Understanding and mitigating gradient flow pathologies in physics-informed neural networks

Paris Perdikaris
Department of Mechanical Engineering
University of Pennsylvania
email: pgp@seas.upenn.edu

Sifan Wang
Applied Mathematics & Computational Science
University of Pennsylvania
email: sifanw@sas.upenn.edu

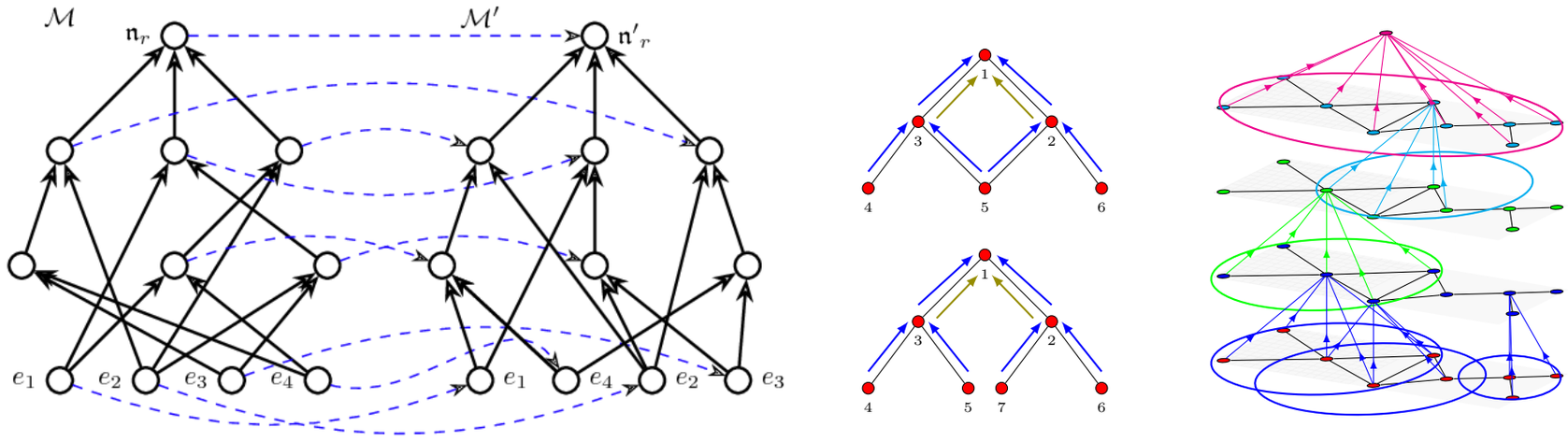
Supported by:

ICERM
Computational Statistics and Data-Driven Models
April 21, 2020

Penn
University of Pennsylvania

Physics of AI: Two schools of thought

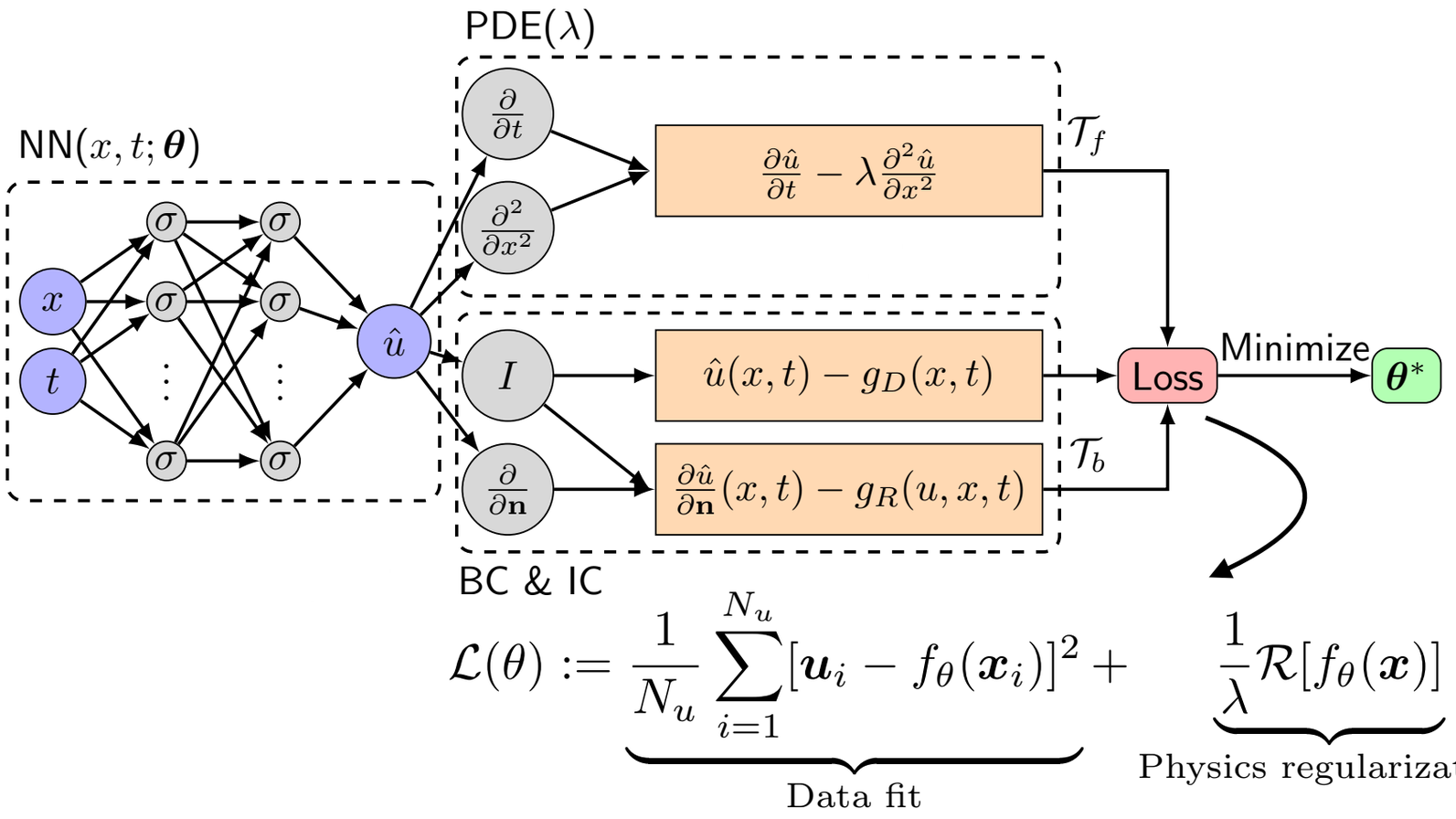
1. Physics is implicitly baked in specialized neural architectures with strong inductive biases (e.g. invariance to simple group symmetries).



**figures from Kondor, R., Son, H.T., Pan, H., Anderson, B., & Trivedi, S. (2018). Covariant compositional networks for learning graphs. arXiv preprint arXiv:1801.02144.*

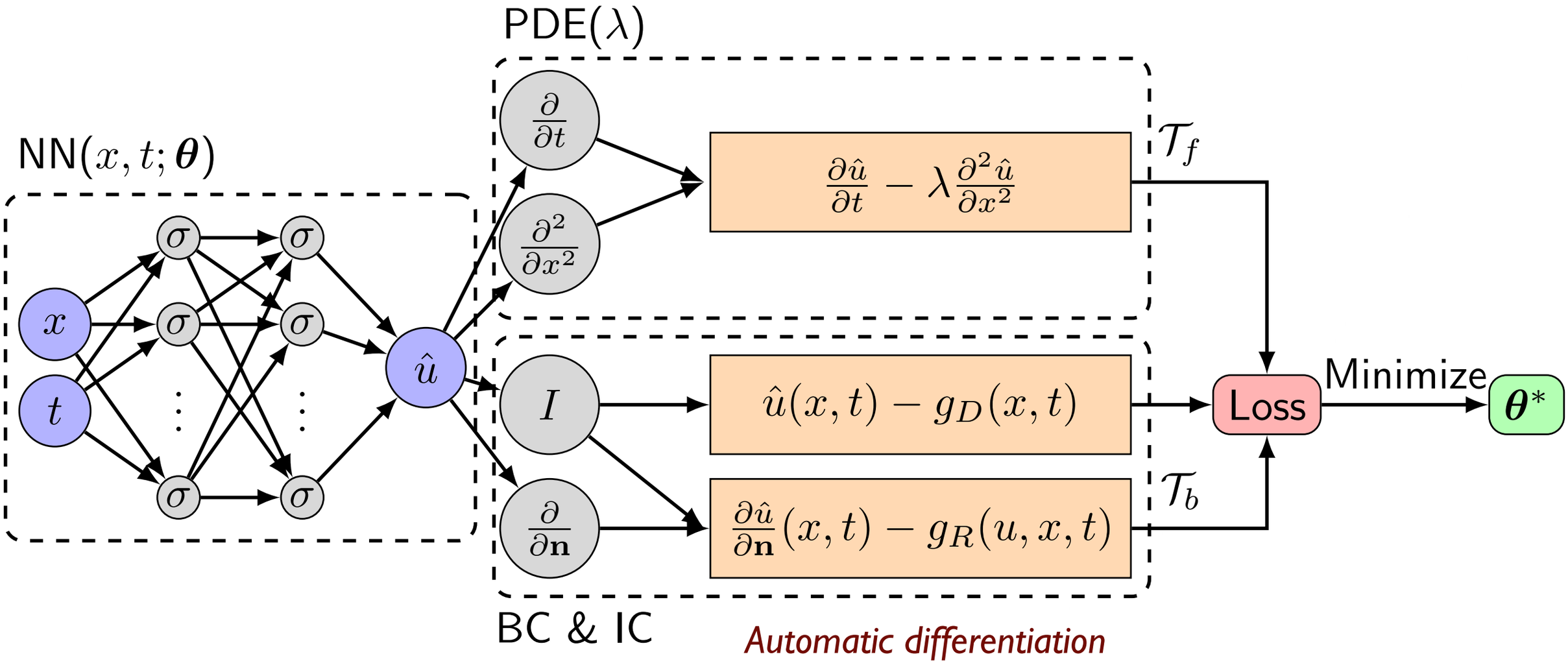
2. Physics is explicitly imposed by constraining the output of conventional neural architectures with weak inductive biases.

Psichogios & Ungar, 1992
Lagaris et. al., 1998
Raissi et. al., 2019
Lu et. al., 2019
Zhu et. al., 2019



Physics-informed Neural Networks

$$f\left(\mathbf{x}; \frac{\partial u}{\partial x_1}, \ldots, \frac{\partial u}{\partial x_d}; \frac{\partial^2 u}{\partial x_1 \partial x_1}, \ldots, \frac{\partial^2 u}{\partial x_1 \partial x_d}; \ldots; \boldsymbol{\lambda}\right) = 0, \quad \mathbf{x} \in \Omega, \qquad \mathcal{B}(u, \mathbf{x}) = 0 \quad \text{on} \quad \partial\Omega,$$



Psichogios, D. C., & Ungar, L. H. (1992). A hybrid neural network–first principles approach to process modeling. AIChE Journal, 38(10), 1499-1511.

Lagaris, I. E., Likas, A., & Fotiadis, D. I. (1998). Artificial neural networks for solving ordinary and partial differential equations. IEEE transactions on neural networks, 9(5), 987-1000.

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics, 378, 686-707.

Lu, L., Meng, X., Mao, Z., & Karniadakis, G. E. (2019). DeepXDE: A deep learning library for solving differential equations. arXiv preprint arXiv: 1907.04502.

General formulation of PINNs

Physics-informed neural networks (PINNs) aim at inferring a continuous latent function $\boldsymbol{u}(\boldsymbol{x}, t)$ that arises as the solution to a system of nonlinear partial differential equations (PDE) of the general form

$$\begin{aligned}
&\boldsymbol{u}_t + \mathcal{N}_{\boldsymbol{x}}[\boldsymbol{u}] = 0, \quad \boldsymbol{x} \in \Omega, t \in [0, T] \\
&\boldsymbol{u}(\boldsymbol{x}, 0) = h(\boldsymbol{x}), \quad x \in \Omega \\
&\boldsymbol{u}(\boldsymbol{x}, t) = g(\boldsymbol{x}, t), \quad t \in [0, T], \quad \boldsymbol{x} \in \partial\Omega
\end{aligned}$$

We proceed by approximating $\boldsymbol{u}(\boldsymbol{x}, t)$ by a deep neural network $f_\theta(\boldsymbol{x}, t)$, and define the residual of the PDE as

$$\boldsymbol{r}_\theta(\boldsymbol{x}, t) := \frac{\partial}{\partial t} f_\theta(\boldsymbol{x}, t) + \mathcal{N}_{\boldsymbol{x}}[f_\theta(\boldsymbol{x}, t)]$$

The corresponding loss function is given by

$$\mathcal{L}(\theta) := \underbrace{\mathcal{L}_u(\theta)}_{\text{Data fit}} + \underbrace{\mathcal{L}_r(\theta)}_{\text{PDE residual}} + \underbrace{\mathcal{L}_{u_0}(\theta)}_{\text{ICs fit}} + \underbrace{\mathcal{L}_{u_b}(\theta)}_{\text{BCs fit}}$$

Training via stochastic gradient descent:

$$\theta_{n+1} = \theta_n - \eta \nabla_\theta \mathcal{L}(\theta_n)$$

**all gradients are computed via automatic differentiation*

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networks:A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics, 378, 686-707.

Physics-informed Neural Networks

Example: Burgers' equation in 1D

$$
\begin{aligned}
&u_t + uu_x - (0.01/\pi)u_{xx} = 0, \quad x \in [-1, 1], \quad t \in [0, 1], \qquad (3)\\
&u(0, x) = -\sin(\pi x),\\
&u(t, -1) = u(t, 1) = 0.
\end{aligned}
$$

Let us define $f(t, x)$ to be given by

$$f := u_t + uu_x - (0.01/\pi)u_{xx},$$

```
def u(t, x):
    u = neural_net(tf.concat([t,x],1), weights, biases)
    return u
```

Correspondingly, the *physics informed neural network* $f(t, x)$ takes the form

```
def f(t, x):
    u = u(t, x)
    u_t = tf.gradients(u, t)[0]
    u_x = tf.gradients(u, x)[0]
    u_xx = tf.gradients(u_x, x)[0]
    f = u_t + u*u_x - (0.01/tf.pi)*u_xx
    return f
```

Physics-informed Neural Networks

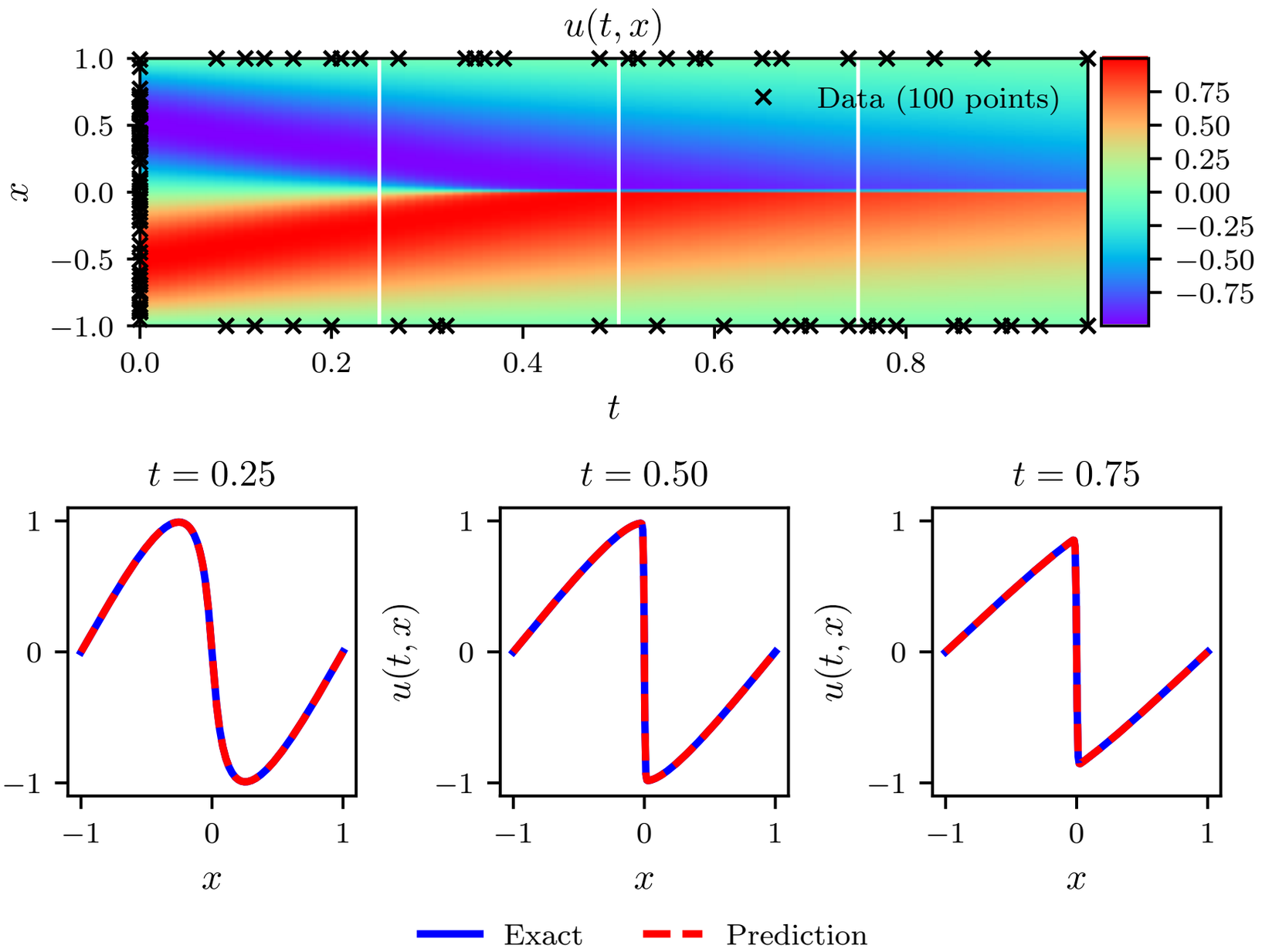
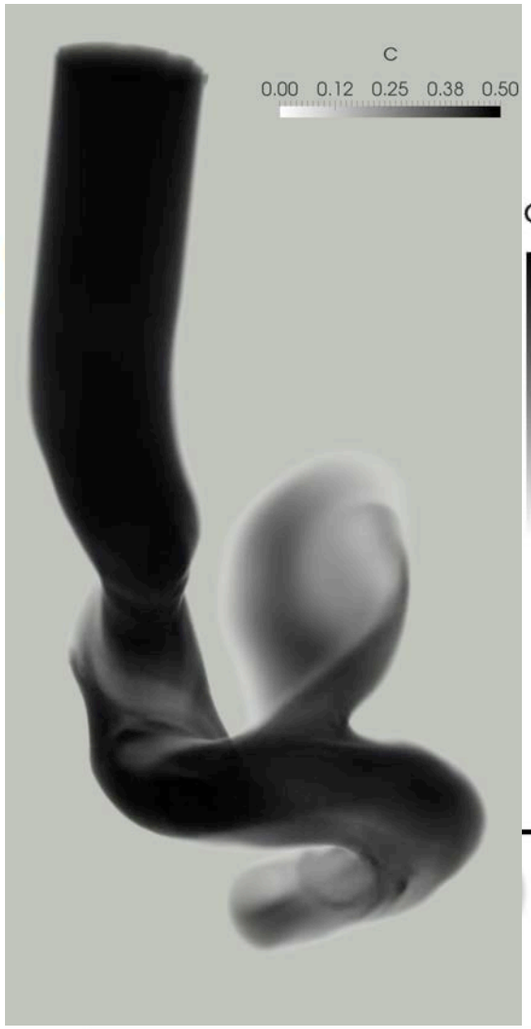
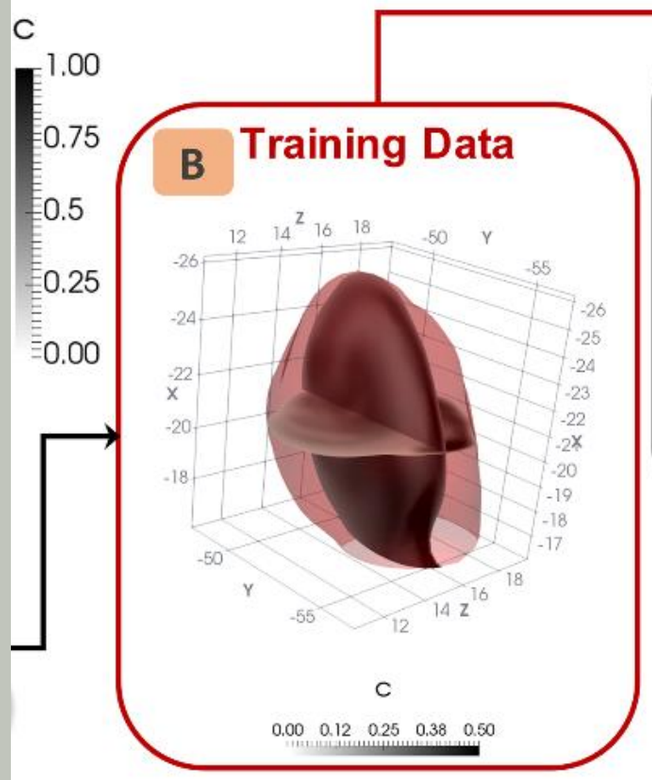
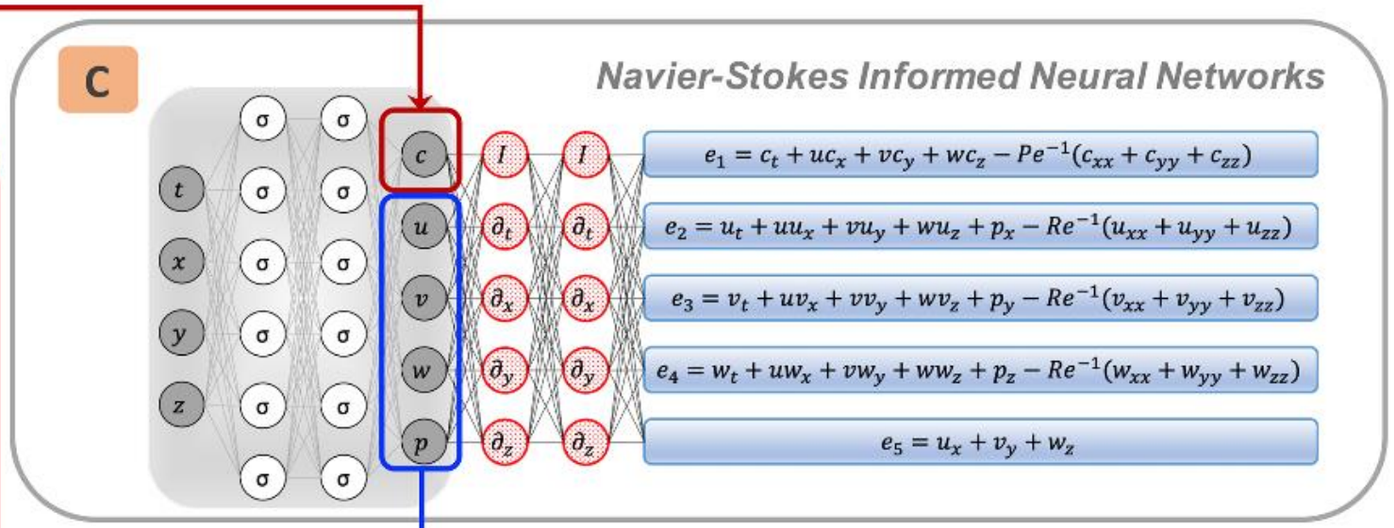


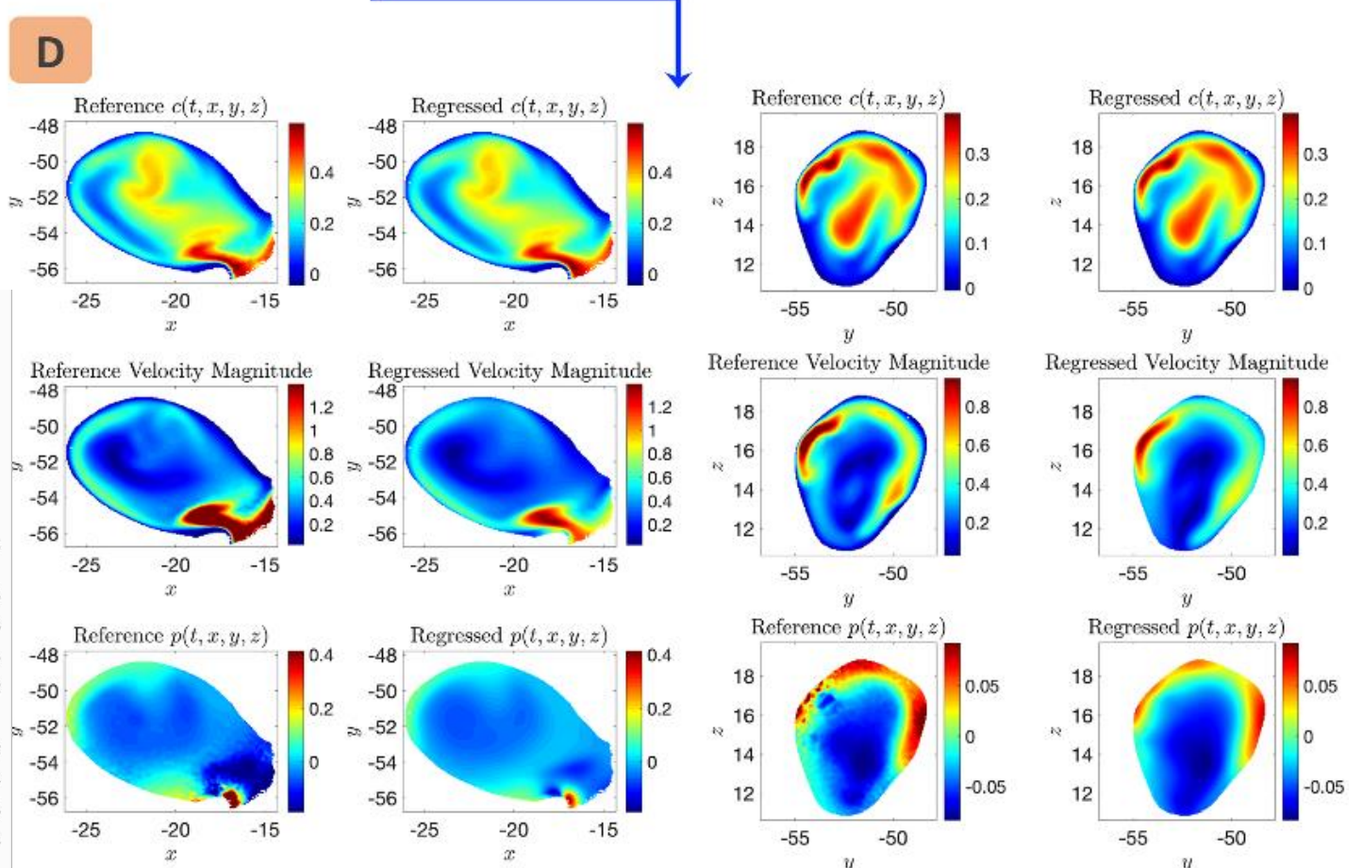
Figure 1: *Burgers' equation: Top:* Predicted solution $u(t,x)$ along with the initial and boundary training data. In addition we are using 10,000 collocation points generated using a Latin Hypercube Sampling strategy. *Bottom:* Comparison of the predicted and exact solutions corresponding to the three temporal snapshots depicted by the white vertical lines in the top panel. The relative $\mathcal{L}_2$ error for this case is $6.7 \cdot 10^{-4}$. Model training took approximately 60 seconds on a single NVIDIA Titan X GPU card.

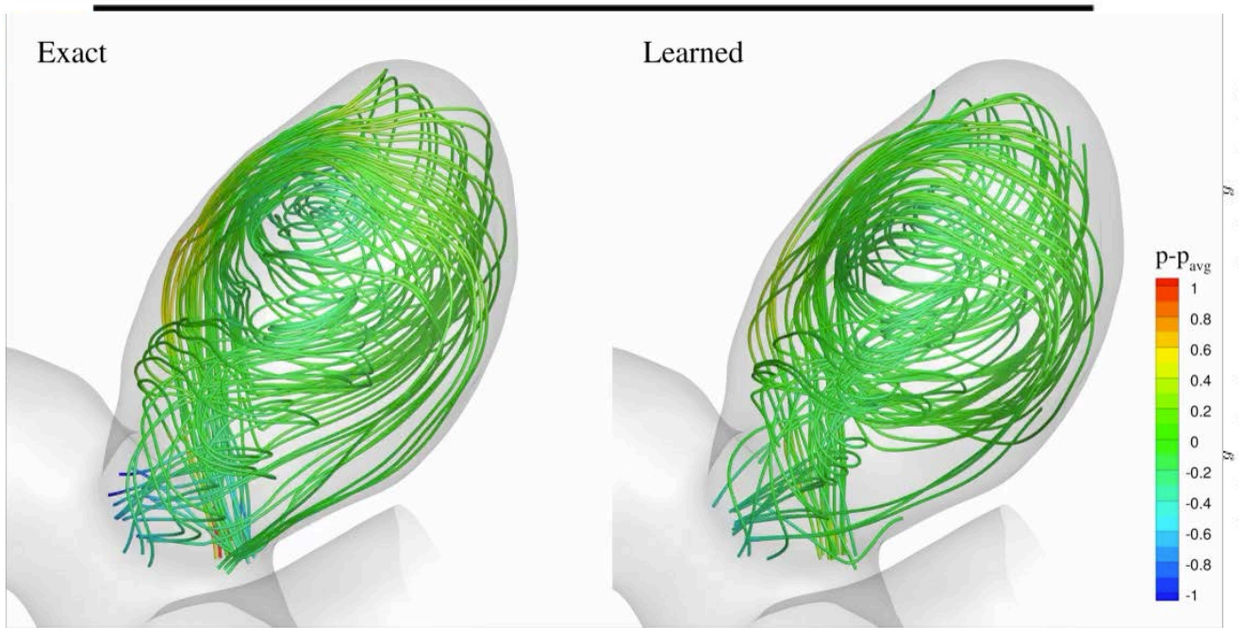
Physics-informed neural networks







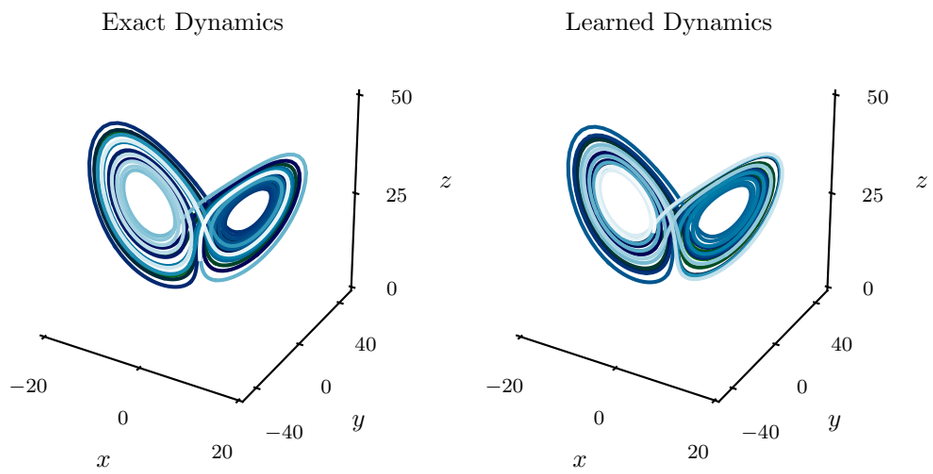




Raissi, M., Yazdani, A., & Karniadakis, G. E. (2020). Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. Science.

Recent advances

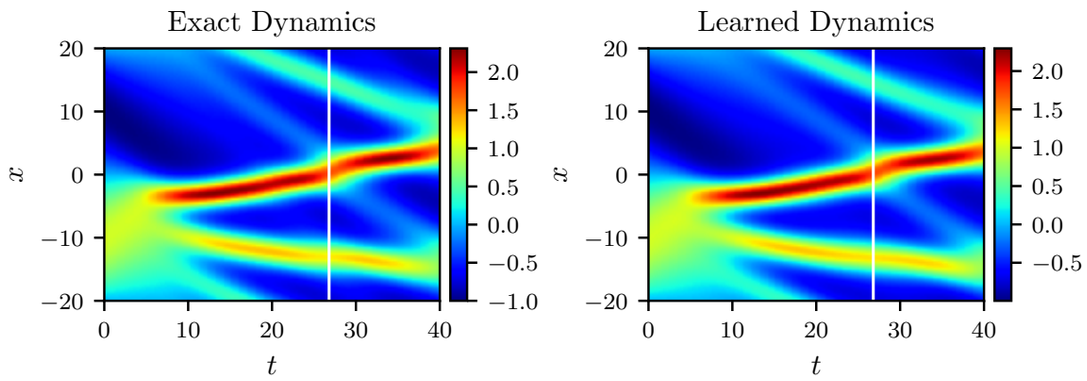
Discovery of ODEs



Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2018). Multistep Neural Networks for Data-driven Discovery of Nonlinear Dynamical Systems. arXiv preprint

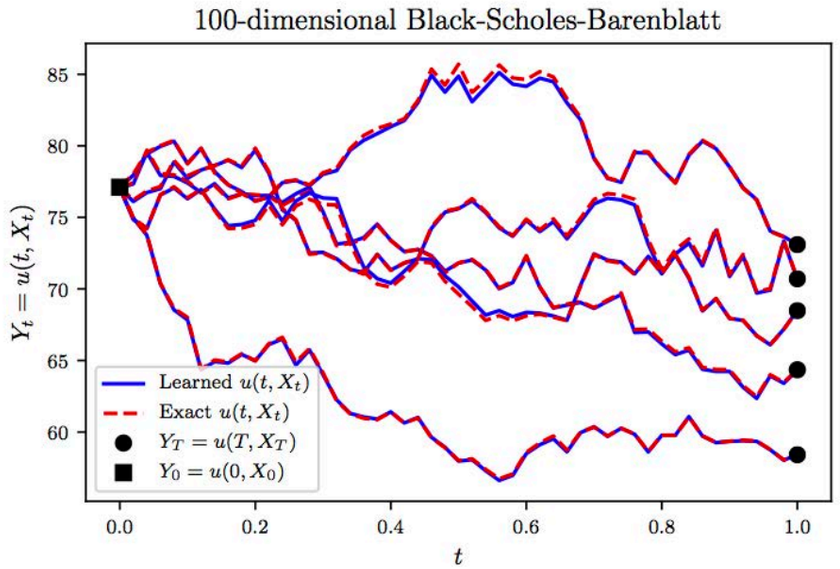
Discovery of PDEs

$$u_t = -uu_x - u_{xxx}$$



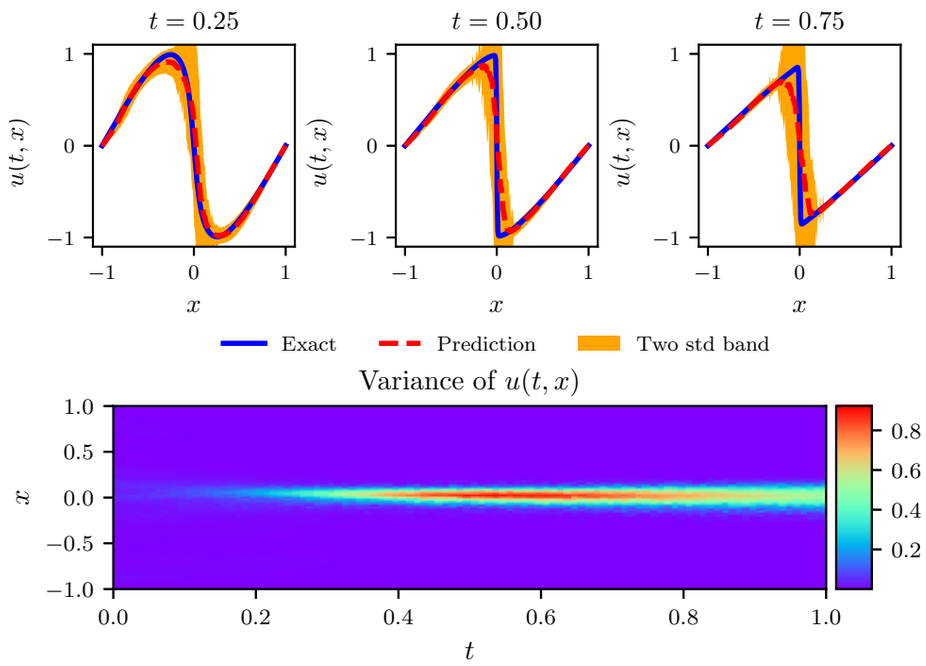
Raissi, M. (2018). Deep Hidden Physics Models: Deep Learning of Nonlinear Partial Differential Equations. arXiv preprint arXiv:1801.06637.

High-dimensional PDEs



Raissi, M. (2018). Forward-backward stochastic neural networks: Deep learning of high-dimensional partial differential equations. arXiv preprint arXiv: 1804.07010.

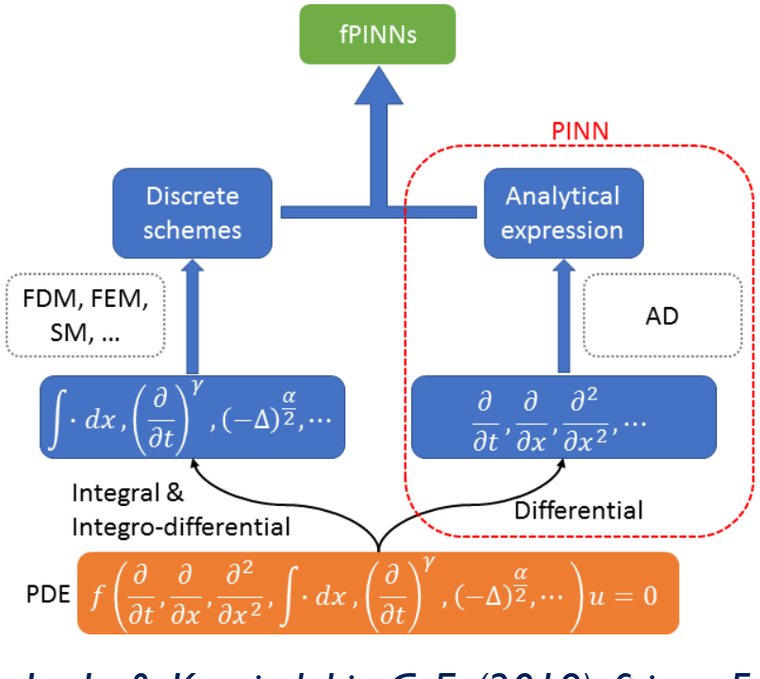
Stochastic PDEs



Yang, Y., & Perdikaris, P. (2019). Adversarial uncertainty quantification in physics-informed neural networks. Journal of Computational Physics.

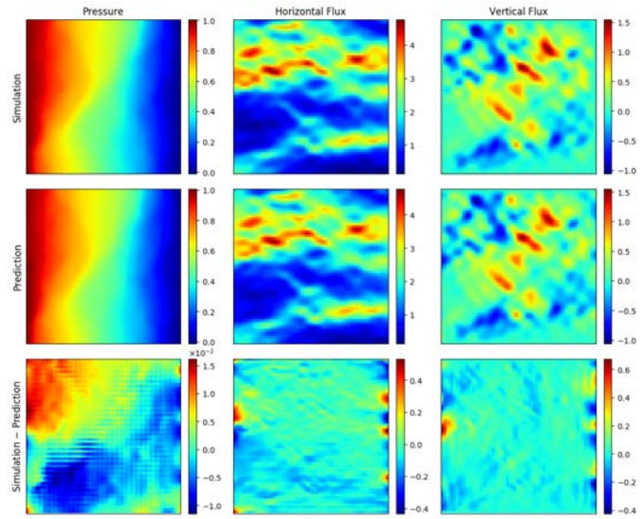
Recent advances

Fractional PDEs



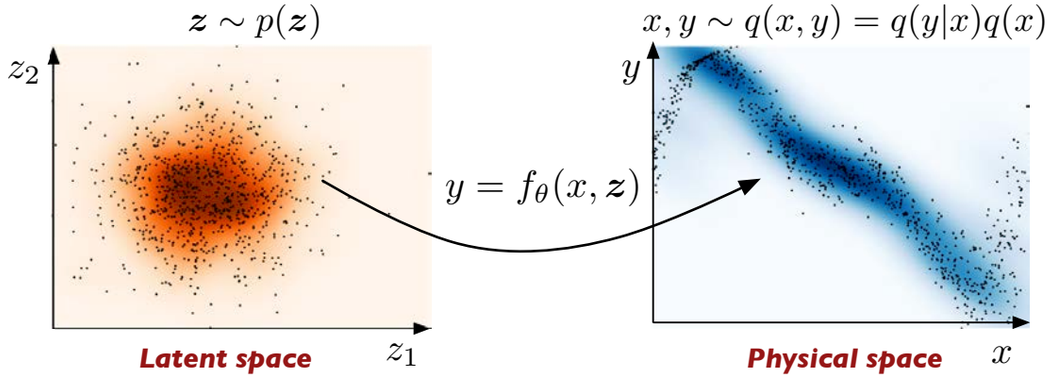
Pang, G., Lu, L., & Karniadakis, G. E. (2018). fpinns: Fractional physics-informed neural networks. arXiv preprint arXiv: 1811.08967.

Surrogate modeling & high-dimensional UQ



Zhu,Y., Zabaras, N., Koutsourelakis, P. S., & Perdikaris, P. (2019). Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. Journal of Computational Physics, 394, 56-81.

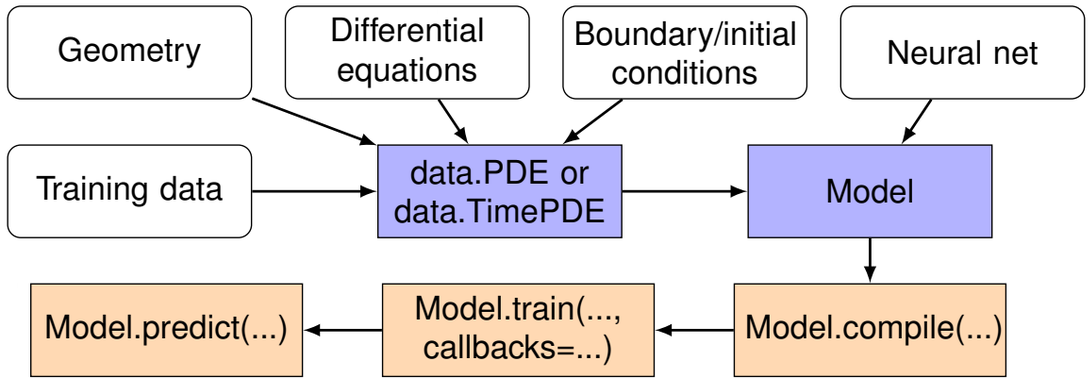
Multi-fidelity modeling for stochastic systems



$$y = f_\theta(x, \boldsymbol{z}),\ \boldsymbol{z} \sim p(\boldsymbol{z}) \ \Leftrightarrow\ y \sim p_\theta(y|x, \boldsymbol{z})$$

Conditional deep surrogate models for stochastic, high-dimensional, and multi-fidelity systems. Computational Mechanics, 1-18.

Integrated software



Lu, L., Meng, X., Mao, Z., & Karniadakis, G. E. (2019). DeepXDE: A deep learning library for solving differential equations. arXiv preprint arXiv:1907.04502.

Physics a prior in deep learning

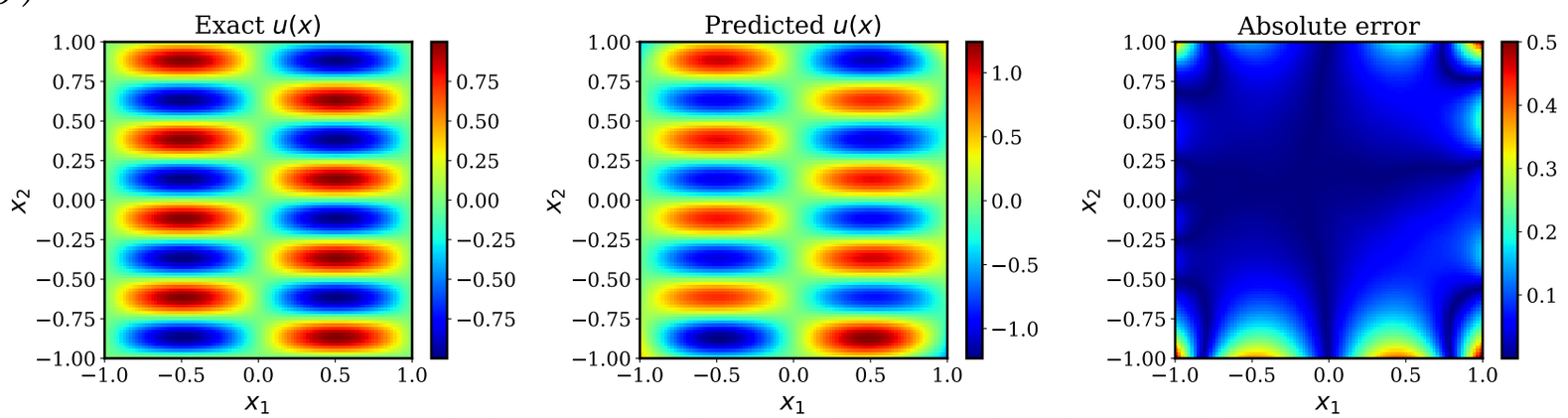
$$\mathcal{L}(\theta) := \underbrace{\frac{1}{N_u}\sum_{i=1}^{N_u}[\boldsymbol{u}_i - f_\theta(\boldsymbol{x}_i)]^2}_{\text{Data fit}} + \underbrace{\frac{1}{\lambda}\mathcal{R}[f_\theta(\boldsymbol{x})]}_{\text{Physics regularization}}$$

Recent results showcase remarkable promise, but failure looms even for the simplest problems…

Example: Helmholtz equation in 2D.

$$\Delta u(x,y) + k^2 u(x,y) = q(x,y), \quad (x,y) \in \Omega := (-1,1)$$
$$u(x,y) = h(x,y), \quad (x,y) \in \partial\Omega$$

where Δ is the Laplace operator.



20% error in the prediction of a dense, 4-layer deep PINN model

An "unconventional" regularizer/prior that requires us to revisit standard deep learning practices:

- loss function
- network initialization
- data normalization
- optimization
- network architecture

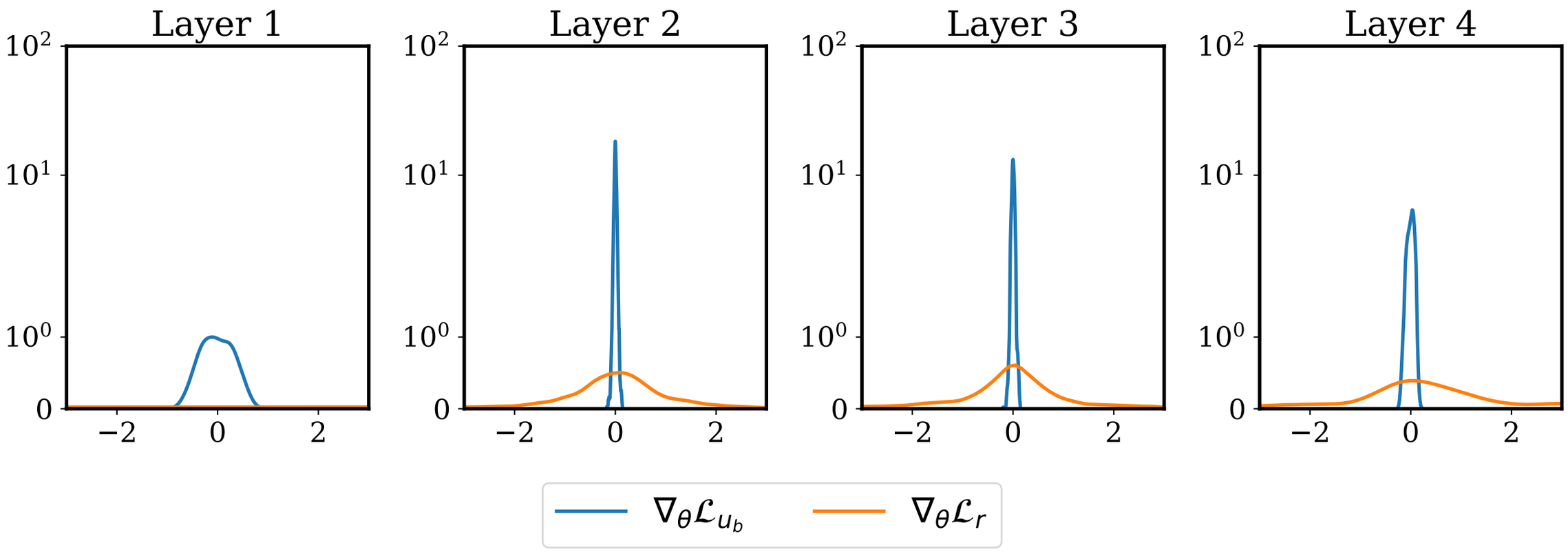
→ This talk { Overcoming gradient pathologies in PINNs via
- Adaptive learning rate strategies
- Improved neural architectures

Gradient pathologies in PINNs

Example: Helmholtz equation in 2D. The loss function is given by

$$\mathcal{L}(\theta) = \mathcal{L}_r(\theta) + \mathcal{L}_{u_b}(\theta)$$

Hypothesis: Different terms in the loss function may have different nature and magnitudes, leading to imbalanced gradients during back-propagation.



Histograms of back-propagated gradients $\nabla_\theta \mathcal{L}_r(\theta)$ and $\nabla_\theta \mathcal{L}_{u_b}(\theta)$ at each hidden layer.

Gradient analysis for PINNs

Example: Poisson equation in 1D

$$\Delta u(x) = g(x), \quad x \in [0,1]$$
$$u(x) = h(x), \quad x = 0 \text{ and } x = 1.$$

Let us consider exact solutions of the form $u(x) = \sin(Cx)$. Then we can use a deep neural network $f_\theta(x)$ to approximating $u(x)$.
The loss function is given by

$$\begin{aligned}\mathcal{L}(\theta) &= \mathcal{L}_r(\theta) + \mathcal{L}_{u_b}(\theta) \\ &= \frac{1}{N_b}\sum_{i=1}^{N_b}[f_\theta(x_b^i) - h(x_b^i)]^2 + \frac{1}{N_r}\sum_{i=1}^{N_r}[\frac{\partial^2}{\partial x^2} f_\theta(x_r^i) - g(x_r^i)]^2.\end{aligned}$$

Assumptions:

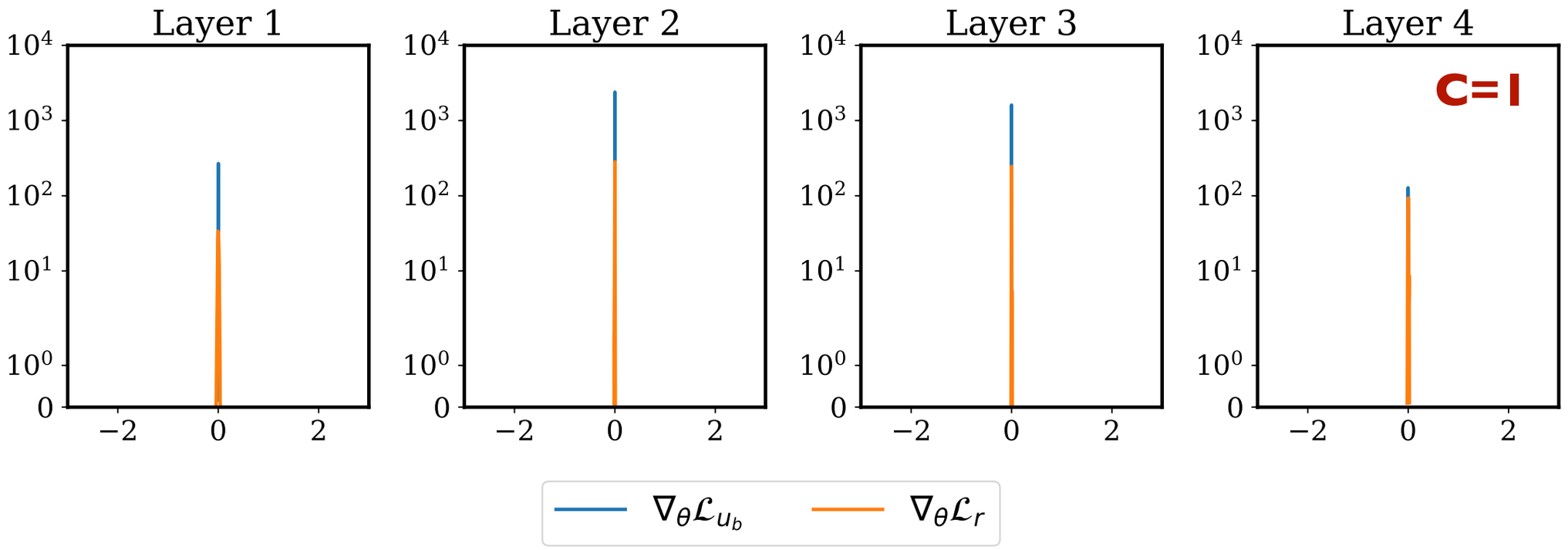
- $f_\theta(x) = u(x)\epsilon_\theta(x)$, where $\epsilon_\theta(x)$ is a smooth function defined in $[0,1]$.
- There exists $\epsilon > 0$ such that $|\epsilon_\theta(x) - 1| \leq \epsilon$ and $\|\frac{\partial^k \epsilon_\theta(x)}{\partial x^k}\|_{L^\infty} < \epsilon$, for all non-negative integer k.

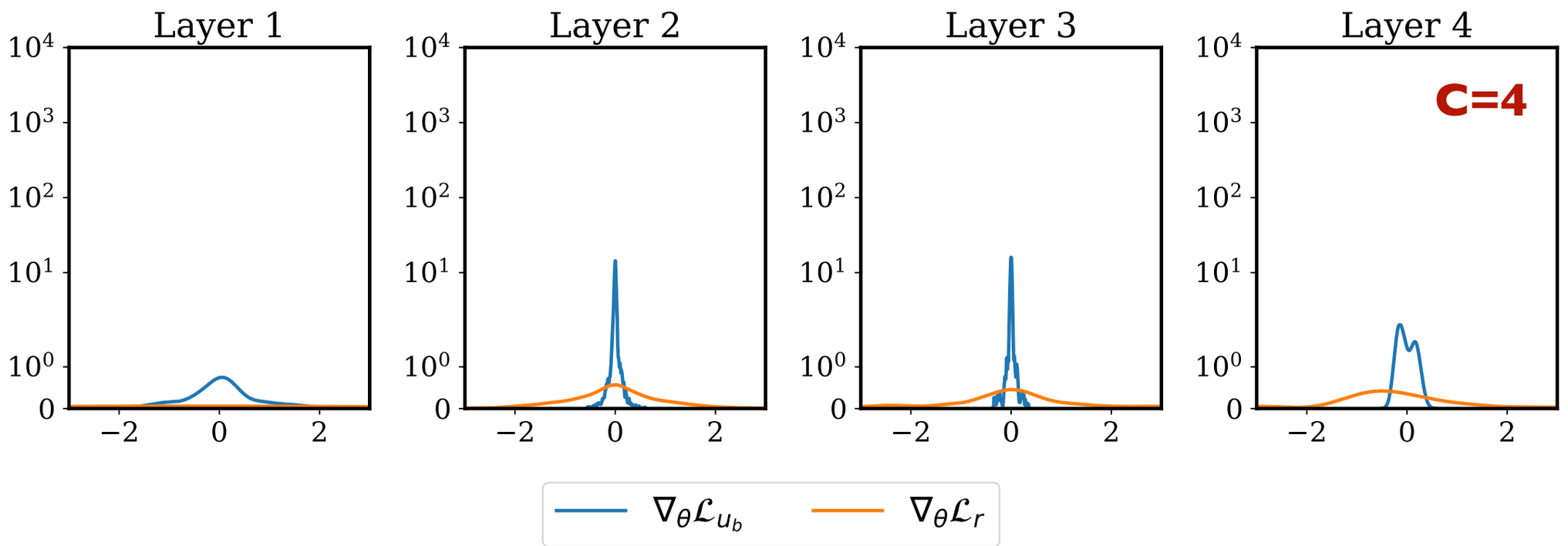
We can show that

$$\|\nabla_\theta \mathcal{L}_{u_b}(\theta)\|_{L^\infty} \leq 2\epsilon \cdot \|\nabla_\theta \epsilon_\theta(x)\|_{L^\infty}$$
$$\|\nabla_\theta \mathcal{L}_r(\theta)\|_{L^\infty} \leq O(C^4) \cdot \epsilon \cdot \|\nabla_\theta \epsilon_\theta(x)\|_{L^\infty}$$

Wang, S., Teng, Y., & Perdikaris, P. (2020). Understanding and mitigating gradient pathologies in physics-informed neural networks. arXiv preprint arXiv:2001.04536.

Gradient analysis for PINNs





Histograms of back-propagated gradients $\nabla_\theta \mathcal{L}_r(\theta)$ and $\nabla_\theta \mathcal{L}_{u_b}(\theta)$ at each layer for different constant C.

Stiffness in the gradient flow dynamics

$$\frac{d\theta}{dt} = -\nabla_\theta \mathcal{L}_r(\theta) - \sum_{i=1}^{M} \nabla_\theta \mathcal{L}_i(\theta)$$

Stiffness:

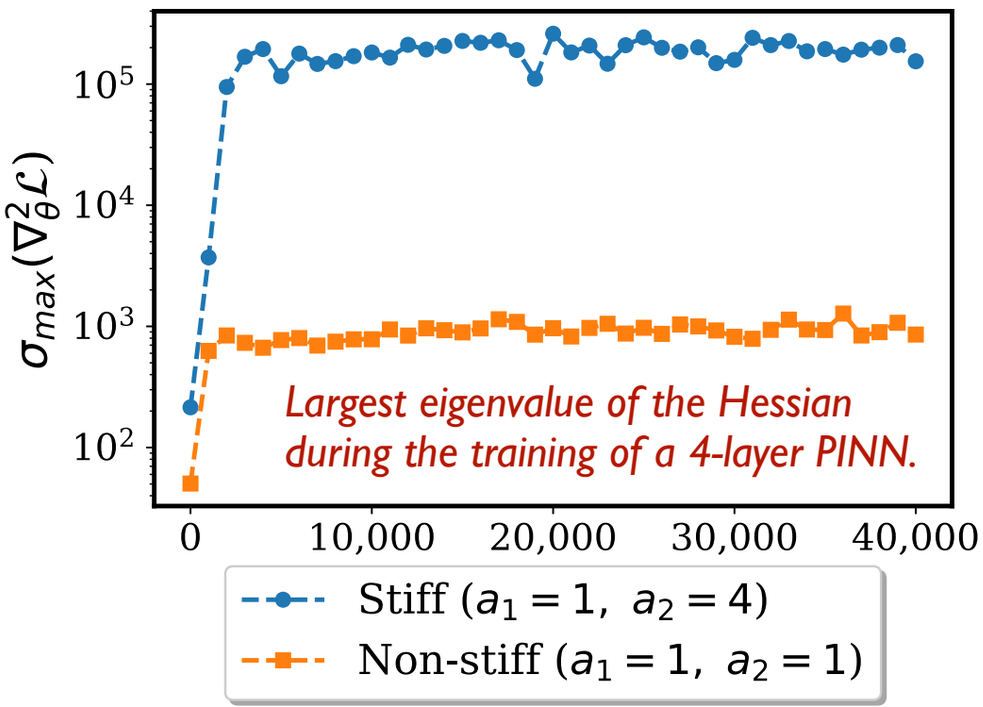
- Stiffness of the gradient flow can be characterized by the largest eigenvalue of $\nabla_\theta^2 \mathcal{L}(\theta)$.

Hypothesis:

- The stiffness exists in the gradient flow dynamics of PINNs.
- The stiffness leads to imbalanced gradients during model training using gradient descent.
- The stiffness leads to difficulties in training PINNs via gradient descent $\theta_{n+1} = \theta_n - \eta \nabla_\theta \mathcal{L}(\theta_n)$.

Example:
Helmholtz equation in 2D

$$\Delta u(x,y) + k^2 u(x,y) = q(x,y), \quad (x,y) \in \Omega := (-1,1)$$
$$u(x,y) = \sin(a_1 \pi x)\sin(a_2 \pi y)$$



Stiffness in the gradient flow dynamics

Example: Helmholtz equation in 2D, gradient descent update:

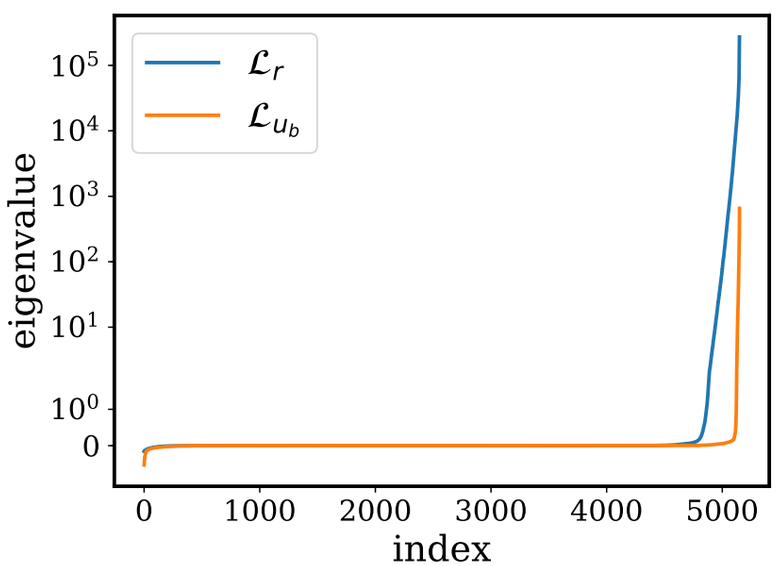
$$\theta_{n+1} = \theta_n - \eta \nabla_\theta \mathcal{L}(\theta_n) = \theta_n - \eta \left(\nabla_\theta \mathcal{L}_r(\theta_n) + \nabla_\theta \mathcal{L}_{u_b}(\theta_n) \right)$$

Applying second order Taylor expansion to the loss function $\mathcal{L}(\theta)$ at θ_n we can show that

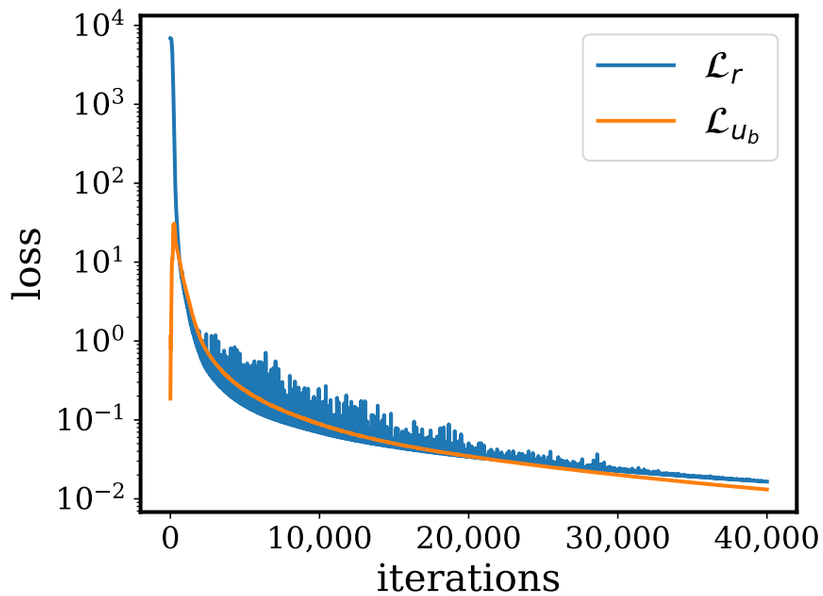
$$\mathcal{L}_r(\theta_{n+1}) - \mathcal{L}_r(\theta_n) = \eta \|\nabla_\theta \mathcal{L}(\theta_n)\|_2^2 (-1 + \frac{1}{2}\eta \sum_{i=1}^{N} \lambda_i^r y_i^2)$$

$$\mathcal{L}_{u_b}(\theta_{n+1}) - \mathcal{L}_{u_b}(\theta_n) = \eta \|\nabla_\theta \mathcal{L}(\theta_n)\|_2^2 (-1 + \frac{1}{2}\eta \sum_{i=1}^{N} \lambda_i^{u_b} y_i^2),$$

for some $\mathbf{y} = (y_1, \ldots, y_N)$ satisfying $\|\mathbf{y}\|_2^2 = \sum y_i^2 = 1$, where $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_N$ are eigenvalues of $\nabla_\theta^2 \mathcal{L}$.



Large Hessian eigenvalues indicate stiffness in the gradient flow dynamics



Even full-batch gradient descent can get trapped in limit cycles and does not guarantee a monotonic decrease of the loss.

Gradient pathologies in physics-informed neural networks

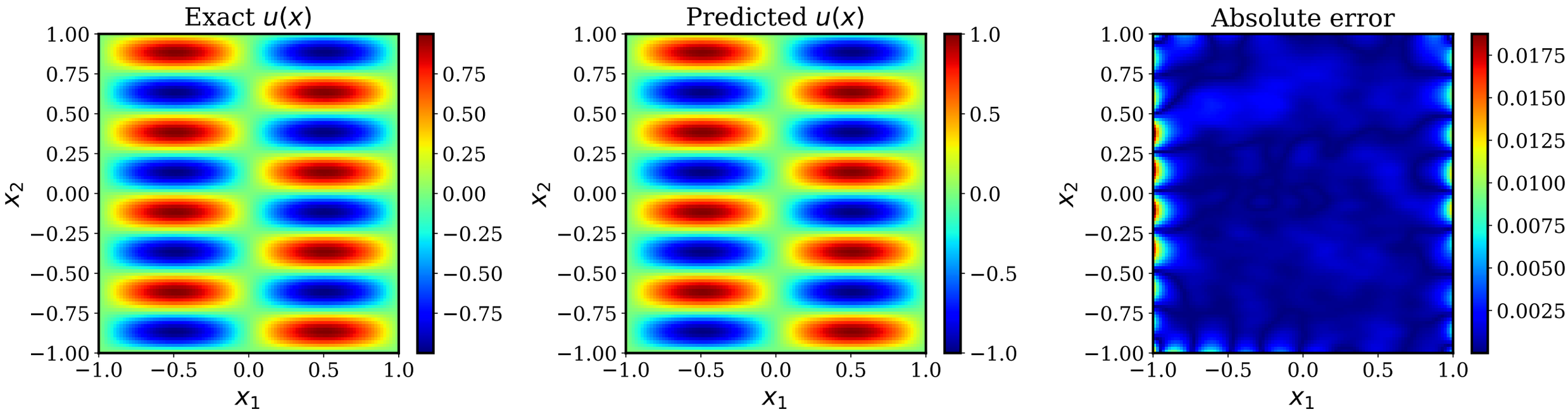
A simple benchmark (2D Helmholtz equation):

$$\Delta u + k^2 u = q(x,y) \;\; (x,y) \in [-1,1]$$

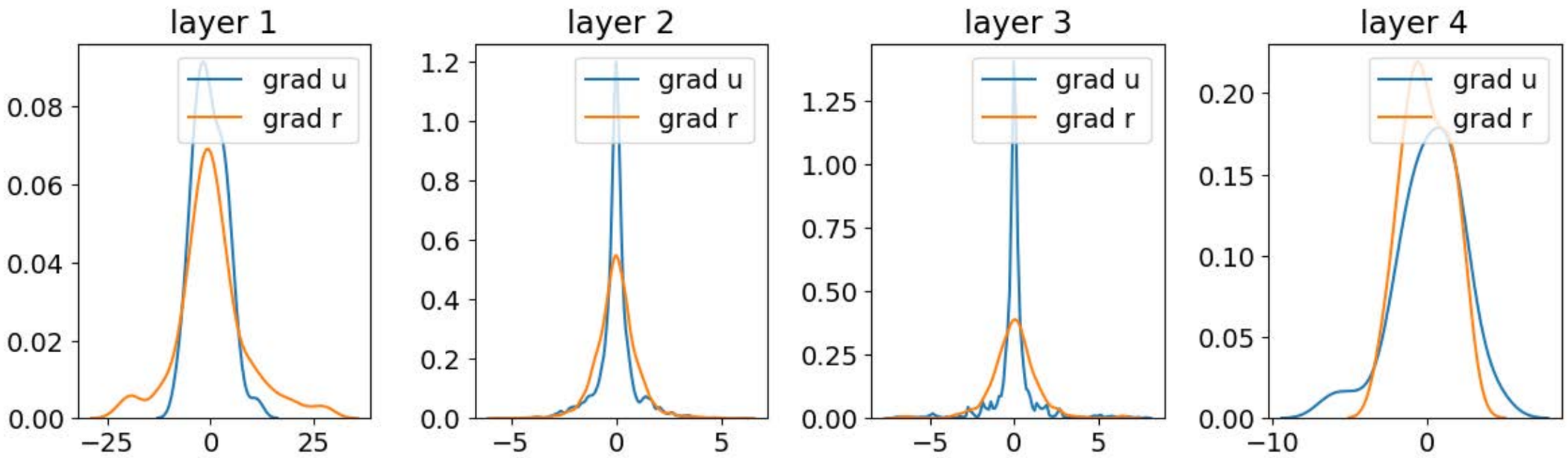
$$u(x,y) = (x+y)\sin(\pi x)\sin(6\pi y)$$

Loss function:

$$\mathcal{L}(\theta) := \lambda_1 \underbrace{\mathcal{L}_r(\theta)}_{\text{PDE residual}} + \lambda_2 \underbrace{\mathcal{L}_{u_b}(\theta)}_{\text{BCs fit}}$$



Prediction of a fully connected 4-layer deep physics-informed neural network (0.5% relative error)



Histograms of back-propagated gradients $\nabla_\theta \mathscr{L}_{u_b}(\theta), \nabla_\theta \mathscr{L}_r(\theta)$ *at each hidden layer*

…but how to choose the weights/learning rates?

$$\mathcal{L}(\theta) := \lambda_1 \underbrace{\mathcal{L}_u(\theta)}_{\text{Data fit}} + \lambda_2 \underbrace{\mathcal{L}_r(\theta)}_{\text{PDE residual}} + \lambda_3 \underbrace{\mathcal{L}_{u_0}(\theta)}_{\text{ICs fit}} + \lambda_4 \underbrace{\mathcal{L}_{u_b}(\theta)}_{\text{BCs fit}}$$

Adaptive moment estimation

Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize 1$^{\text{st}}$ moment vector)
$v_0 \leftarrow 0$ (Initialize 2$^{\text{nd}}$ moment vector)
$t \leftarrow 0$ (Initialize timestep)
while θ_t not converged **do**
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
$\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
$\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
end while
return θ_t (Resulting parameters)

…i.e. use the gradient statistics during training to adaptively adjust the learning rate.

A learning rate annealing algorithm for PINNs

Algorithm 1: Learning rate annealing for physics-informed neural networks

Consider a physics-informed neural network $f_\theta(\boldsymbol{x})$ with parameters θ and a loss function

$$\mathcal{L}(\theta) := \mathcal{L}_r(\theta) + \sum_{i=1}^{M} \lambda_i \mathcal{L}_i(\theta),$$

where $\mathcal{L}_r(\theta)$ denotes the PDE residual loss, the $\mathcal{L}_i(\theta)$ correspond to data-fit terms (e.g., measurements, initial or boundary conditions, etc.), and $\lambda_i = 1, i = 1, \ldots, M$ are free parameters used to balance the interplay between the different loss terms. Then use S steps of a gradient descent algorithm to update the parameters θ as:

for $n = 1, \ldots, S$ **do**

(a) Compute $\hat{\lambda}_i$ by

$$\hat{\lambda}_i = \frac{\max_\theta\{|\nabla_\theta \mathcal{L}_r(\theta_n)|\}}{\overline{|\nabla_\theta \lambda_i \mathcal{L}_i(\theta_n)|}}, \quad i = 1, \ldots, M, \tag{40}$$

where $\overline{|\nabla_\theta \lambda_i \mathcal{L}_i(\theta_n)|}$ denotes the mean of $|\nabla_\theta \lambda_i \mathcal{L}_i(\theta_n)|$ with respect to parameters θ.

(b) Update the weights λ_i using a moving average of the form

$$\lambda_i = (1 - \alpha)\lambda_i + \alpha \hat{\lambda}_i, \quad i = 1, \ldots, M. \tag{41}$$

(c) Update the parameters θ via gradient descent

$$\theta_{n+1} = \theta_n - \eta \nabla_\theta \mathcal{L}_r(\theta_n) - \eta \sum_{i=1}^{M} \lambda_i \nabla_\theta \mathcal{L}_i(\theta_n) \tag{42}$$

end

The recommended hyper-parameter values are: $\eta = 10^{-3}$ and $\alpha = 0.1$.

Wang, S., Teng, Y., & Perdikaris, P. (2020). Understanding and mitigating gradient pathologies in physics-informed neural networks. arXiv preprint arXiv:2001.04536.

Soft physics-informed learning, a recap

$$\mathcal{L}(\theta) := \underbrace{\frac{1}{N_u}\sum_{i=1}^{N_u}[\boldsymbol{u}_i - f_\theta(\boldsymbol{x}_i)]^2}_{\text{Data fit}} + \underbrace{\frac{1}{\lambda}\mathcal{R}[f_\theta(\boldsymbol{x})]}_{\text{Physics regularization}}$$

An "unconventional" regularizer/prior that requires us to revisit standard deep learning practices:

- loss functions (e.g., square residual, *variational principle, Hamiltonian, etc.?*)
- network initialization (e.g., Glorot, *adaptive?*)
- normalization (e.g., zero-mean/unit-variance, *PDE solution bounds?*)
- optimization (e.g., Adam, *adaptive learning rates, proximal algorithms, meta-learning?*)
- network architecture (e.g., fully connected, *residual/recurrent/convolutional layers, attention?*)

$$\frac{d\theta}{dt} = -\nabla_\theta \mathcal{L}_r(\theta) - \sum_{i=1}^{M} \nabla_\theta \mathcal{L}_i(\theta)$$

$$\theta_{n+1} = \theta_n - \eta \nabla_\theta \mathcal{L}_r(\theta_n) - \eta \sum_{i=1}^{M} \nabla_\theta \mathcal{L}_i(\theta_n)$$

Stiffness in the gradient flow dynamics.

An improved neural architecture

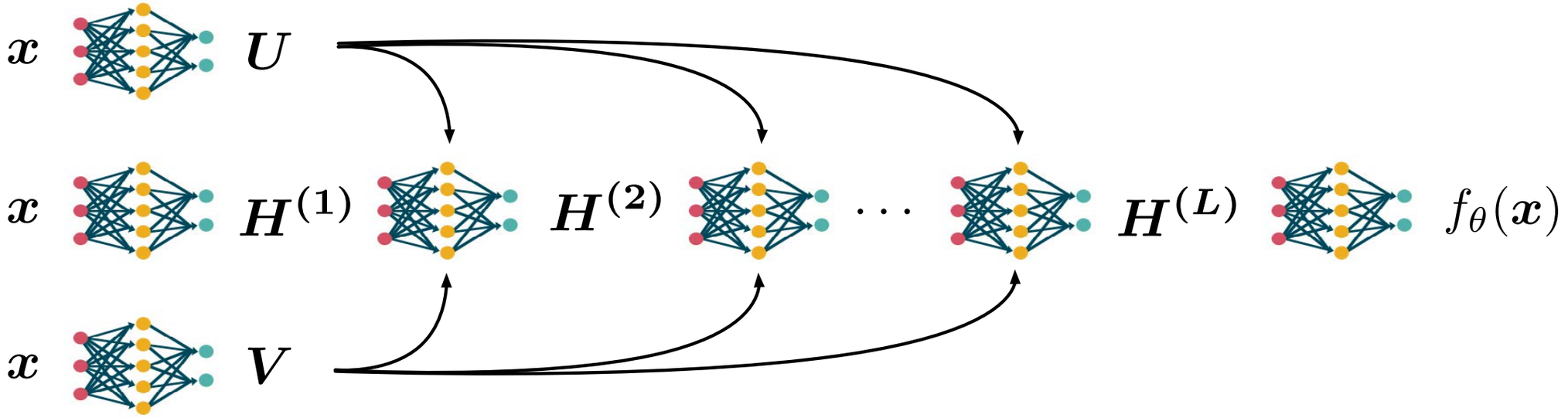
$$U = \phi(W^1 \vec{x} + b^1), \;\; V = \phi(W^2 \vec{x} + b^2)$$

$$H^{(1)} = \phi(W^{z,1} \vec{x} + b^{z,1})$$

$$Z^{(k)} = \phi(W^{z,k} H^{(k)} + b^{z,k}), \;\; k = 1, \ldots, L$$

$$H^{(k+1)} = (1 - Z^{(k)}) \odot U + Z^{(k)} \odot V, \;\; k = 1, \ldots, L$$

$$f(x; \theta) = W H^{(L+1)} + b$$



$$\theta = \{W^1, b^1, W^2, b^2, (W^{z,l}, b^{z,l})_{l=1}^{L}, W, b\}$$

Key points:

- Account for multiplicative interactions of the inputs, similar to attention mechanisms.
- Residual connections improve resilience against vanishing gradient pathologies.

Systematic comparison

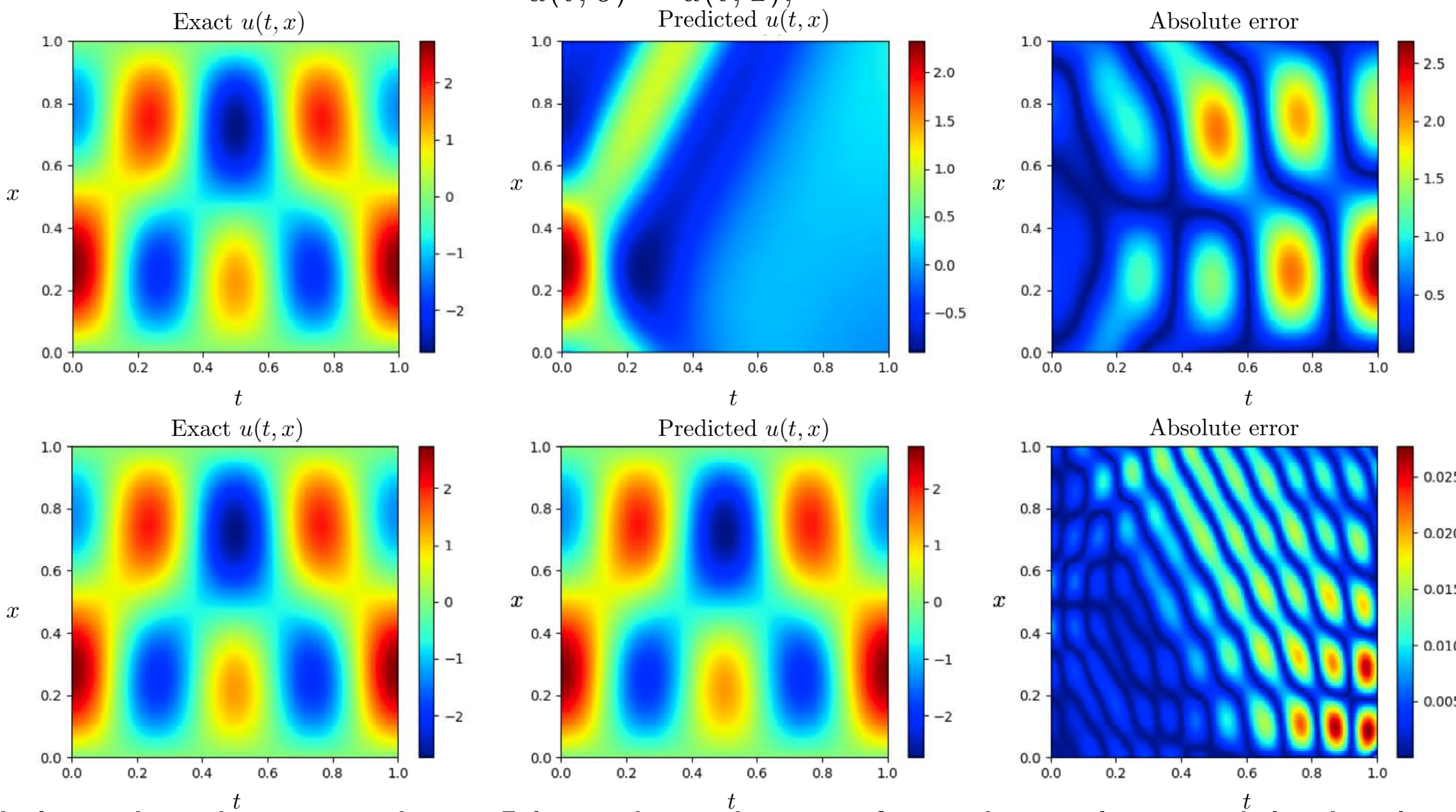
M1: Baseline PINN model (Raissi et. al., 2019)
M2: PINN with the proposed learning rate annealing
M3: PINN with the proposed neural architecture
M4: PINN with the proposed learning rate annealing and improved neural architecture

Architecture	M1	M2	M3	M4
30 units / 3 hidden layers	2.44E-01	3.98E-02	5.31E-02	**2.56E-03**
50 units / 3 hidden layers	1.06E-01	1.58E-02	2.46E-02	**1.81E-03**
100 units / 3 hidden layers	9.07E-02	2.39E-03	1.17E-02	**1.28E-03**
30 units / 5 hidden layers	2.47E-01	8.91E-03	4.12E-02	**1.96E-03**
50 units / 5 hidden layers	1.40E-01	8.08E-03	1.97E-02	**1.86E-03**
100 units / 5 hidden layers	1.15E-01	3.25E-03	1.08E-02	**1.22E-03**
30 units / 7 hidden layers	3.10E-01	7.86E-03	3.17E-02	**1.98E-03**
50 units / 7 hidden layers	1.98E-01	3.66E-03	2.37E-02	**1.54E-03**
100 units / 7 hidden layers	8.14E-02	2.57E-03	9.36E-03	**1.40E-03**

Relative prediction error (L2 norm) averaged over 10 independent trials for the 2D Helmholtz benchmark.

Wave equation

$$u_{tt} = 4u_{xx}, \quad (t, x) \in [0, 1]$$
$$u(0, x) = h(x),$$
$$u(t, 0) = u(t, 1),$$



Top: Imbalanced gradients in a dense, 5-layer deep physics-informed neural network lead to large prediction errors (76%).

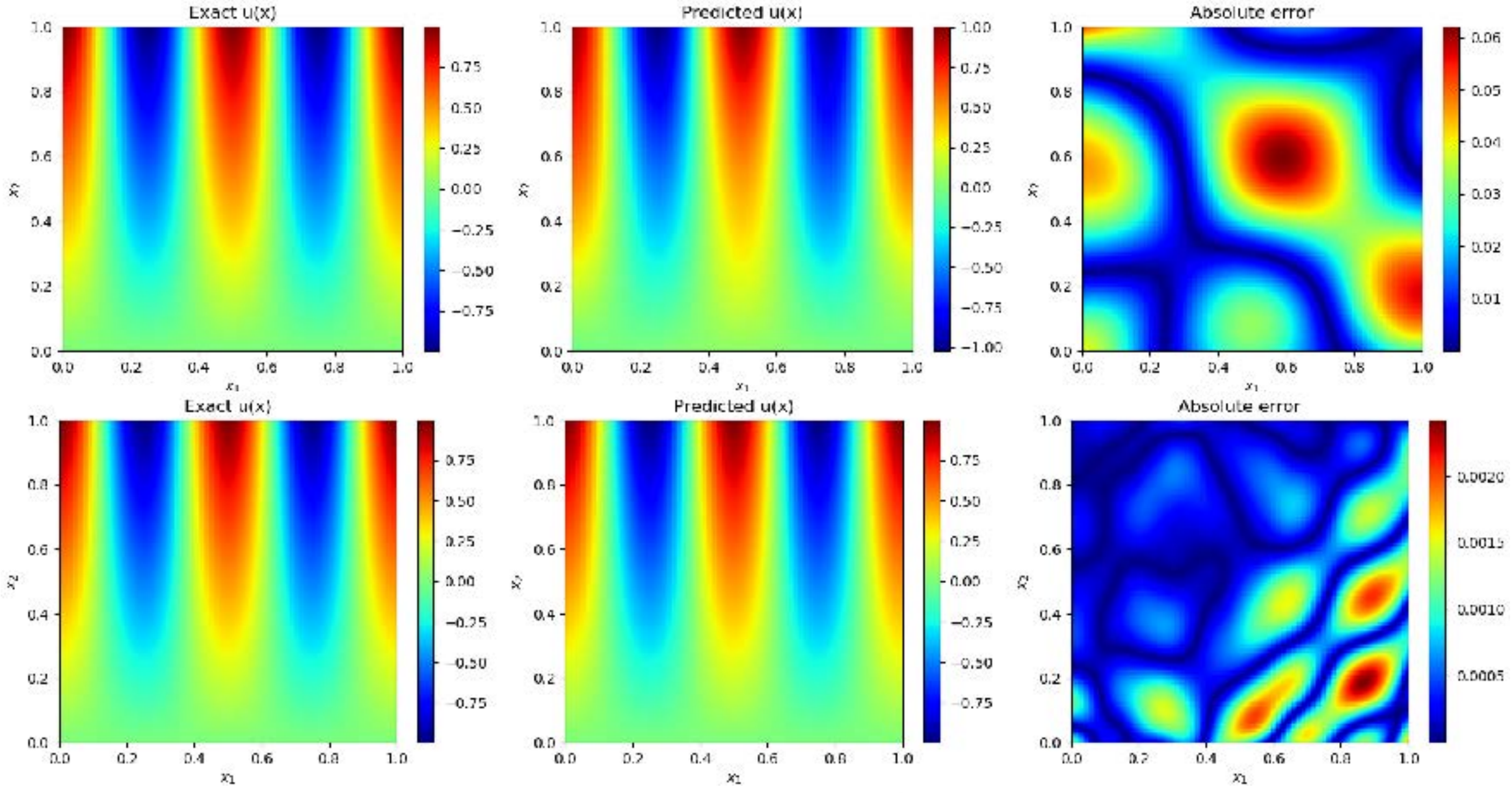
Bottom: Accurate predictions can be obtained using the proposed learning rate annealing and improved neural architecture strategy (relative prediction error: 0.6%).

Klein Gordon equation

$$u_{tt} + \alpha u_{xx} + \beta u + \gamma u^k = f(x,t), \quad (x,t) \in \Omega \times [0,T]$$

$$u(x,0) = g_1(x), \quad x \in \Omega$$
$$u_t(x,0) = g_2(x), \quad x \in \Omega$$

$$u_(x,t) = h(x,t), \quad (x,t) \in \partial\Omega \times [0,T]$$



Top: Imbalanced gradients in a dense, 5-layer deep physics-informed neural network lead to considerable prediction errors (6.7%).

Bottom: Accurate predictions can be obtained using the proposed learning rate annealing and improved neural architecture strategy (relative prediction error: 0.1%).

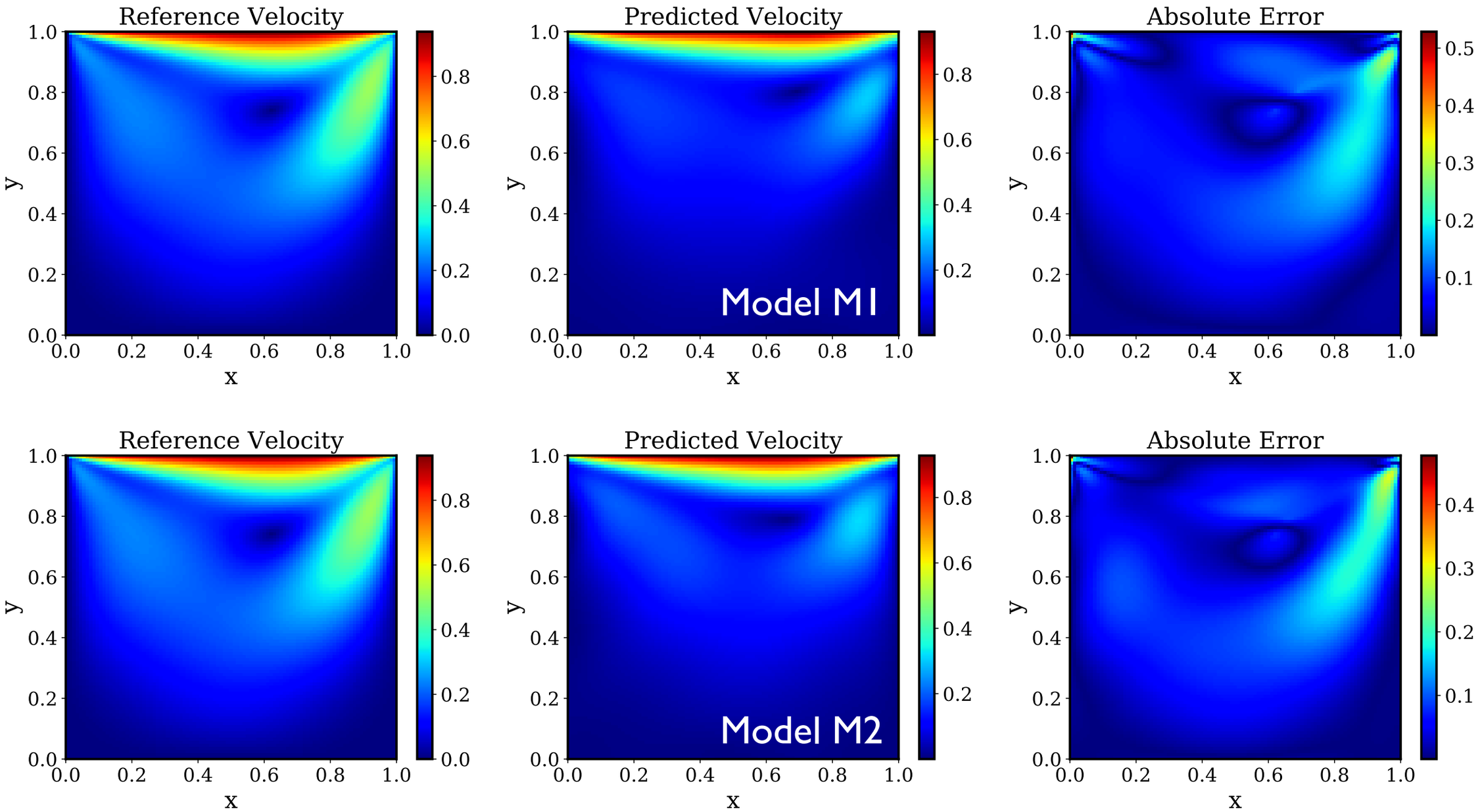
Flow in a lid-driven cavity

$$\boldsymbol{u} \cdot \nabla \boldsymbol{u} + \nabla p - \frac{1}{Re} \Delta \boldsymbol{u} = 0 \qquad \text{in } \Omega$$

$$\nabla \cdot \boldsymbol{u} = 0 \qquad \text{in } \Omega$$

$$\boldsymbol{u}(\boldsymbol{x}) = (1, 0) \qquad \text{on } \Gamma_1$$

$$\boldsymbol{u}(\boldsymbol{x}) = (0, 0) \qquad \text{on } \Gamma_0$$



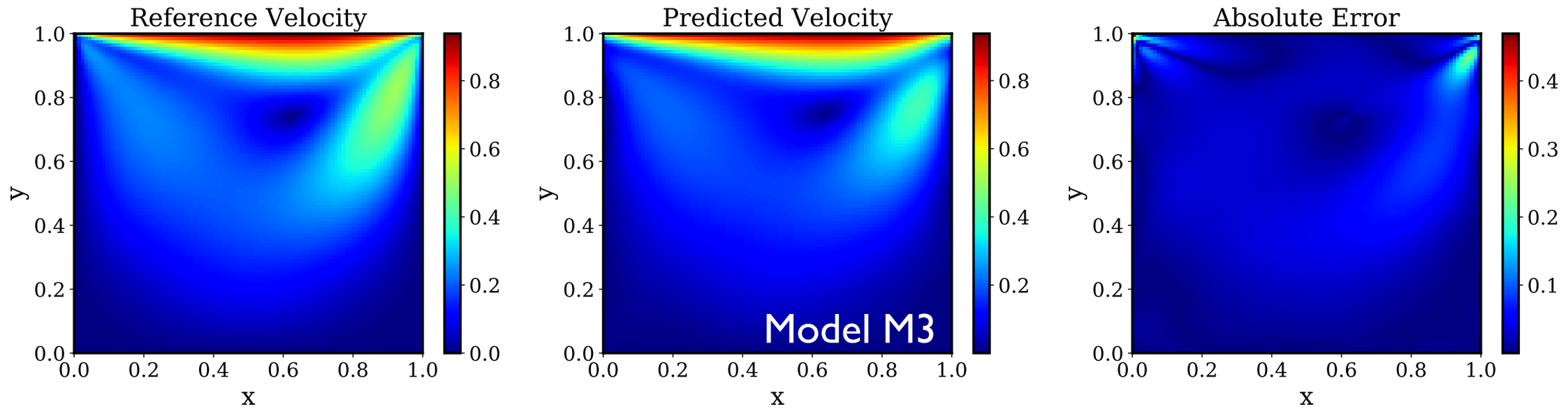
Flow in a lid-driven cavity

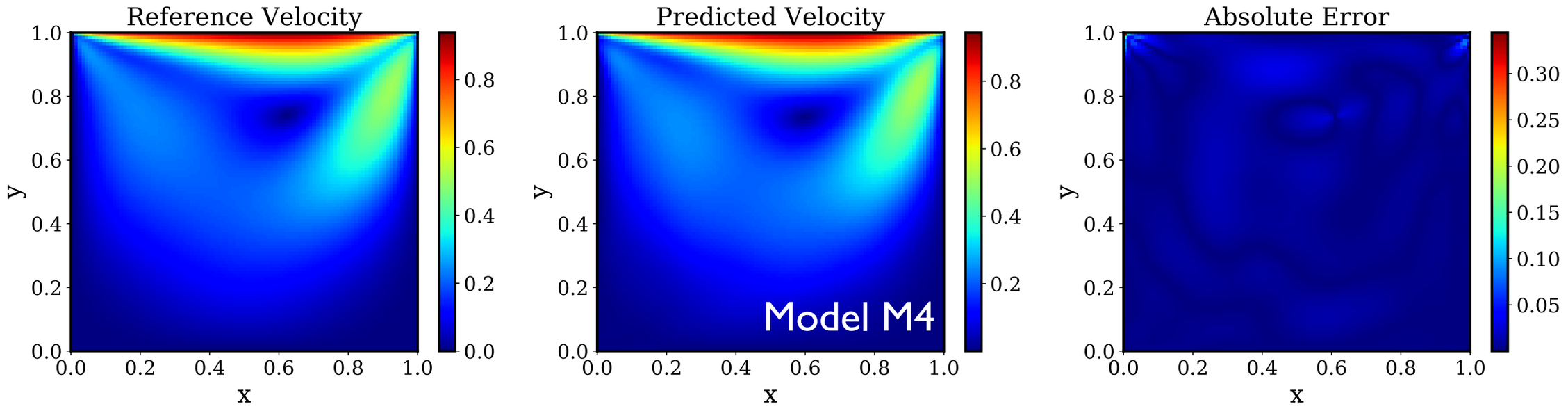
$$\boldsymbol{u} \cdot \nabla \boldsymbol{u} + \nabla p - \frac{1}{Re} \Delta \boldsymbol{u} = 0 \qquad \text{in } \Omega$$

$$\nabla \cdot \boldsymbol{u} = 0 \qquad \text{in } \Omega$$

$$\boldsymbol{u}(\boldsymbol{x}) = (1, 0) \qquad \text{on } \Gamma_1$$

$$\boldsymbol{u}(\boldsymbol{x}) = (0, 0) \qquad \text{on } \Gamma_0$$





Summary

- Function space constraints in introduce "unconventional" regularizers/priors that requires us to revisit standard deep learning practices.
- Constraints alter the loss landscape of neural networks. Different terms in such composite loss function may have different nature and magnitudes, leading to imbalanced gradients during back-propagation.
- Adaptive annealing of learning rates can balance the interplay between different terms in a constrained loss function and lead to improved solutions.
- Novel architectures can also safe-guard against gradient-related pathologies and lead to improved solutions.
- Using the proposed workflow we have observed consistent improvements in the predictive accuracy of physics-informed neural networks by a factor of 50-100x across a range of problems in computational physics.
- These developments are not limited to PINNs, but can be straightforwardly generalized to other tasks involving the interplay of multiple objective functions that may lead to unbalanced gradients issue, e.g. multi-task learning.
- Despite some progress, we are still at the very early stages of understanding the capabilities and limitations of such models.