Neural Methods with Natural Gradient Acceleration

Nicolas PAILLIEZ, IRMA - Strasbourg
Emmanuel FRANCK, INRIA - Strasbourg
Victor MICHEL-DANSAC, INRIA - Strasbourg
Laurent NAVORET, IRMA - Strasbourg
Stanislas PAMELA, UKEAEA - Culham

Table of Contents

Table of Contents

• A Physics-Informed Neural Network (PINN) solves differential equations by embedding physical laws, approximating solutions as $u_\theta(x, t)$.

• For the PINN u_θ, these equations are approximated as follows:

$$\begin{cases} \mathcal{L}(u_\theta, x, t, \mu) \approx 0, & \text{for } (x, t, \mu) \in \Omega \times (0, T) \times P, \\ u_\theta(x, t, \mu) \approx g(x, t, \mu), & \text{for } (x, t) \in \partial\Omega \times (0, T) \times P, \\ u_\theta(x, 0, \mu) \approx u_0(x, \mu), & \text{for } x \in \Omega \times P. \end{cases}$$

- The idea behind training PINNs is to construct a loss function based on the residual of the PDE, given by:

$$J_{\mathbf{PDE}}(\theta) = \int_\Omega \int_P \int_0^T \|\mathcal{L}(u_\theta, x, t, \mu)\|_2^2 \, dt \, d\mu \, dx + \int_{\partial\Omega} \int_P \int_0^T \|u_\theta(x, t, \mu) - g(x, t, \mu)\|_2^2 \, dt \, d\mu \, dx$$
$$+ \int_\Omega \int_P \|u_\theta(x, 0, \mu) - u_0(x, \mu)\|_2^2 \, d\mu \, dx,$$

- We applying the Monte Carlo method :

$$J_{\mathbf{PDE}}(\theta) = \frac{1}{N_\mu} \frac{1}{N_x} \frac{1}{N_t} \sum_{i=1}^{N_\mu} \sum_{j=1}^{N_x} \sum_{k=1}^{N_t} \|\mathcal{L}(\tilde{u}_\theta, x_j, t_k; \mu_i)\|_2^2 + \frac{1}{N_\mu} \frac{1}{N_x} \sum_{i=1}^{N_\mu} \sum_{j=1}^{N_x} \|\tilde{u}_\theta(x_j, 0; \mu_i) - u_0(x_j; \mu_i)\|_2^2$$

- Optimization then consists of solving:

$$\theta_{\text{opt}} = \arg\min_\theta J_{\text{PDE}}(\theta)$$

Advantages: PINNs associated with the Monte Carlo method have the following advantages

- Does not require a mesh
- Independent of the dimension

Drawbacks: PINNs associated with the Monte Carlo method have the following disadvantages

- Lower accuracy compared to classical methods
- Violates temporal causality

Natural Gradient

Goal: Minimize a cost function $L(\theta) = \mathcal{E}(u_\theta)$, where $u_\theta \in \mathcal{F}$, a Hilbert space of functions.

Limitation of standard gradient:

- The Euclidean gradient $\nabla L(\theta)$ does not reflect the geometry of the function space.
- Small changes in parameters θ may induce large, uncontrolled changes in u_θ.

Need: A gradient direction that is optimal in function space, not just in parameter space.

Reference: S. Amari (1998), *Natural Gradient Works Efficiently in Learning*

Natural Gradient

Key idea: Equip the parameter space with a metric induced by the function space $\mathcal{F}$:

$$G_{ij}(\theta) = \langle \partial_{\theta_i} u_\theta, \partial_{\theta_j} u_\theta \rangle_{\mathcal{F}}$$

Natural Gradient:

$$\nabla_{\text{nat}} L(\theta) := G(\theta)^{+} \nabla L(\theta)$$

where G^{+} denotes pseudo-inverse.

Geometric interpretation: The natural gradient follows the steepest descent direction **in $\mathcal{F}$**, i.e., the most effective way to change u_θ.

References:

J. Müller and M. Zeinhofer (2023), *Achieving High Accuracy with PINNs via Energy Natural Gradients*.

N. Schwencke and C. Furtlehner (2025), *ANAGRAM: A Natural Gradient Relative to Adapted Model for Efficient PINNs Learning*, ICLR 2025.

Elliptic and Linear PDE System

We consider the following elliptic and linear PDE system:

$$\begin{cases} L(u(x)) = -\nabla \cdot (A(x)\nabla u(x)) + \nabla \cdot (\beta(x)u(x)) + c(x)u(x) = f(x), & \forall x \in \Omega \subset \mathbb{R}^d, \\ u(x) = 0, & \forall x \in \partial\Omega. \end{cases}$$

Here:

- $A(x)$ is a matrix of regular functions,
- $\beta(x)$ is a vector of regular functions,
- $c(x)$ and $f(x)$ are regular functions.

Example 1 : Poisson's Equation

We are interested in the following PDE (Poisson's equation):

$$-\Delta u = f$$

where f is defined as:

$$f(x, \mu) = \mu \cdot 8\pi^2 \sin(2\pi x_1) \sin(2\pi x_2)$$

The exact solution is given by:

$$u_{\text{exact}}(x, \mu) = \mu \cdot \sin(2\pi x_1) \sin(2\pi x_2)$$

Example 1

Poisson's equation, layers [40] = 201 ndof

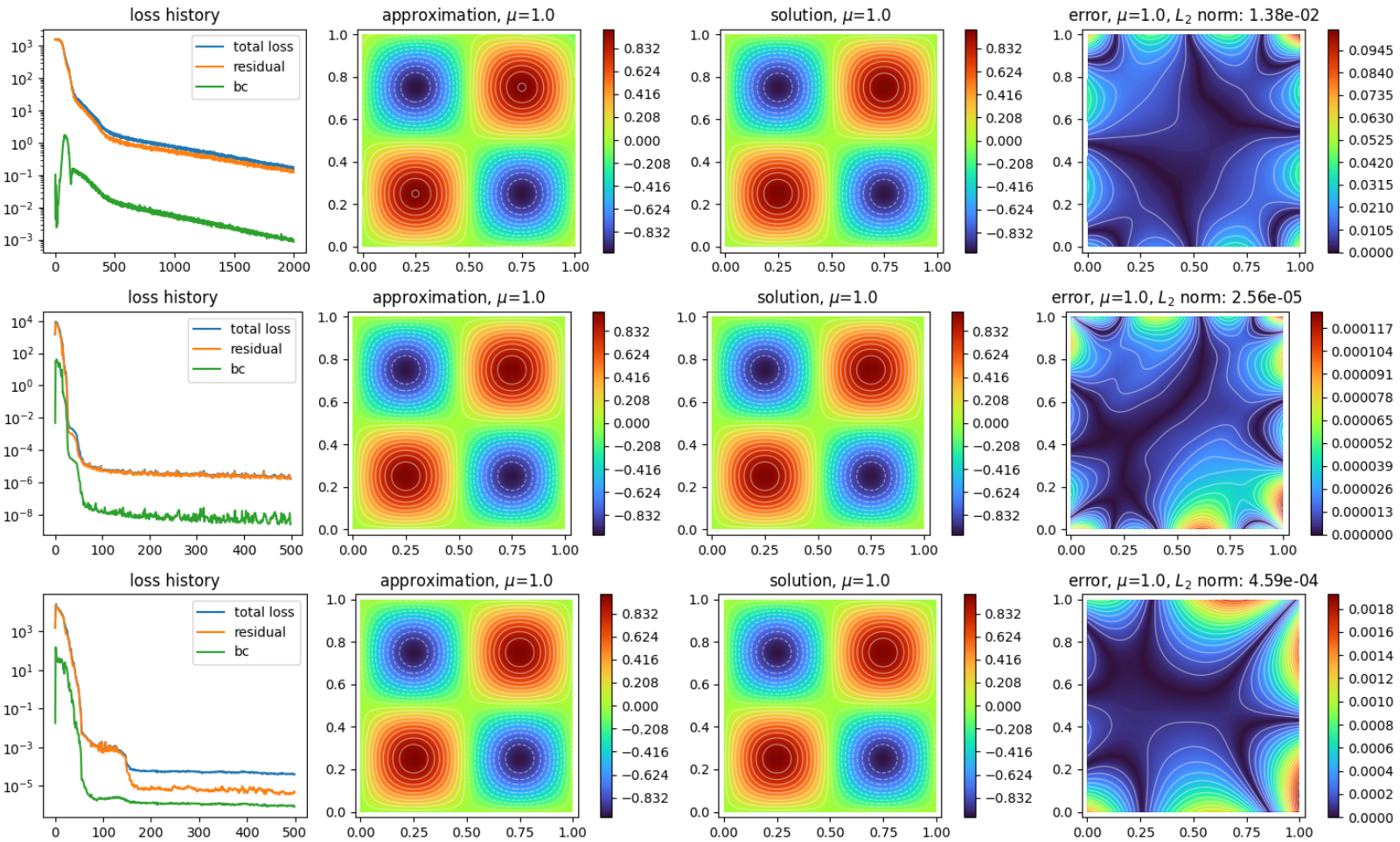

Figure: Result of a PINNs with Scimba

Example 1

Poisson's equation, layers [40] = 201 ndof

	Mean	Standard Deviation	Min	Max
Classical	3.35×10^{-2}	1.85×10^{-2}	1.16×10^{-2}	6.38×10^{-2}
ENG	9.67×10^{-5}	1.38×10^{-4}	1.32×10^{-5}	4.89×10^{-4}
ANA	8.69×10^{-4}	1.07×10^{-3}	2.55×10^{-4}	3.95×10^{-3}

Table: Statistical summary of relative L2 errors

Example 2 : Anistropic problem

We are interested in the following PDE:

$$-\nabla \cdot (A \nabla u) = f$$

where:

- $A(x) = \begin{pmatrix} \varepsilon x_1^2 + x_2^2 & (\varepsilon - 1) x_1 x_2 \\ (\varepsilon - 1) x_1 x_2 & x_1^2 + \varepsilon x_2^2 \end{pmatrix}$,
- $f(x_1, x_2) = a \exp\left(- \frac{(x_1 - c_1)^2 + (x_2 - c_2)^2}{2\sigma^2} \right)$,
- $a = 10, \quad c_1 = 0.5, \quad c_2 = 0.5, \quad \sigma = 0.01, \quad \varepsilon = 0.005.$

Example 2

Anistropic equation, FEM solution

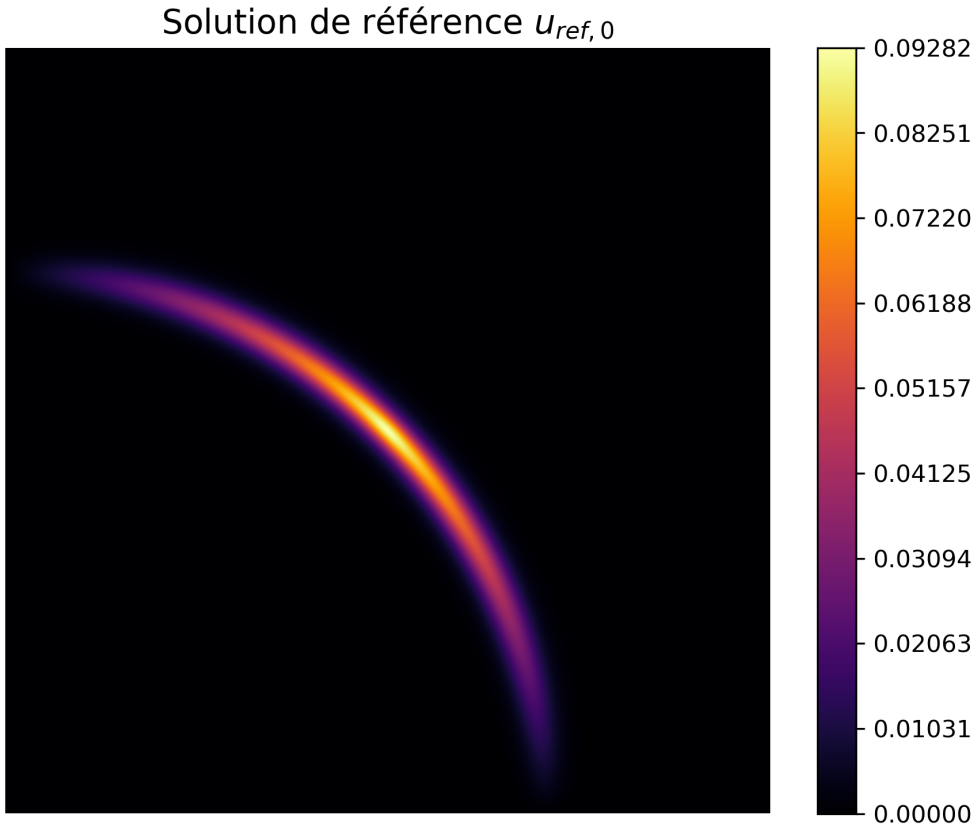

Figure: FEM solution with FEnics

Example 2

Anistropic equation, layers = [30]*3, nb_colloc = 8000

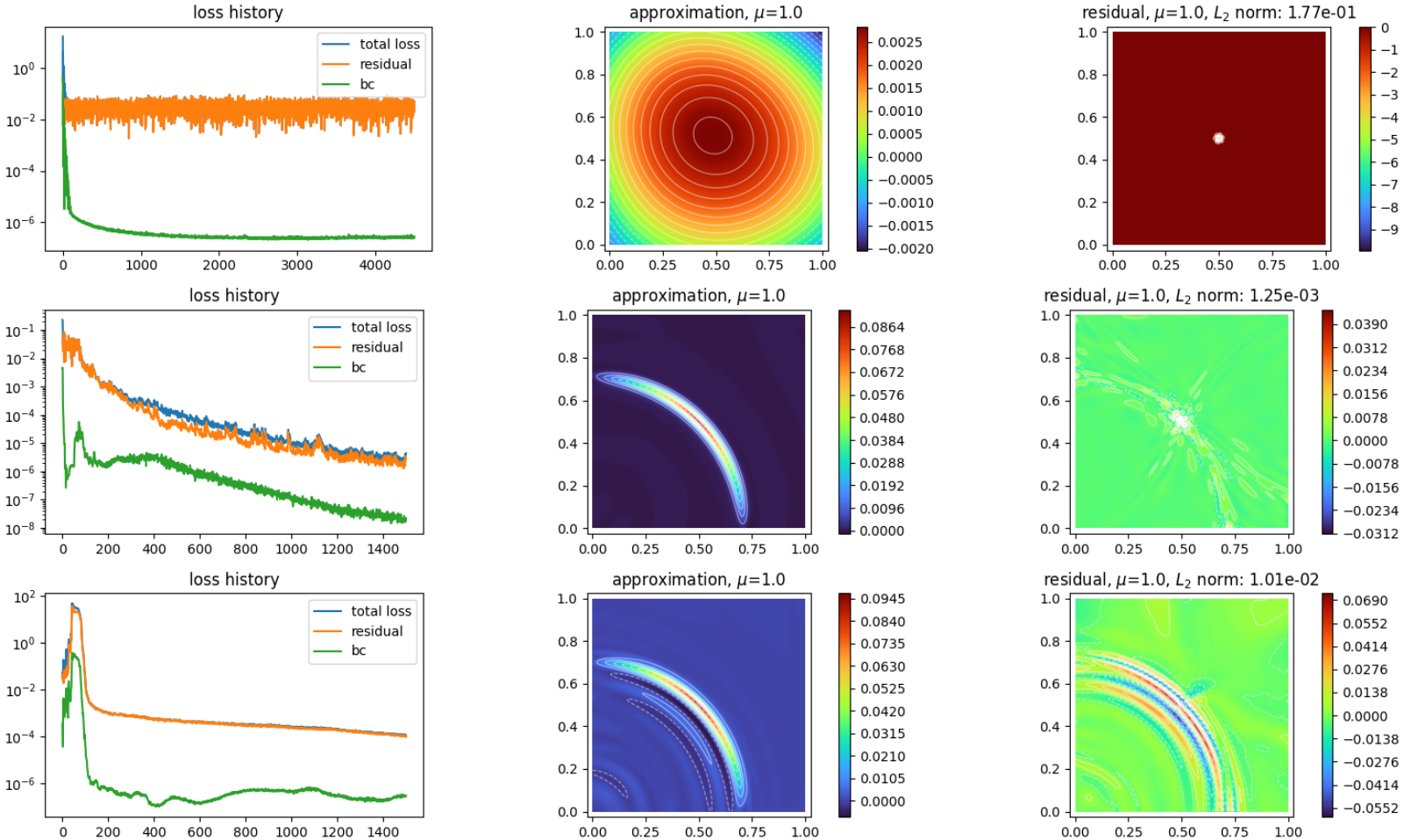

Figure: PINNs solution with Scimba

Example 2

Anistropic equation, layers = [30]*3, nb_colloc = 8000

	Mean	Std. Dev.	Min	Max
Classical	9.90×10^{-1}	1.71×10^{-2}	9.73×10^{-1}	1.02×10^{0}
ENG	5.72×10^{-2}	1.50×10^{-2}	3.90×10^{-2}	7.30×10^{-2}
ANA	1.79×10^{-1}	2.12×10^{-2}	1.52×10^{-1}	2.14×10^{-1}

Table: Statistical summary of relative L2 errors (uniform sampling, with ADAM)

Deep Ritz Method

Consider the Poisson problem:

$$\begin{cases} -\Delta W = f & \text{in } \Omega, \\ W = 0 & \text{on } \partial\Omega. \end{cases}$$

The energy formulation is:

$$W = \mathit{argmin}_{\psi \in H_0^1(\Omega)} \left(\frac{1}{2} \int_\Omega |\nabla \psi|^2 \, dx - \int_\Omega f \psi \, dx \right).$$

Approximate W using a nonlinear function $\phi(x, \theta)$:

$$W_\theta(x) = \phi(x, \theta).$$

Optimize θ:

$$\theta = \mathit{argmin}_{\vartheta \in \mathbb{R}^N} J(\vartheta), \quad J(\vartheta) = \frac{1}{2} \int_\Omega |\nabla \phi(x, \vartheta)|^2 \, dx - \int_\Omega f(x) \phi(x, \vartheta) \, dx.$$

Table of Contents

Temporal Causality and Time-Space PINNs

Time-Space Physics-Informed Neural Network (PINN) does not respect temporal causality.

- **Temporal Causality**: This principle requires that a neural network must be sufficiently trained at time t before progressing to time $t + 1$.
- **Time-Space PINNs**: These networks incorporate both spatial and temporal dimensions but do not inherently respect the progression of time in the training process.

Thus, Time-Space PINNs may fail to accurately model dynamic systems with time-dependent states.

Discrete PINN is based on a discrete-time formulation.
We consider the following PDE:

$$\partial_t u = \mathcal{L}(u)$$

- We discretize in time (Euler scheme):

$$u(t_{k+1}, x) = u(t_k, x) + \Delta t G(u(t_k, x)) + O(t^2)$$

- We assume that the model parameters evolve over time, meaning:

$$u(t, x) = nn_{\theta(t)}(x)$$

This leads to the Discrete PINN associated with the Euler scheme:

$$\theta_{k+1} = \min_{\theta} \int_{\Omega} \|nn_{\theta}(x) - nn_{\theta_k}(x) - \Delta t G(nn_{\theta_k}(x))\|_2^2$$

At each time step, we solve this using a Monte Carlo approach, similar to the classical PINN method, followed by an optimization process combining ADAM and L-BFGS algorithms.

- We start from the discrete PINN formulation:

$$\theta_{k+1} = \min_{\theta} \int_{\Omega} \|nn_{\theta}(x) - nn_{\theta_k}(x) - \Delta t G(nn_{\theta_k}(x))\|_2^2.$$

- Since $nn_{\theta}(x)$ is intended to approximate the solution at time t_{k+1}, we assume that θ_k is not far from the optimal solution and propose to linearize around θ_k. The linearization is as follows:

$$nn_{\theta}(x) = (\nabla_{\theta_k} u_{\theta_k})(\theta - \theta_k) + nn_{\theta_k}(x) + O((\theta - \theta_k)^2).$$

- Substituting this into the original formulation gives:

$$\theta_{k+1} = \min_{\theta} \int_{\Omega} \|(\nabla_{\theta_k} u_{\theta_k})(\theta - \theta_k) - \Delta t G(nn_{\theta_k}(x))\|_2^2.$$

1. The above problem is a least-squares problem, which can be solved numerically using an appropriate solver.
2. It can also be solved analytically using the normal equation in the L^2 inner product:

$$M(\theta_k)\theta_{k+1} = M(\theta_k)\theta_k - \Delta t R(\theta_k),$$

where:

$$M(\theta_k) = \int_\Omega (\nabla_{\theta_k} nn_{\theta_k}) \otimes (\nabla_{\theta_k} nn_{\theta_k}) dx, \quad \text{(mass matrix)}$$

$$R(\theta_k) = \int_\Omega (\nabla_{\theta_k} nn_{\theta_k}) G(nn_{\theta_k}(x)) dx.$$

Since matrix inversion can be problematic, it is preferable to solve:

$$(M(\theta_k) + \varepsilon I_d)\theta_{k+1} = M(\theta_k)\theta_k - \Delta t R(\theta_k).$$

Exemple 3 : Heat equation

We are interested in solving the heat equation:

$$\frac{\partial u}{\partial t} - D\Delta u = f$$

with:

- Diffusion coefficient: $D = 0.02$
- Source term: $f = 0$
- Final time: $T = 0.3$, time step: $\Delta t = 0.001$
- Exact solution:

$$u_{\text{exact}}(t, x_1, x_2) = e^{-D\pi^2 t} \cdot \sin(\pi x_1) \cdot \sin(\pi x_2)$$

Example 3

Heat equation : DPINNs, T = 0.3, dt = 0.003, nb_colloc = 2000

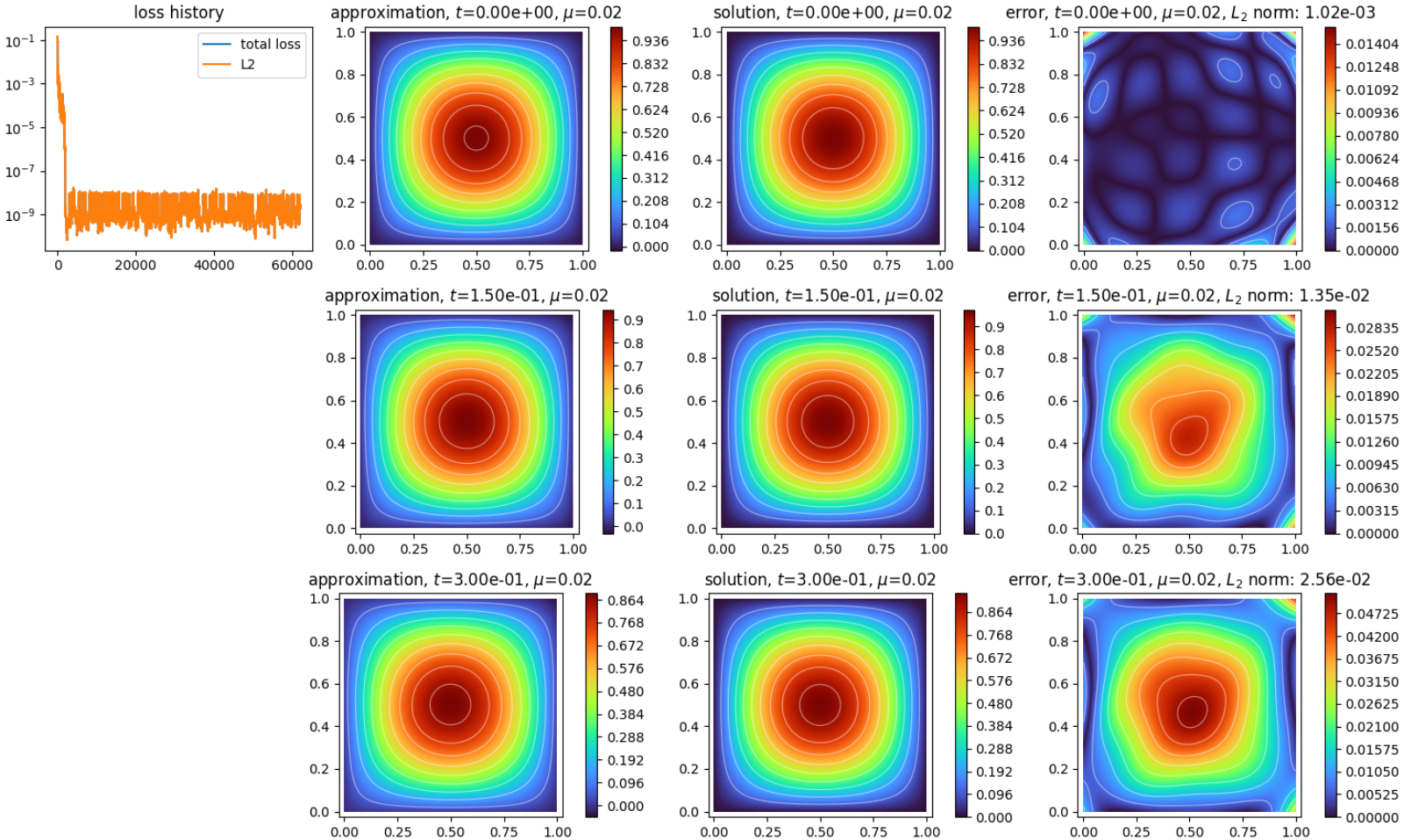

Figure: Results of Discrete PINN

Example 3

Heat equation : DPINNs with Natural Gradient (ENG)

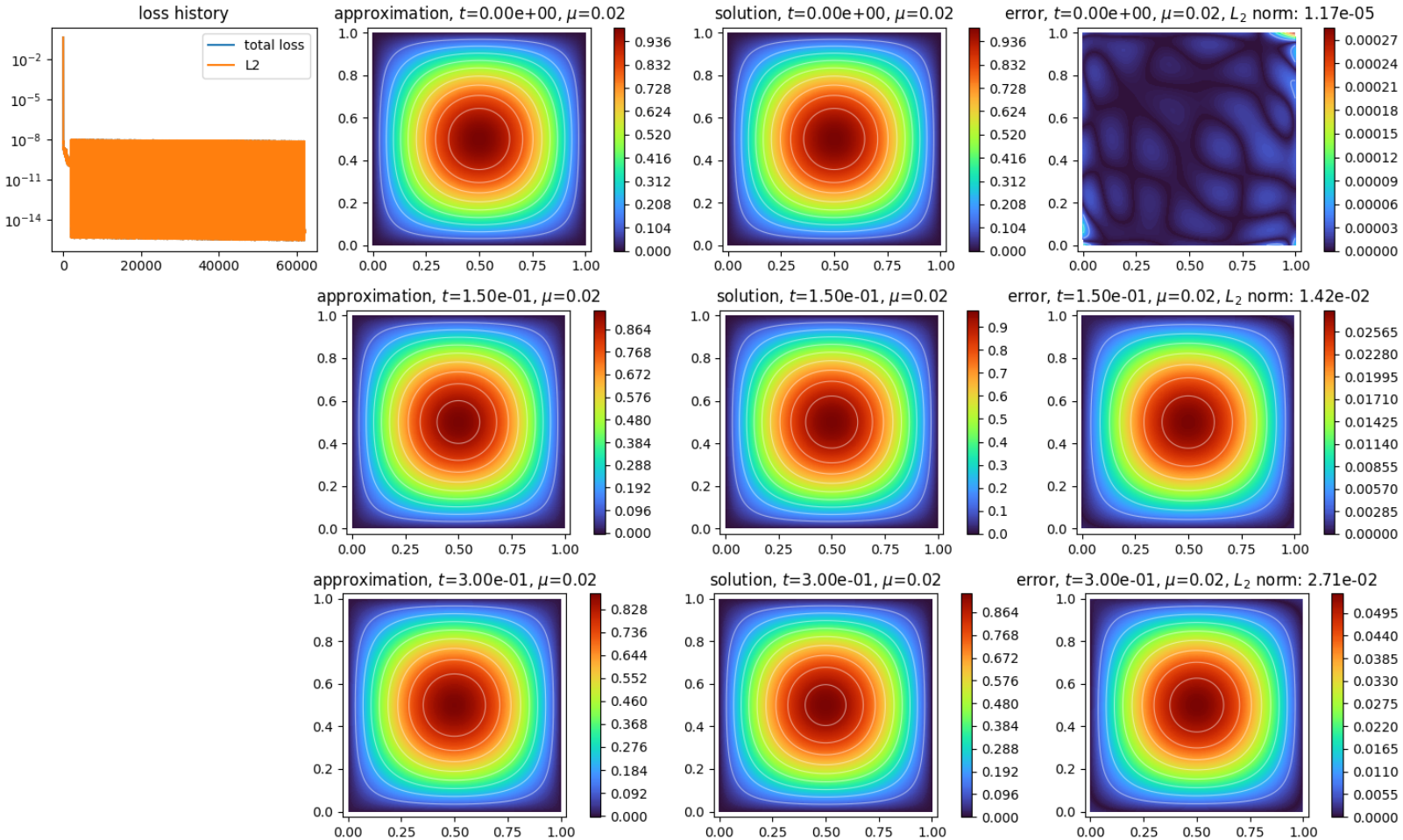

Figure: Results of Discrete PINN with Natural Gradient

Example 3

Heat equation : Neural Galerkin, T = 0.3, dt = 0.003, nb_colloc = 2000

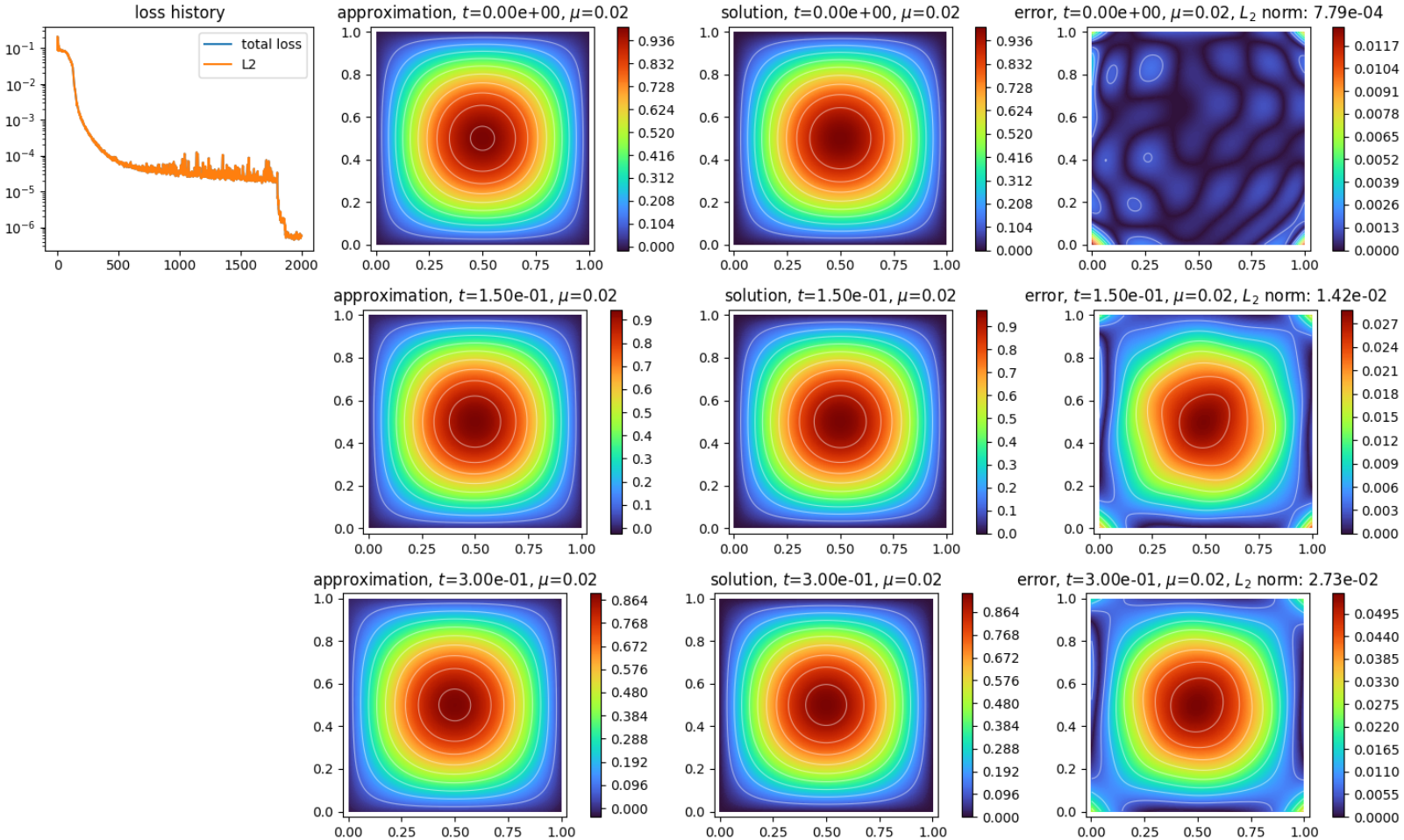

Figure: Results of Neural Galerkin Method

Example 3

Heat equation : Neural Galerkin with Natural Gradient (ANAGRAM)

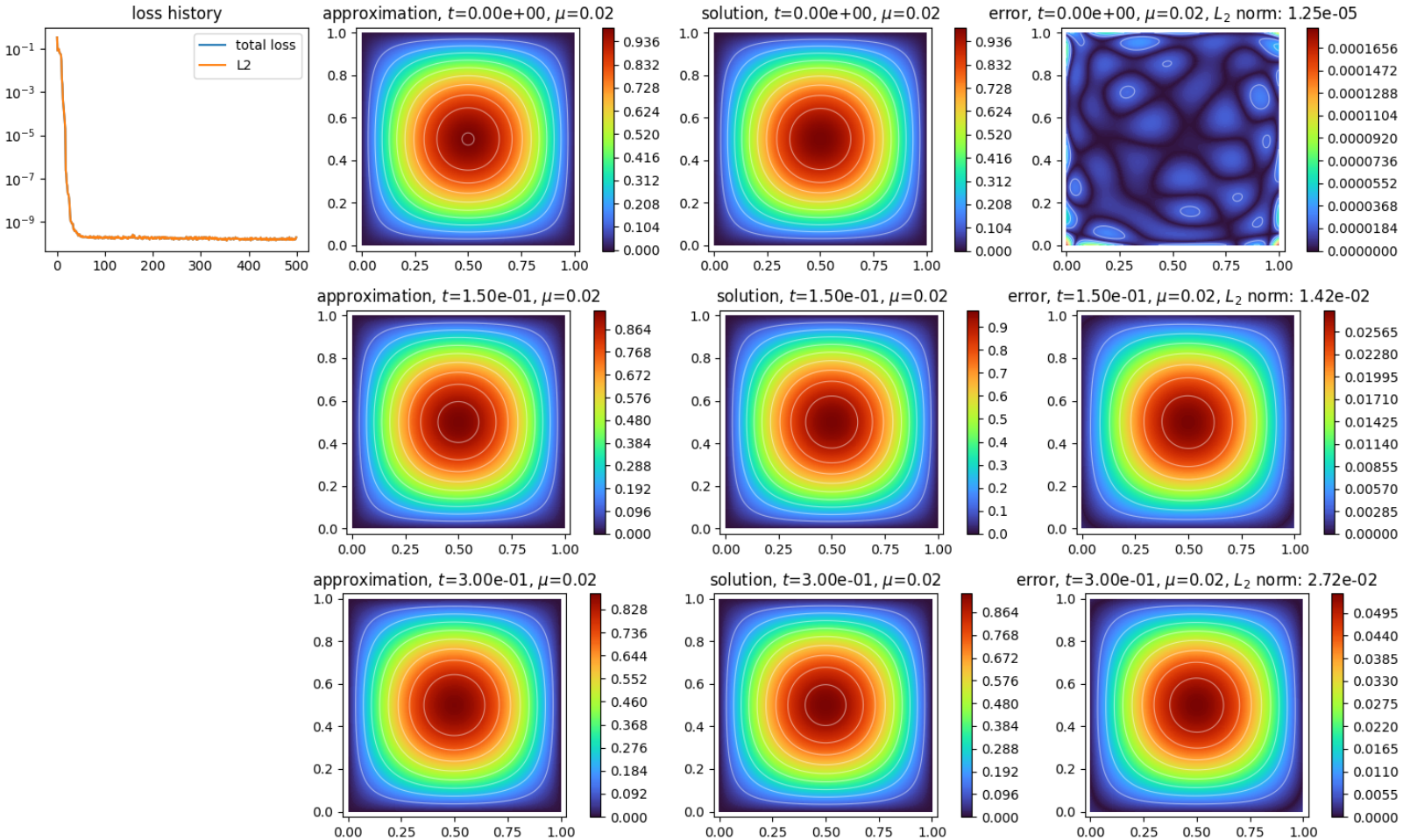

Figure: Results of Neural Galerkin Method

References

- M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations," Journal of Computational physics, 378:686–707, 2019.
- S. Chen, B. Shan, and Y. Li, "Efficient Discrete Physics-Informed Neural Networks for Addressing Evolutionary Partial Differential Equations," Preprint, December 22, 2023.
- J. Bruna, B. Peherstorfer, and E. Vanden-Eijnden, "Neural Galerkin Schemes with Active Learning for High-Dimensional Evolution Equations," Preprint, February 29, 2024.