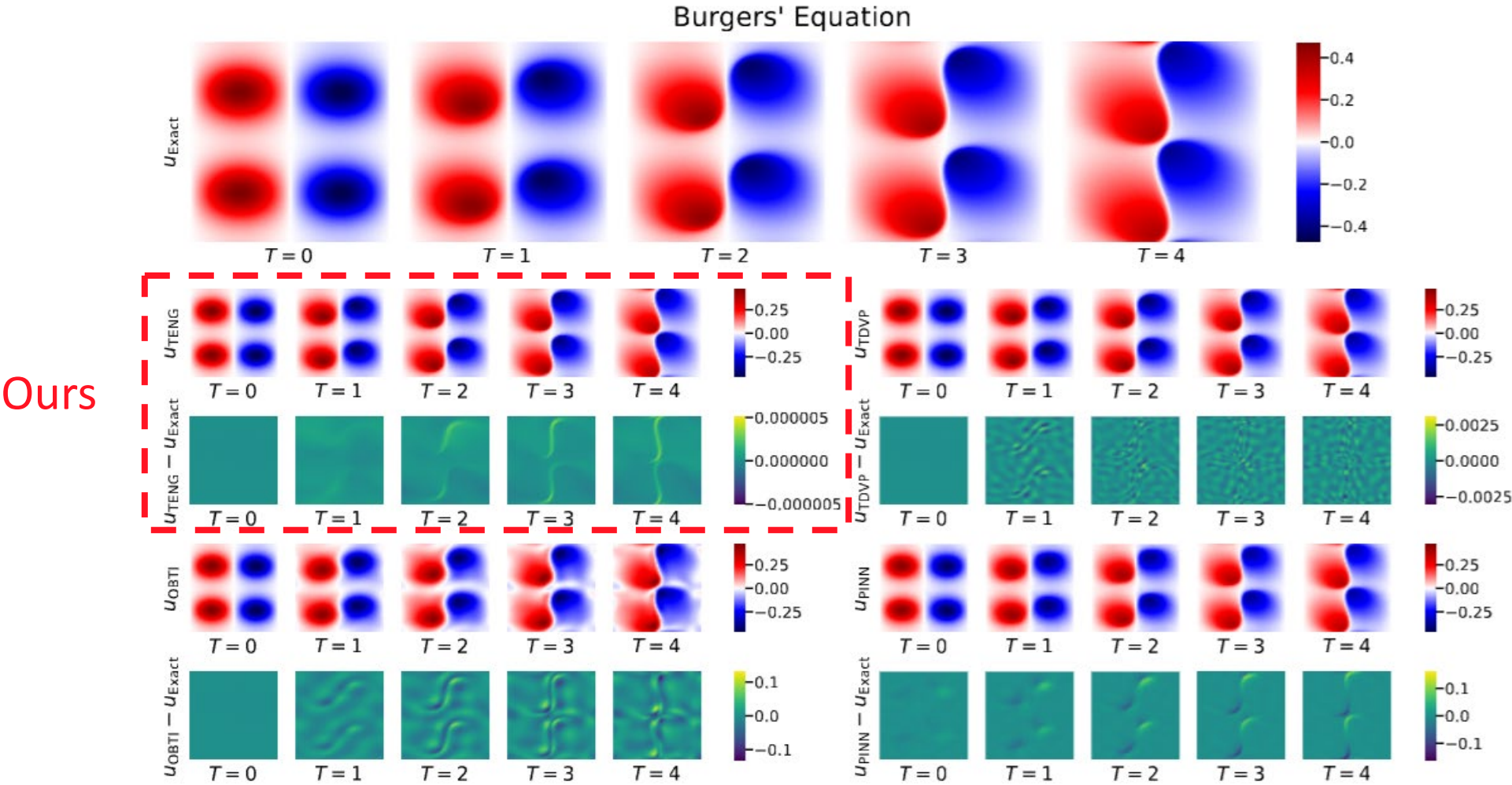
TENG-Euler is better than most algorithms
TENG-Heun achieves the best results in all cases

Experiments: Heat Equation



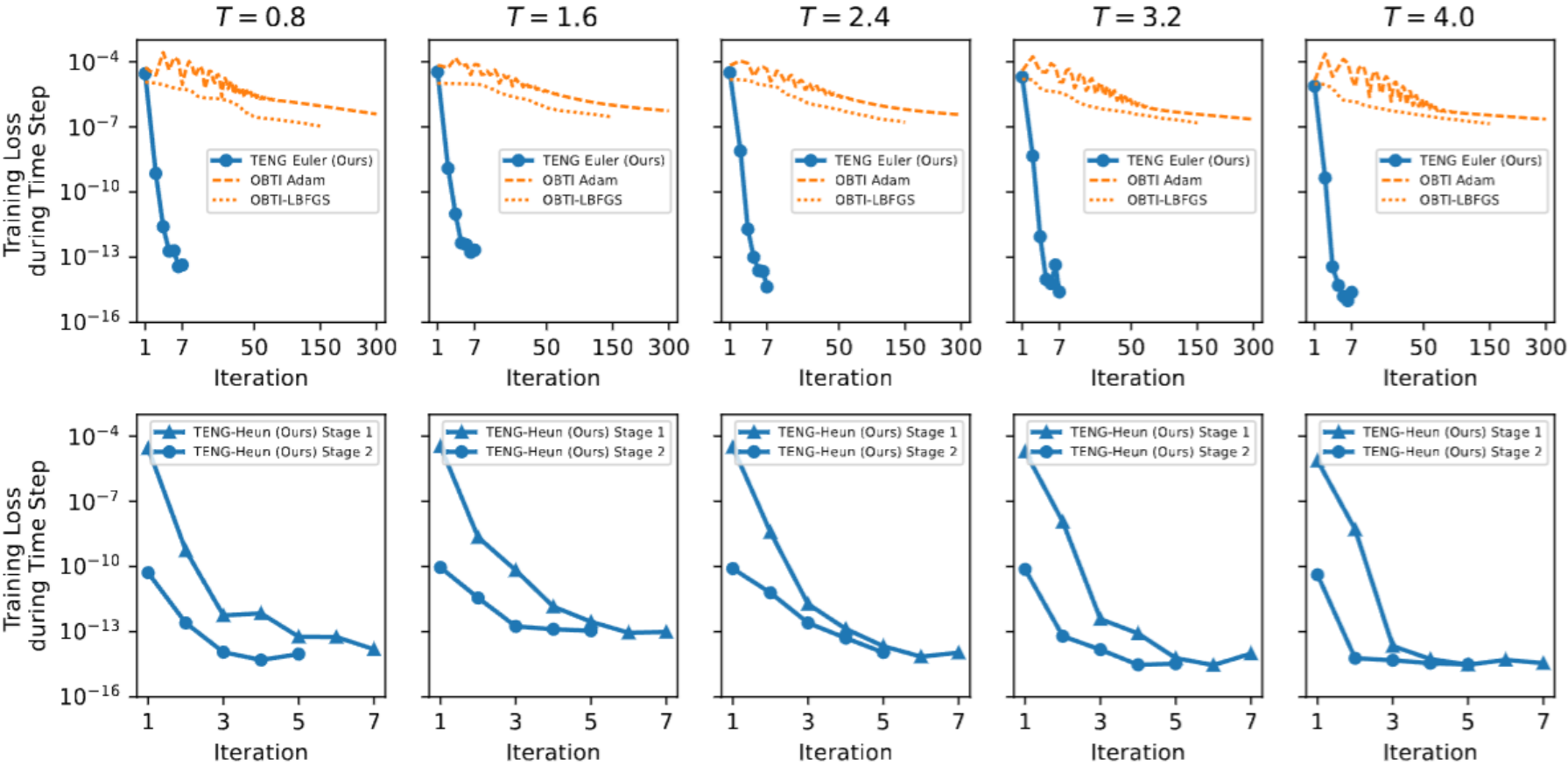
TENG achieves the best results in all cases

Experiments: Burgers' Equation



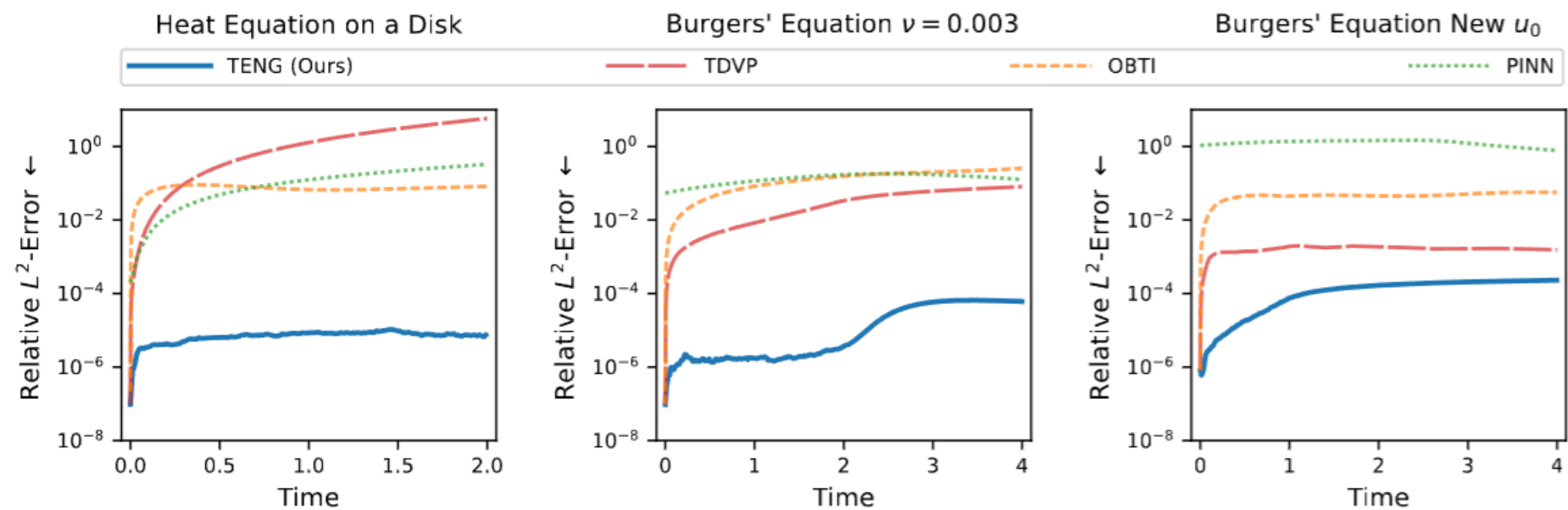
TENG achieves the best results in all cases

Experiments: Loss



Loss close to machine precision

Experiments: More Challenging Examples



TENG remains the best algorithm even at challenging examples

Summary and Outlook

Summary

- We introduce TENG, generalizing TDVP and OBTI
- TENG is a highly accurate and efficient PDE solver
- TENG encompass TENG-Euler and higher-order TENG-Heun, significantly outperforms existing state-of-the-art methods
- TENG achieves loss close to machine precision

Outlook

- Generalize to vector-valued PDE, higher dimensions, and challenging boundary conditions
- Extend TENG to nonlinear and multiscale physics PDEs in other domains
- Integrate TENG with cutting-edge machine learning architectures
- Optimize TENG's performance for large-scale computational tasks