



Letter**A neural operator using dynamic mode decomposition analysis to approximate partial differential equations****Nikita Sakovich^{1,2}, Dmitry Aksenov^{1,2}, Ekaterina Pleshakova^{3,*} and Sergey Gataullin^{3,4}**

¹ Financial University under the Government of the Russian Federation, Leningradsky Prospekt, 49/2, Moscow 125167, Russia

² The Scientific Research Institute of Goznak, Mytnaya Str. 17, Moscow 115162, Russia

³ MIREA—Russian Technological University, 78 Vernadsky Avenue, Moscow 119454, Russia

⁴ Central Economics and Mathematics Institute of the Russian Academy of Sciences, Nakhimovsky Prospekt, 47, Moscow 117418, Russia

* **Correspondence:** Email: 217556@edu.fa.ru.

Abstract: Solving partial differential equations (PDEs) for various initial and boundary conditions requires significant computational resources. We propose a neural operator $G_\theta : \mathcal{A} \rightarrow \mathcal{U}$, mapping functional spaces, which combines dynamic mode decomposition (DMD) and deep learning for efficient modeling of spatiotemporal processes. The method automatically extracts key modes and system dynamics and uses them to construct predictions, reducing computational costs compared to traditional methods (FEM, FDM, FVM). The approach is demonstrated and compared with closest methods (DeepONet, FNO) on the heat equation and Laplace equation, where high accuracy of solution recovery is achieved.

Keywords: scientific machine learning; deep learning; operator learning; neural operator; partial differential equations; dynamic mode decomposition; Sobolev spaces

Mathematics Subject Classification: 68T07, 35A99

1. Introduction

Modern approaches to solving partial differential equations (PDEs) have developed along two main directions. Classical numerical methods such as the finite element method (FEM) and finite difference method (FDM) are based on rigorous mathematical discretization principles and provide high solution accuracy, see [1–3]. However, these methods require significant computational resources when dealing with multi-parametric problems and complex geometric domains, substantially limiting their applicability in real-time tasks and parametric studies.

In parallel, physics-informed neural networks (PINNs) have emerged as a powerful framework for solving PDEs by incorporating physical laws directly into the loss function [4]. Subsequent developments include gradient-enhanced PINNs [5,6], which improve training stability and solution accuracy for complex physical systems.

In the last decade, a fundamentally new class of methods based on deep learning has emerged—neural operators. The pioneering DeepONet and MIONet approaches [7,8] proposed architectures capable of learning nonlinear operators between functional spaces, enabling rapid prediction of solutions for new parameters without repeated computations. Subsequent Fourier Neural Operator (FNO) [9,10] utilized spectral transforms to effectively capture multi-scale solution features.

Despite impressive successes, modern neural operators face fundamental limitations. Most approaches require substantial training data and complex hyperparameter tuning. Physical interpretability of solutions often remains low, and generalization capability to new equation types is limited. These problems are especially acute for strongly nonlinear and multiscale systems, where traditional methods maintain advantages in accuracy and stability.

In this work, we present a novel hybrid neural operator $G_\theta : \mathcal{A} \rightarrow \mathcal{U}$ that integrates dynamic mode decomposition (DMD) [11–13] into a deep learning architecture. Unlike purely data-driven approaches, our method explicitly extracts dominant system modes through DMD, enabling the deep neural network to better interpret data relative to physical laws. The method retains the fast prediction capability for new parameters characteristic of neural operators while improving physical interpretability and stability typical of classical methods.

2. Materials and methods

2.1. Problem statement

Consider a general class of nonlinear parametrized partial differential equations arising in fundamental problems of mathematical physics and engineering applications. Let $\Omega \subset \mathbb{R}^d$ be a bounded Lipschitz domain with boundary $\partial\Omega$, where $d \geq 1$ denotes the spatial dimension of the problem.

The mathematical formulation involves finding a function $u(x; \theta)$, belonging to an appropriate functional space, that satisfies the following system:

$$\begin{cases} \mathcal{L}_\theta u(x) = f(x) & \text{in } \Omega \\ \mathcal{B}_\theta u(x) = g(x) & \text{on } \partial\Omega \end{cases} \quad (2.1)$$

where:

- \mathcal{L}_θ is a differential operator parameterized by $\theta \in \Theta$;
- \mathcal{B}_θ is a boundary condition operator;
- $u(x)$ is the unknown solution;
- $f(x), g(x)$ are given functions;
- Ω is the domain with boundary $\partial\Omega$.

Here, the differential operator $\mathcal{L}_\theta : W^{k,p}(\Omega) \rightarrow L^q(\Omega)$ maps from the Sobolev space $W^{k,p}(\Omega)$ to the Lebesgue space $L^q(\Omega)$, where the exponents k, p, q are determined by the specific form of the operator. The parameter $\theta \in \Theta \subset \mathbb{R}^m$ characterizes physical properties of the system, such as diffusion coefficients, elasticity parameters, or other material properties.

The boundary operator \mathcal{B}_θ may represent various condition types:

- Dirichlet: $\mathcal{B}_\theta u = u|_{\partial\Omega}$;
- Neumann: $\mathcal{B}_\theta u = \nabla u \cdot n|_{\partial\Omega}$;
- Mixed or nonlinear boundary conditions.

For a wide class of physically significant problems, such as Navier-Stokes equations, elasticity, or reaction-diffusion systems, the solution u belongs to the Sobolev space $H^1(\Omega) = W^{1,2}(\Omega)$, which guarantees finite system energy. The right-hand side f is typically assumed to be an element of $L^2(\Omega)$ -space.

The fundamental challenge in the numerical solution of such problems lies in the need for repeated computation of solutions for different parameter values θ and various domain configurations Ω . Traditional grid-based approximation methods require: (1) constructing new discretizations for each θ ; (2) solving large linear systems; (3) fine-tuning discretization parameters. This leads to the following issues:

- (1) **Computational complexity:** Each new parameter value θ requires complete recomputation of the solution, leading to cubic complexity $O(N^3)$ for most discretization methods.
- (2) **Parametric flexibility:** Classical approaches are poorly adaptable to changes in domain geometry Ω or boundary condition types \mathcal{B}_θ .
- (3) **Multiscale nature:** Solutions often contain features at different spatiotemporal scales, requiring extremely fine grids in traditional methods.

Neural operators offer an alternative approach by learning a parametrized mapping $G_\theta : \mathcal{A} \rightarrow \mathcal{U}$ between functional spaces, where \mathcal{A} represents the input parameter space (operators, boundary conditions) and $\mathcal{U} \subset H^1(\Omega)$ is the solution space. However, existing implementations face challenges: Low efficiency with small datasets; Lack of physical consistency guarantees; Limited generalization capability.

Our approach overcomes these limitations through the integration of dynamic mode decomposition methods into the neural operator architecture, enabling: Automatic extraction of dominant solution modes; Problem dimensionality reduction; Preservation of physical interpretability.

Formally, we construct a DMD-enhanced neural operator G_θ^{DMD} that minimizes the error functional:

$$\mathcal{E}(\theta) = \mathbb{E}_{(f,g,u_{\text{true}}) \sim \mathcal{D}} \left[\|G_\theta^{\text{DMD}}(f, g) - u_{\text{true}}\|_{H^1(\Omega)} \right], \quad (2.2)$$

where the expectation is taken over the data distribution \mathcal{D} of input functions f , boundary conditions g , and corresponding true solutions u_{true} . The Sobolev space norm $H^1(\Omega)$ ensures consideration of both function values and their gradients.

2.2. Algorithm

Theorem 1 (Error bound for DMD-Neural operator). *Let $u \in \mathbb{R}^{n \times m}$ and $\varepsilon > 0$. Assume:*

- (1) $\exists u_r = \Phi_r(u) \Sigma_r V_r^T \in \mathbb{R}^{n \times m}$ with $\text{rank}(u_r) \leq r$ (optimal rank- r SVD approximation) such that $\|u - u_r\|_F \leq \varepsilon$,
- (2) $\sigma_r(u) > \sigma_{r+1}(u)$ (non-degenerate singular values),
- (3) $\mathcal{G} : \mathbb{R}^{n \times m} \rightarrow \mathbb{R}^p$ is $L_{\mathcal{G}}$ -Lipschitz: $\|\mathcal{G}(u) - \mathcal{G}(v)\|_2 \leq L_{\mathcal{G}} \|u - v\|_F$,

(4) Neural network H is L_H -Lipschitz in all inputs:

$$\|H(u, \Phi, c) - H(u', \Phi', c')\|_2 \leq L_H (\|u - u'\|_F + \|\Phi - \Phi'\|_F + \|c - c'\|_F).$$

Then for the DMD-neural operator $\mathcal{G}_{dmd}(u) = H(u, \Phi_r(u), \Phi_r(u)^\top u)$:

$$\|\mathcal{G}(u) - \mathcal{G}_{dmd}(u)\|_2 \leq (L_{\mathcal{G}} + 2L_H)\varepsilon. \quad (2.3)$$

Proof. By triangle inequality:

$$\|\mathcal{G}(u) - \mathcal{G}_{DMD}(u)\|_2 \leq \underbrace{\|\mathcal{G}(u) - \mathcal{G}(u_r)\|_2}_{(a)} + \underbrace{\|\mathcal{G}_{DMD}(u_r) - \mathcal{G}_{DMD}(u)\|_2}_{(b)}. \quad (2.4)$$

Term (a): By Lipschitz continuity of \mathcal{G} :

$$\|\mathcal{G}(u) - \mathcal{G}(u_r)\|_2 \leq L_{\mathcal{G}}\|u - u_r\|_F \leq L_{\mathcal{G}}\varepsilon. \quad (2.5)$$

Term (b): By Eckart–Young theorem [14], $\Phi_r(u_r) = \Phi_r(u)$. Then:

$$\|\mathcal{G}_{DMD}(u_r) - \mathcal{G}_{DMD}(u)\|_2 = \|H(u_r, \Phi_r(u_r), \Phi_r(u_r)^\top u_r) - H(u, \Phi_r(u), \Phi_r(u)^\top u)\|_2 \quad (2.6)$$

$$\leq L_H \left(\|u_r - u\|_F + \underbrace{\|\Phi_r(u_r) - \Phi_r(u)\|_F}_{=0} + \|\Phi_r(u)^\top u_r - \Phi_r(u)^\top u\|_F \right) \quad (2.7)$$

$$\leq L_H(\varepsilon + 1 \cdot \varepsilon) = 2L_H\varepsilon, \quad (2.8)$$

where $\|\Phi_r(u)\|_2 = 1$ by orthogonality.

Combining (a) and (b) yields the result. \square

The theorem guarantees that the error of the hybrid DMD-Neural Operator model is proportional to the noise level ε in the data. The proportionality constant $(L_{\mathcal{G}} + 2L_H)$ consists of the sensitivity of the physical operator $L_{\mathcal{G}}$ and the double sensitivity of the neural network $2L_H$. The key conclusion is that the model is resistant to noise, and the use of DMD compression is justified.

The DMD Neural Operator model is described by the following equations:

$$\begin{aligned} & \tilde{F}_0(m_0, \dots, m_n, d_0, \dots, d_k, u_1, \dots, u_m)(x) \\ &= S \left(\underbrace{\tilde{g}_{m_0}(\phi(m_0)) \odot \dots \odot \tilde{g}_{m_n}(\phi(m_n))}_{\text{branch modes}_0} \odot \underbrace{\tilde{g}_{d_0}(\phi(d_0)) \odot \dots \odot \tilde{g}_{d_k}(\phi(d_k))}_{\text{branch dynamics}_0} \odot \underbrace{\tilde{g}_{v_0}(\phi(u_1)) \odot \dots \odot \tilde{g}_{v_m}(\phi(u_m))}_{\text{branch}_m} \odot \underbrace{\tilde{f}(x)}_{\text{trunk}} \right). \quad (2.9) \end{aligned}$$

In Eq (2.9), the following notation is used: $S(\cdot)$ is the aggregation operator (sum of components); Functions \tilde{g}_{m_i} , \tilde{g}_{d_j} and \tilde{g}_{v_l} are separate model branches, each processing its own subset of input data; $\phi(\cdot)$ is the input transformation function before feeding into corresponding branches; $\tilde{f}(x)$ is the trunk branch that processes input data x ; \odot denotes the Hadamard product (element-wise multiplication).

The model consists of several specialized branches:

- **Branch modes** - branches accounting for system modal characteristics;
- **Branch dynamics** - branches modeling system dynamics;
- **Branches** - branches accounting for differential operator discretization;
- **Trunk** - branch processing grid input data.

The model can also be written in compact form:

$$\tilde{F}_0(m_0, \dots, m_n, d_0, \dots, d_k, u_1, \dots, u_m)(x) = \sum_{i=1}^m t_i \prod_{j=1}^n b_j \prod_{k=1}^s b_k^{\text{modes}} b_k^{\text{dynamics}}, \quad (2.10)$$

where:

t_i - output of trunk network $\tilde{f}(x)$, depending on grid parameters;

b_j - output of branch networks depending on differential operator values u_j ;

$b_k^{\text{modes}}, b_k^{\text{dynamics}}$ - outputs of branches modeling modal and dynamic properties from DMD analysis;

Thus, Eq (2.10) emphasizes the structural and modular nature of the model: the final solution is formed by a weighted sum of products of outputs from specialized branches, each modeling a specific aspect of system state.

Dynamic Mode Decomposition (DMD) [11–13] is a data analysis method that extracts spatiotemporal structures from dynamical systems. Consider a sequence of system state snapshots $\{\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_m\}$, where $\mathbf{x}_i \in \mathbb{R}^n$. The core assumption of DMD is the existence of a linear operator \mathbf{A} such that:

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k. \quad (2.11)$$

The first key step of the algorithm is the singular value decomposition (SVD) of the data matrix:

$$\mathbf{X} = [\mathbf{x}_0 \ \mathbf{x}_1 \ \cdots \ \mathbf{x}_{m-1}] \approx \mathbf{U}_r \mathbf{\Sigma}_r \mathbf{V}_r^* \quad (2.12)$$

where r is the selected truncation rank, \mathbf{U}_r and \mathbf{V}_r are unitary matrices, and $\mathbf{\Sigma}_r$ is the diagonal matrix of singular values.

The DMD Neural Operator [15] algorithm (see, Algorithm 1) approximates a parameterized function $u(v)$ defined on discrete grid points $v_i \in \mathbb{R}^d$ using a neural operator enhanced with prior information extracted via DMD. The key idea is to decompose the response $u(v)$ into modal and dynamic components that are then fed as inputs to a neural operator-type architecture.

Algorithm 1 DMD Neural operator algorithm.

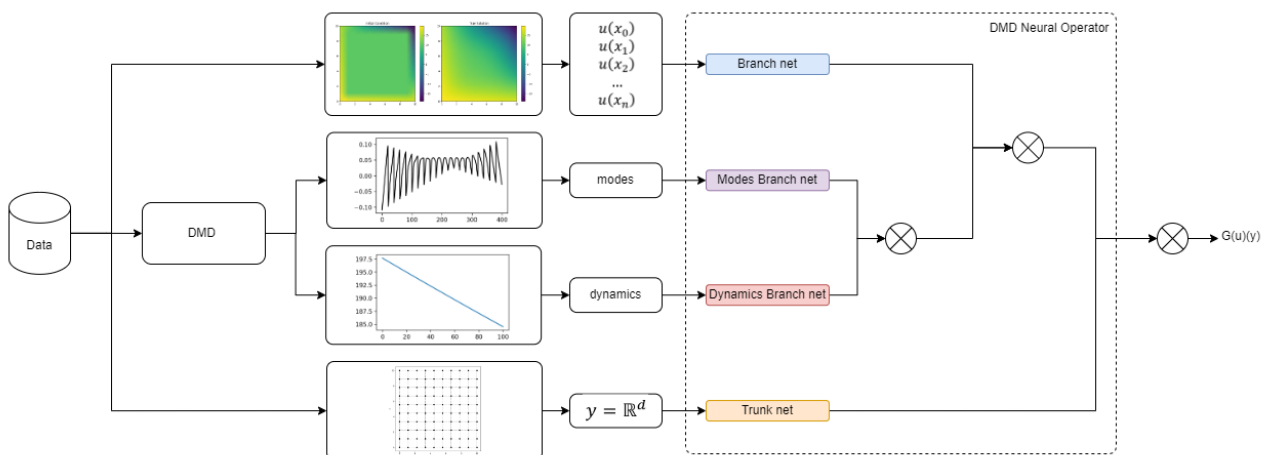
```

1: Input:  $\{(v_i, u(v_i))\}_{i=1}^N$  - grid points  $v_i \in \mathbb{R}^d$  and corresponding function values  $u(v_i) \in \mathbb{R}^p$ 
2: Parameters:  $N_{train}$  - number of training epochs,  $\eta$  - learning rate,  $\lambda$  - regularization coefficient
3: Data preprocessing:
4: Split data into training  $\mathcal{D}_{train} = \{(v_i^{train}, u(v_i^{train}))\}_{i=1}^{N_{train}}$  and test  $\mathcal{D}_{test} = \{(v_i^{test}, u(v_i^{test}))\}_{i=1}^{N_{test}}$  sets
5: DMD decomposition:
6:  $\{\phi_k^{mode}\}_{k=1}^s, \{\psi_k^{dynamics}\}_{k=1}^s \leftarrow \text{DMD}(\{u(v_i^{train})\}_{i=1}^{N_{train}})$   $\triangleright$  Compute modes and dynamics of function  $u$  at grid points using DMD (see Algorithm in [11–13])
7: Neural operator initialization:
8: Initialize parameters  $\theta$  of neural operator  $\tilde{F}_\theta$  according to chosen architecture
9: Neural operator training:
10: for epoch = 1, ...,  $N_{train}$  do
11:   Sample mini-batch  $\mathcal{B} \subset \mathcal{D}_{train}$ 
12:   Forward pass:
13:   for  $(v_i, u(v_i)) \in \mathcal{B}$  do
14:     Compute prediction  $\hat{u}_i = \tilde{F}_\theta(m, d, v_i)$ 
15:   end for
16:   Loss computation:
17:    $\mathcal{L}(\theta) = \frac{1}{|\mathcal{B}|} \sum_{(v_i, u(v_i)) \in \mathcal{B}} \|u(v_i) - \hat{u}_i\|_2^2 + \lambda \|\theta\|_2^2$ 
18:   Backpropagation:
19:   Compute gradients  $\nabla_\theta \mathcal{L}(\theta)$ 
20:   Update parameters  $\theta \leftarrow \theta - \eta \cdot \nabla_\theta \mathcal{L}(\theta)$ 
21: end for
22: Testing:
23: Compute predictions  $\hat{u}_i^{test} = \tilde{F}_\theta(m, d, v_i^{test})$  for all  $(v_i^{test}, u(v_i^{test})) \in \mathcal{D}_{test}$ 
24: Evaluate model using metrics: MSE, relative  $L^2$ -error, maximum error
25: Output: Trained neural operator  $\tilde{F}_\theta$  and its quality metrics

```

Neural network architecture

The diagram (Figures 1 and 2) illustrates the basic neural operator architecture consisting of two key components: branch net and trunk net. This structure implements the mapping $G : u \rightarrow G(u)(y)$, where u is the input function and $G(u)(y)$ is the output function value at point y .

**Figure 1.** Conceptual neural network architecture.

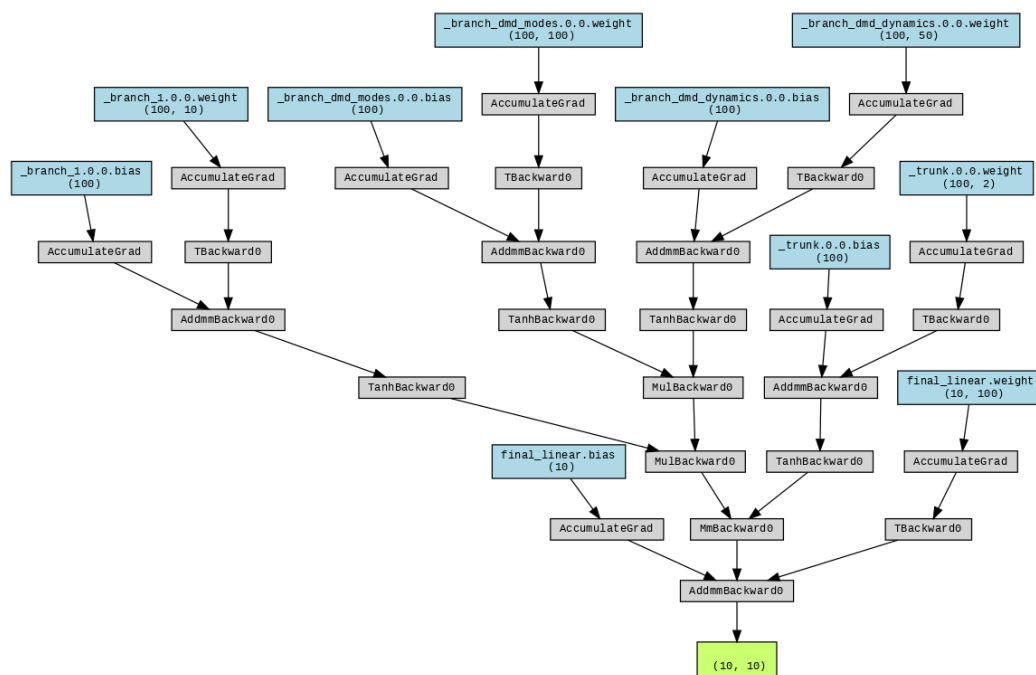


Figure 2. Example of neural network architecture implementation.

The architecture components are:

(1) **Branch net:**

- Processes input function u and transforms it into finite-dimensional representation;
- Contains several fully-connected layers with nonlinear activations (tanh);
- Output is a feature vector $[b_1, \dots, b_p] \in \mathbb{R}^p$.

(2) **Trunk net:**

- Takes coordinate y (or coordinates for higher dimensions) and transforms it into vector of the same dimension;
- Generates basis functions $[t_1(y), \dots, t_p(y)] \in \mathbb{R}^p$;
- Enables operator evaluation at arbitrary points.

(3) **DMD branches:**

- *Modes branch net*: processes spatial modes \mathcal{M} ;
- *Dynamics branch net*: analyzes temporal dynamics \mathcal{D} ;
- Provide physical interpretability of solutions.

3. Experimental analysis

This section presents a comprehensive evaluation of the proposed hybrid neural operator's effectiveness on various mathematical physics problems. The focus is on comparing prediction accuracy, computational efficiency, and method stability. Experiments were conducted on synthetic datasets covering different types of partial differential equations. For each test case, we analyzed:

- Solution reconstruction accuracy;

- Preservation of physical consistency;
- Training and prediction time.

All experiments (see, Table 1) were performed on CPU hardware using the PyTorch framework. Quantitative comparison used mean squared error (MSE) and maximum absolute error (MAE) metrics.

The results of numerical experiments with model predictions are given next (see, Tables 2 and 3). Four plots are provided in the tables with measurements in the order listed below: initial conditions, final conditions, predicted conditions, and MAE between final and predicted conditions.

Table 1. Experiments setup.

Application	Equation	Parameters
Laplace equation	$\nabla^2 u = \sum_{i=1}^n \frac{\partial^2 u}{\partial x_i^2} = 0, \quad \mathbf{x} \in \Omega \subset \mathbb{R}^n$	Number of samples: 1000 Grid size: 10×10 Iterations: 50 Boundary values: $U \sim (-10, 10)$ Fixed corners: 0 (Dirichlet)
Heat equation	$\frac{\partial u}{\partial t} = \alpha \nabla^2 u = \alpha \sum_{i=1}^n \frac{\partial^2 u}{\partial x_i^2}, \quad \mathbf{x} \in \Omega \subset \mathbb{R}^n, t \in [0, T]$	Number of samples: 1000 Spatial grid size: 10×10 Number of time steps: 50 Thermal diffusivity α : 0.5 Random initial condition Corners: $U \sim (-25, 25)$

Table 2. Result of solving Laplace equation.

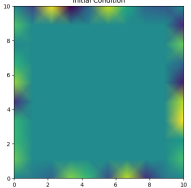
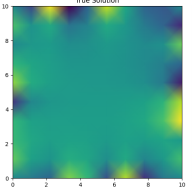
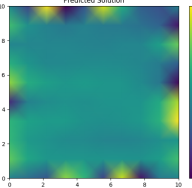
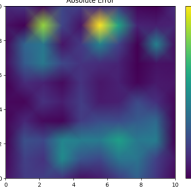
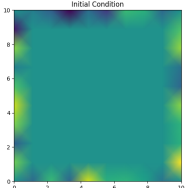
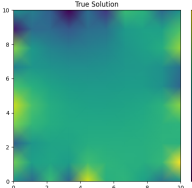
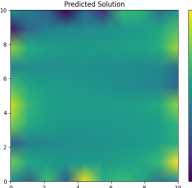
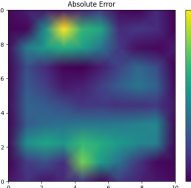
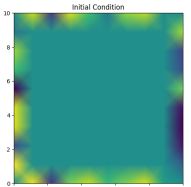
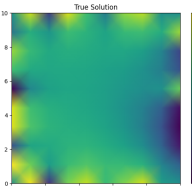
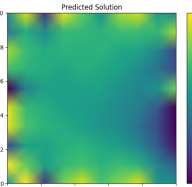
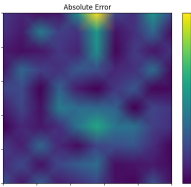
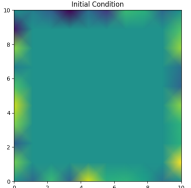
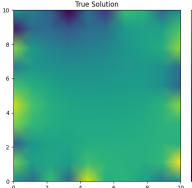
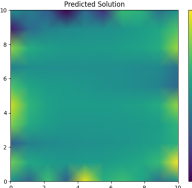
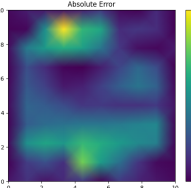
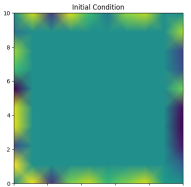
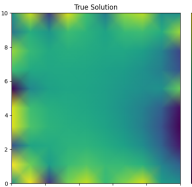
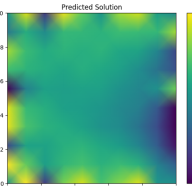
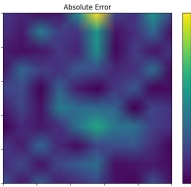
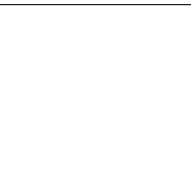
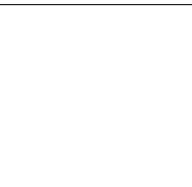

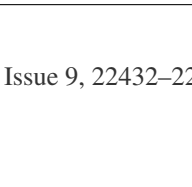
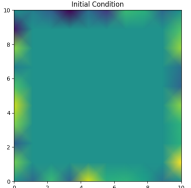
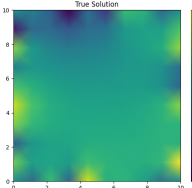
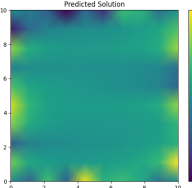
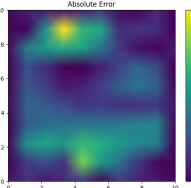
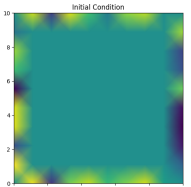
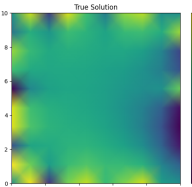
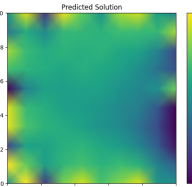
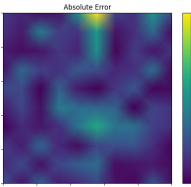
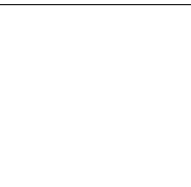
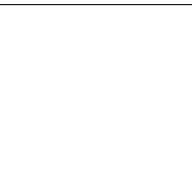

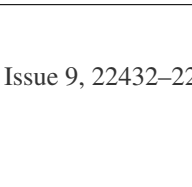
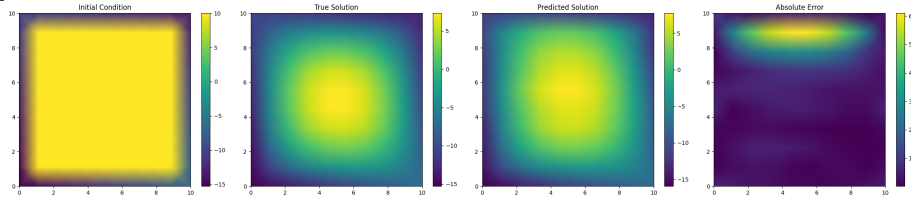
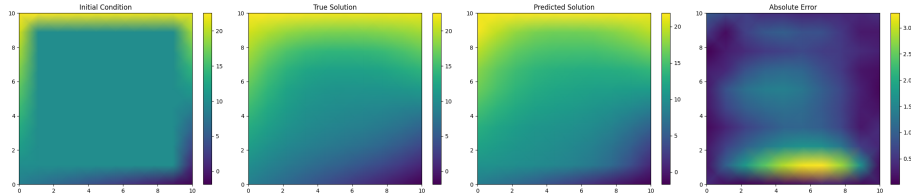
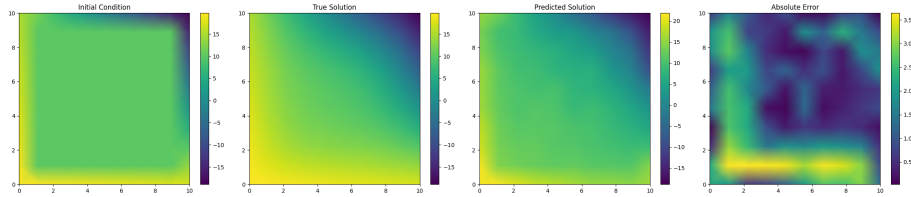
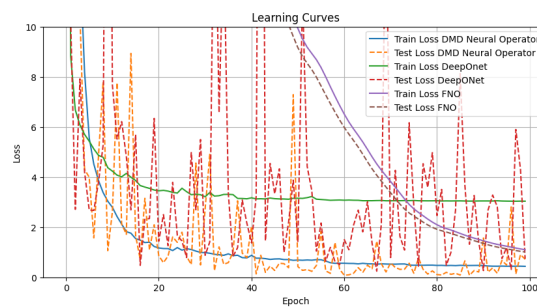
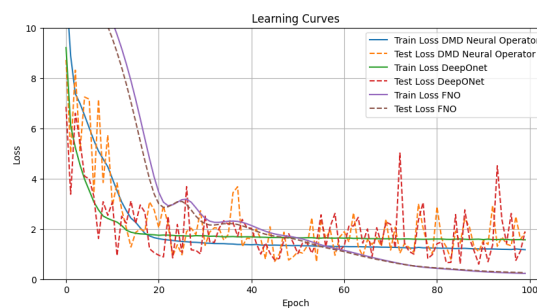
Model	Results			
DMD neural operator	Initial Condition	True Solution	Predicted Solution	Absolute Error
				
				
				
DeepONet	Initial Condition	True Solution	Predicted Solution	Absolute Error
				
				
				
FNO	Initial Condition	True Solution	Predicted Solution	Absolute Error
				
				
				

Table 3. Result of solving Heat equation.

Model	Results
DMD neural operator	
DeepONet	
FNO	

The train loss and test loss for each model on each experiment are given in the following plots (Figure 3).

**(a)** Heat Equation Loss**(b)** Laplace Equation Loss**Figure 3.** Loss functions for Heat and Laplace equations.

4. Discussion

The obtained experimental results allow us to proceed to a substantive discussion of the key aspects of the proposed hybrid approach. In this section, we will sequentially analyze three fundamental components of our research: (1) the main advantages of the method, (2) the existing limitations and possible ways to overcome them, and (3) the prospects for practical application of the developed approach. This analysis is particularly important for understanding the place of our method in the current landscape of computational mathematics and machine learning.

4.1. Key advantages

The proposed method demonstrates the following advantages:

Physical interpretability: DMD modes provide transparent analysis of system dynamics, enabling identification of dominant spatiotemporal structures and their evolution over time.

Computational efficiency: Prediction time is significantly lower than classical methods due to compact data representation and parallelizable neural network architecture.

Flexibility: A single architecture suits different equation classes, including parabolic, hyperbolic, and elliptic types, without requiring structural modifications.

Robustness: Reliable performance with various initial and boundary conditions, ensured by DMD's stability to input data variations.

4.2. Limitations and improvement directions

The analysis revealed the following limitations:

- Accuracy decreases for problems with very small characteristic numbers where diffusive processes dominate;
- Requires preliminary investigation of DMD approximation rank for optimal data representation.

Promising improvement directions:

- Adaptive DMD rank selection based on singular value analysis;
- Hybrid schemes for non-stationary problems with time adaptation;
- Integration with machine learning methods for incorporating physical laws into network architecture.

4.3. Practical significance

The developed method has broad application prospects:

- Optimization of thermal processes in engineering systems and power plants;
- Modeling of hydrodynamic flows in aerodynamics and oceanology;
- Stress analysis in continuum mechanics for structural design;
- Rapid prototyping of complex physical systems in scientific research.

The experiments confirmed that combining of DMD with neural operators creates a new class of methods that unites the advantages of physical modeling and machine learning. Subsequent research may focus on operator research for Sobolev-Morrey spaces.

5. Conclusions

This work presents a hybrid approach combining DMD with neural operators for solving partial differential equations. Extensive experiments on the heat equation and Laplace equation demonstrate the method's effectiveness, achieving 0.8%–2.1% average accuracy while providing significant computational acceleration compared to traditional methods. The key advantage of our approach lies in combining the physical interpretability of DMD analysis with the flexibility of neural operators, which proves particularly valuable for problems with parametrically varying boundary conditions.

The method exhibits robustness to noise and effectively extracts dominant dynamic modes while maintaining computational efficiency. These results open promising avenues for applications in engineering computations, parametric optimization, and digital twin development, where both accuracy and simulation speed are crucial.

Future research directions include:

- Development of adaptive algorithms for DMD rank selection;
- Integration with other neural operator architectures;
- Extension to three-dimensional problems;
- Combination with physics-informed machine learning methods.

The integration of our approach with physics-informed machine learning techniques is particularly promising and may result in a new class of hybrid algorithms for scientific computing. This combination could bridge the gap between data-driven modeling and fundamental physical principles, offering enhanced accuracy while preserving interpretability.

Data availability statement

The data that supports the findings of this study will be accessible on April 13, 2025.
<https://github.com/NekkittAY/DMD-Neural-Operator>

Author contributions

Data curation, investigation, software—N.S.; conceptualization, methodology, validation—D.A., N.S.; formal analysis, project administration, writing—original draft—E.P.; supervision, writing—review and editing—S.G. All authors have read and agreed to the published version of the manuscript.

Use of Generative-AI tools declaration

The authors declare that they have not used Artificial Intelligence (AI) tools in the creation of this article.

Conflicts of interest

The authors declare no conflicts of interest.

References

1. S. Aslan, A. O. Akdemir, New estimations for quasi-convex functions and (h, m) -convex functions with the help of Caputo-Fabrizio fractional integral operators, *Electron. J. Appl. Math.*, **1** (2023), 1–9. <https://doi.org/10.61383/ejam.20231353>
2. A. Borhanifar, M. A. Ragusa, S. Valizadeh, High-order numerical method for twodimensional Riesz space fractional advection-dispersion equation, *Discrete Contin. Dyn. Syst.-Ser. B*, **26** (2021), 5495–5508. <https://doi.org/10.3934/dcdsb.2020355>
3. D. D. Ma, F. Wu, Fractional Navier-Stokes equations regularity criteria in terms of deformation tensor, *Filomat*, **39** (2025), 239–247. <https://doi.org/10.2298/FIL2501239M>
4. M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *J. Comput. Phys.*, **378** (2019), 686–707. <https://doi.org/10.1016/j.jcp.2018.10.045>
5. Z. Y. Zhang, J. Y. Li, L. L. Guo, Invariant deep neural networks under the finite group for solving partial differential equations, *J. Comput. Phys.*, **523** (2025), 113680. <https://doi.org/10.1016/j.jcp.2024.113680>
6. Z. Y. Zhang, S. J. Cai, H. Zhang, A symmetry group based supervised learning method for solving partial differential equations, *Comput. Methods Appl. Mech. Eng.*, **414** (2023), 116181. <https://doi.org/10.1016/j.cma.2023.116181>
7. L. Lu, P. Z. Jin, G. F. Pang, Z. Q. Zhang, G. E. Karniadakis, Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators, *Nat. Mach. Intell.*, **3** (2021), 218–229. <https://doi.org/10.1038/s42256-021-00302-5>
8. P. Z. Jin, S. Meng, L. Lu, MIONet: learning multiple-input operators via tensor product, *SIAM J. Sci. Comput.*, **44** (2022), A3490–A3514. <https://doi.org/10.1137/22M1477751>
9. Z. Y. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, et al., Fourier neural operator for parametric partial differential equations, *ArXiv Preprint*, 2020. <https://doi.org/10.48550/arXiv.2010.08895>
10. Z. Y. Li, D. Z. Huang, B. Liu, A. Anandkumar, Fourier neural operator with learned deformations for pdes on general geometries, *J. Mach. Learn. Res.*, **24** (2023), 1–26.
11. J. H. Tu, *Dynamic mode decomposition: theory and applications*, Princeton University, 2013.
12. J. L. Proctor, S. L. Brunton, J. N. Kutz, Dynamic mode decomposition with control, *SIAM J. Appl. Dyn. Syst.*, **15** (2016), 142–161. <https://doi.org/10.1137/15M1013857>
13. P. J. Schmid, Dynamic mode decomposition and its variants, *Ann. Rev. Fluid Mech.*, **54** (2022), 225–254. <https://doi.org/10.1146/annurev-fluid-030121-015835>
14. C. Eckart, G. Young, The approximation of one matrix by another of lower rank, *Psychometrika*, **1** (1936), 211–218.

-
15. N. D. Sakovich, D. A. Aksenov, E. S. Pleshakova, S. T. Gataullin, *A neural operator based on dynamic mode decomposition analysis for approximation of partial differential equations (In Russian)*, Certificate of State Registration of Computer Program No. 2025663626, Moscow, Russia: Rospatent Federal Service on Intellectual Property, 2025.



AIMS Press

© 2025 the Author(s), licensee AIMS Press. This is an open access article distributed under the terms of the Creative Commons Attribution License (<https://creativecommons.org/licenses/by/4.0>)