ISSN 0027-1349, Moscow University Physics Bulletin, 2024, Vol. 79, Suppl. 2, pp. S706–S711. © Allerton Press, Inc., 2024.

Solving Problems of Mathematical Physics on Radial Basis Function Networks

D. A. Stenkin[1]* and V. I. Gorbachenko[1]

[1] *Penza State University, Penza, Russia*

Received October 1, 2024; revised October 10, 2024; accepted October 20, 2024

Abstract—The solution of boundary value problems described by partial differential equations on physics-informed neural networks is considered. Radial basis function networks are proposed as physics-informed neural networks. Such are easier to train compared to the fully connected networks usually used as physics-informed neural networks. An algorithm for solving the system of partial differential equations for the hydrodynamics problem is developed. On the example of the model problem of Kovasznay flow, programs for solving two-dimensional stationary Navier–Stokes equations using physics-informed radial basis function networks trained by the Nesterov method are developed.

Keywords: partial differential equations, physics-informed neural networks, radial basis function networks, Navier–Stokes equations, Kovasznay flow

DOI: 10.3103/S0027134924702163

1. INTRODUCTION

Modeling of objects with distributed parameters is reduced to solving boundary value problems for partial differential equations. Most problems of mathematical physics cannot be solved analytically. The solution by standard numerical methods, such as finite difference, finite element and finite volume methods, requires the construction of a mesh and the solution of systems of mesh equations of high dimensionality. Mesh construction is often quite difficult, especially for domains of complex configuration. The mesh equations are high dimensional and poorly conditioned, which is computationally expensive to solve. Meshless methods are less widespread, for example, the smoothed particle method is used in solving hydrodynamics problems, projection methods using radial basis functions are used in solving boundary value problems. In meshless methods the choice of kernel functions is not formalized, the smoothed particle method uses discrete models rather than traditional partial derivative equations.

Boundary value problems for partial differential equations can be successfully solved on neural networks [1, 2]. The theoretical basis for solving partial differential equations on neural networks is two provisions. First, neural networks are universal function approximators, which is based on the universal approximation theorem (D. Cybenko's theorem) and its development [3–5]. Second, when solving partial differential equations, the variational approach is used—the solution of the problem is found as a result of minimizing the loss function of the neural network, using the inconsistencies of the approximate solution formed by the network in some set of trial points inside, on the boundary of the solution domain and, possibly, in the points of additional conditions. The trial points are usually randomly placed (although it is possible to use a grid). Therefore, the solution on neural networks is a meshless approximate analytical solution of partial differential equations, since the solution to the problem is a function (which is analytically almost impossible to represent due to complexity) determined by the architecture and parameters of the neural network.

With the development of machine learning and neural networks, interest in the use of neural network technologies for solving boundary value problems has increased. This has been facilitated by the widespread use of freely available machine learning libraries that implement automatic (algorithmic) differentiation and deep neural network training algorithms. The neural network approach seems to be particularly effective in comparison with traditional methods for multivariate and/or nonlinear partial derivative equations and inverse boundary value

*E-mail: stynukin@mail.ru

problems, when it is necessary to determine the parameters of the mathematical model based on the experimental results. A new scientific direction has been formed-physics-informed neural networks [6–8]. Physics-informed neural networks incorporate a mathematical model of a physical phenomenon, for example, in the form of partial derivative equations into the network structure and do not require examples with known values for training when solving direct problems. They solve the problem of low data availability in many scientific and engineering problems where traditional machine learning methods are ineffective. Additional conditions, such as experimental information about the solution, can also be used. In such networks, the mathematical model is a regulating factor that improves the quality of the solution.

Integrating physical law verification into the architecture and training process of neural networks allows physics-informed neural networks to solve complex physical problems using both data and underlying physical principles. Physics-informed neural networks approximate the solution of a boundary value problem by minimizing the residual of the equation at collocation points within and at the boundary of the solution domain. Estimation of the norm of this residual gives a more explicit estimate of the accuracy of the solution of the problem than estimates of the accuracy of the solution of mesh methods, for which only the order of accuracy of the approximation and the residual of the solution of the system of mesh equations are known. This feature of physics-informed neural networks, as well as the incorporation of mathematical models into the architecture of physics-informed neural networks, facilitates the interpretability of solution results. When solving forward problems, training data are usually not used, so physics-informed neural networks are not retrained in such situations. But when solving inverse problems, retraining is possible. Physics-informed neural networks are meshless and after training allow to find a solution in arbitrary points of the domain.

Physics-informed neural networks can be constructed using neural networks of different architectures. Most commonly used are fully connected neural networks implemented using freely available libraries, such as TensorFlow and PyTorch, which implement automatic differentiation. Networks of this architecture face the challenges of efficiently integrating physical laws, ensuring that the network architecture is capable of capturing the necessary physical details, and training difficulties due to the use of optimizers not designed to train physics-informed neural networks. A physics-informed neural network needs to be trained for each task, which requires a large amount of computational power and training time. In addition, the existing optimizers in the libraries are focused on solving pattern recognition problems rather than on solving computational problems.

The use of radial basis function networks as physics-informed neural network is promising [2, 9–11]. Radial basis function networks are simpler than fully connected ones, they contain only two layers: the first layer is a layer of radial basis functions [12] and a linear layer. Such networks are easier to train. For such networks, gradient learning algorithms of the second order have been developed [10, 13], in which not only the weights of the network are adjusted, but also the parameters of the radial basis functions. The authors demonstrated the effectiveness of solving direct [14, 15] and inverse [16, 18] boundary value problems on radial basis function networks for piecewise homogeneous media. Radial basis function networks designed to solve partial differential equations are becoming popular. A new type of physically informed neural networks is being formed—physics-informed radial basis function networks [17–20].

The interest in the study of physics-informed radial basis function networks is also due to the fact that radial basis function networks are a type [21] of a new type of neural networks—Kolmogorov–Arnold neural networks [22], promising as physics-informed neural networks [23].

An important class of partial differential equations is the Navier–Stokes equations, which are the most important ones in hydrodynamics and are used in modeling many natural phenomena and technical problems. There are known works devoted to the solution of the Navier–Stokes equations on fully connected physics-informed neural networks [24–28]. Intel Labs, a research division of Intel, has integrated the computational fluid dynamics fully connected physics-informed neural network into the OpenFOAM framework for solving computational hydrodynamics problems [29]. Such integration has shown high efficiency in solving hydrodynamics problems, especially ill-conditioned inverse problems.

The purpose of this paper is to develop and experimentally investigate an algorithm for solving the system of Navier–Stokes differential equations on physics-informed radial basis function networks. The study is carried out on the example of a model problem for the Kovasznay flow, which has an analytical solution.

2. SOLVING A SYSTEM OF DIFFERENTIAL EQUATIONS FOR THE KOVASZNAY FLOW

The output of the radial basis function network is described by the expression

$$v(x) = \sum_{k=1}^{n_{\text{RBF}}} \omega_k \varphi_k(x),$$

where n_{RBF} is the number of radial basis functions (or number of neurons), ω_k is the weight of the kth neuron, and $\varphi_k(x)$ is the value of the kth radial basis function at point x.

The boundary value problem in operator form has the form

$$Lu(\mathbf{x}) = f(\mathbf{x}), \quad x \in \Omega,$$

$$Bu(\mathbf{x}) = p(\mathbf{x}), \quad x \in \partial\Omega,$$

where u is the desired solution, L is the differential operator, operator B sets the boundary conditions, Ω is the solution area, $\partial\Omega$ is the regional boundary, and f and p are the known functions.

Let us consider the Kovasznay model flow [30]. The Kovasznay flow is a two-dimensional stationary flow. Two-dimensional stationary Navier–Stokes equations for an incompressible medium have the form:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0, \quad (x, y) \in \Omega,$$

$$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{\partial p}{\partial x} + \frac{1}{\text{Re}}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right), \quad (x, y) \in \Omega,$$

$$u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{\partial p}{\partial y} + \frac{1}{\text{Re}}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right), \quad (x, y) \in \Omega,$$

where Ω is the calculated area, u is the first velocity component, v is the second velocity component, and p is the pressure. Dirichlet conditions are set at the boundary of the region.

The Kovasznay flow is the movement of the flow through the lattice. It does not have an initial condition. Due to the well-known analytical solution, this model is convenient for testing the effectiveness of numerical methods for solving differential equations.

The analytical solution has the form [30]

$$u(x, y) = -e^{(-\lambda x)}\cos(2\pi y),$$

$$v(x, y) = -\frac{\lambda}{2\pi}e^{(-\lambda x)}\sin(2\pi y),$$

$$p(x, y) = -\frac{1}{2}e^{(-2\lambda x)},$$

where parameter λ is defined by the following expression

$$\lambda = \sqrt{\frac{\text{Re}^2}{4} + 4\pi^2} - \frac{\text{Re}}{2},$$

where Re is the Reynolds number.

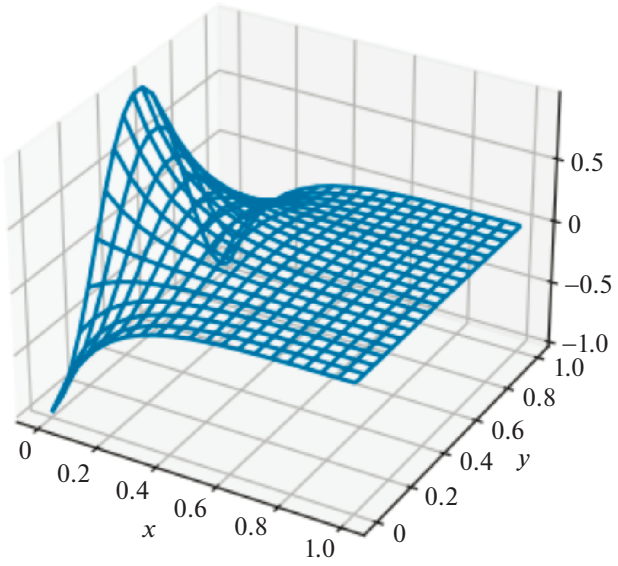

Fig. 1. Analytical solution for the first velocity component.

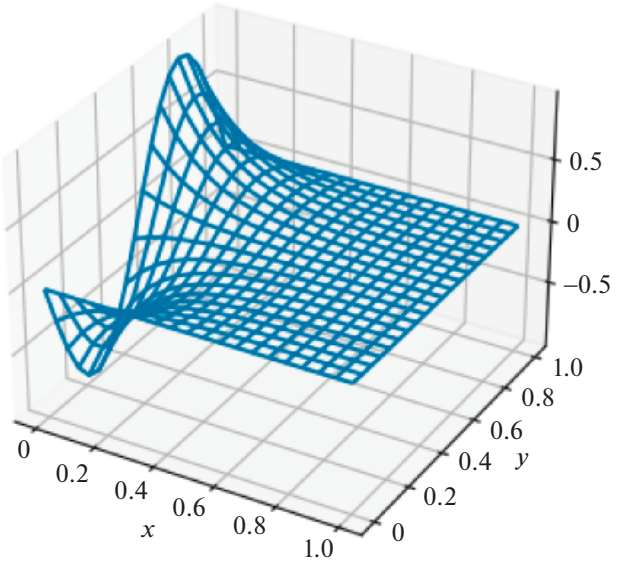

Fig. 2. Analytical solution for the second velocity component.

If the liquid is incompressible [31], then

$$\rho = \text{const},$$

where ρ is the density of the liquid.

If the movement is stationary, then

$$\frac{\partial V}{\partial t} = 0.$$

For this model problem, the solution area is a square with dimensions $x \in [0; 1], y \in [0; 1]$.

The solution algorithm is based on the results presented in article [20]. The components of the gradient

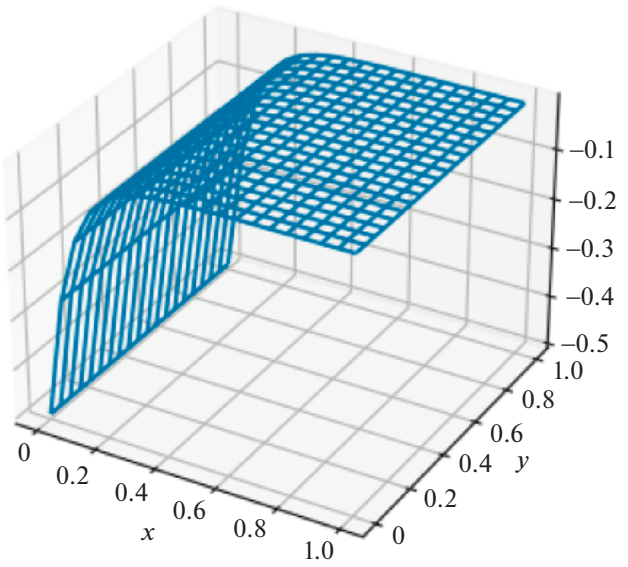

Fig. 3. Analytical solution for pressure.

of the functional in terms of weights were obtained using automatic differentiation implemented in the freely available TensorFlow library of the Python language. Three physics-informed radial basis function networks were applied to the solution. Each network approximates one of the three variables included in the system of Navier–Stokes differential equations. The networks are connected by a common mean-squared error functional. The functional is the sum of the squares of the residuals. The $\frac{1}{2}$ multiplier is introduced to simplify the expression for the gradient of the functional. The condition for the end of the training process of the networks was a small value of the common mean-square error functional. The error functional has the form

$$\frac{1}{2}\sum_{i=1}^{N}\left(u_i\frac{\partial u_i}{\partial x}+v_i\frac{\partial u_i}{\partial y}+\frac{\partial p_i}{\partial x}-\frac{1}{\mathrm{Re}}\left(\frac{\partial^2 u_i}{\partial x^2}+\frac{\partial^2 u_i}{\partial y^2}\right)\right)^2+\frac{1}{2}\sum_{i=1}^{N}\left(u_i\frac{\partial v_i}{\partial x}+v_i\frac{\partial v_i}{\partial y}+\frac{\partial p_i}{\partial y}-\frac{1}{\mathrm{Re}}\left(\frac{\partial^2 v_i}{\partial x^2}+\frac{\partial^2 v_i}{\partial y^2}\right)\right)^2+\frac{1}{2}\sum_{i=1}^{N}\left(\frac{\partial u_i}{\partial x}+\frac{\partial v_i}{\partial y}\right)^2+\frac{\lambda}{2}\sum_{j=1}^{K}(u_j-u_j^A)^2+\frac{\lambda}{2}\sum_{j=1}^{K}(v_j-v_j^A)^2+\frac{\lambda}{2}\sum_{j=1}^{K}(p_j-p_j^A)^2,$$

where Re is the Reynolds number, N is the number of internal test points, K is the number of boundary test points, λ is the penalty factor, and u_j^A, v_j^A, p_j^A is the analytical value of the jth test point at the boundary of the region.

The centers of the radial basis functions of each network were located on a uniform grid. The weight and width vectors were initialized with zero values. Experiments were conducted for each network to select the optimal number of neurons and trial points. The number of neurons for each physics-informed

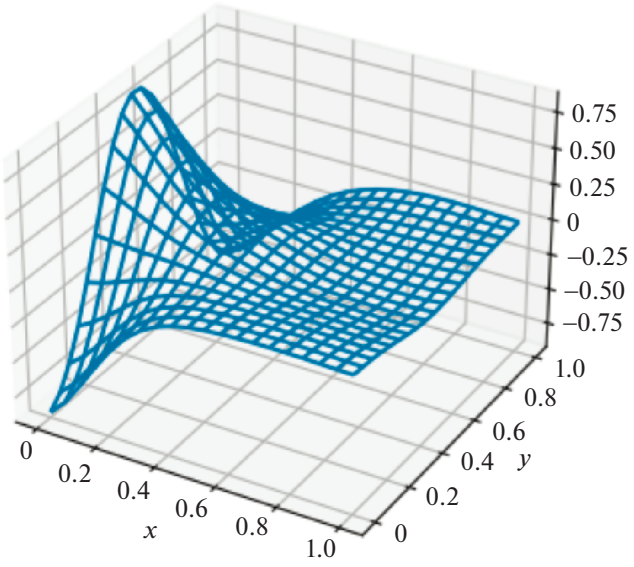

Fig. 4. The first component of speed resulting from network operation.

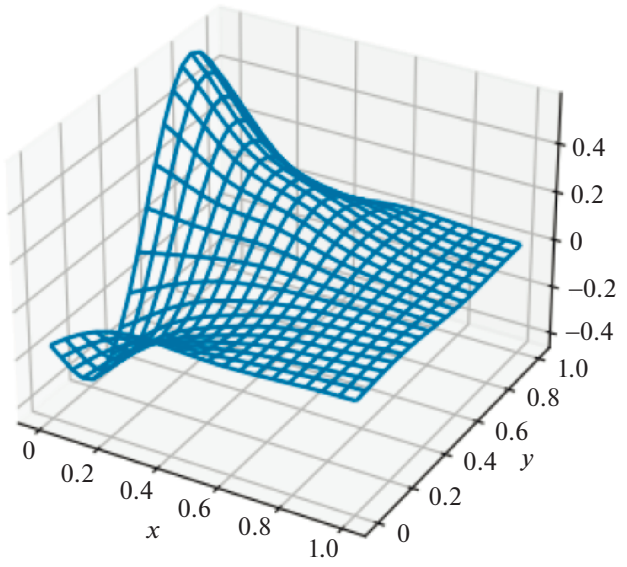

Fig. 5. The second component of speed obtained as a result of network operation.

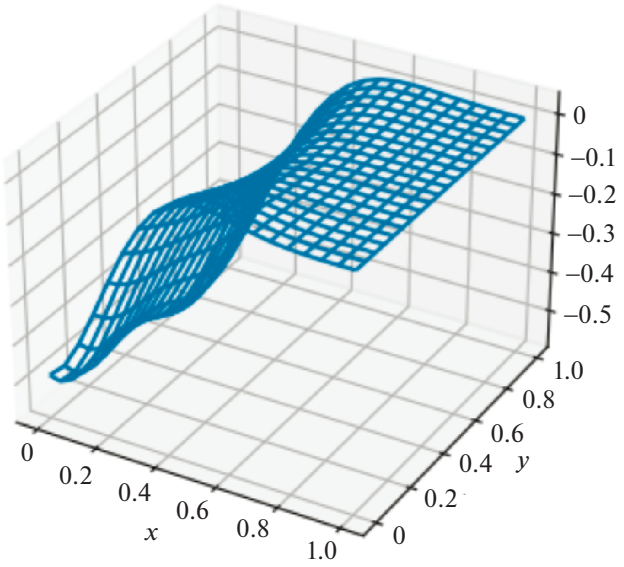

Fig. 6. Pressure obtained as a result of network operation.

radial basis function networks is 64. The number of interior trial points for each network is 100. The number of boundary trial points for each network is 40. The coordinates of collocation points were generated as random numbers that were uniformly distributed in the solution domain. Only the weights were customizable. The value of the error functional equal to 10^{-2} is achieved in 15 iterations on average.

Network training consists in minimizing the error functionality. Batch training is used, since in each iteration of training, the total error for all trial points is minimized. The Nesterov method is used for training. The Nesterov method [32–35] uses the idea of the momentum accumulation method. Additionally, the history of parameter changes is used. The correction to the parameter vector in the Nesterov method is described as follows

$$\triangle\Theta^{(k+1)} = \alpha\triangle\Theta^{(k)} - \gamma \mathrm{grad} I(\Theta^{(k)} - \alpha\triangle\Theta^{(k)}),$$

where Θ is the vector of one of the network parameters (we can consider the vector of all parameters), γ is the selected numerical coefficient (learning rate), and α is the moment coefficient, which takes values in the interval $[-0.5; 1.5]$.

Plots of analytical solutions for velocity and pressure components are presented in Figs. 1–3. The graphs obtained as a result of the network operation are shown in Figs. 4–6. They have a visual similarity to the analytical solution. The MSE values were obtained in about 30 iterations compared to the analytical solution. The MSE for the first component of the velocity is on average 10^{-6}. The MSE for the second component of the velocity is on average 10^{-6}. The MSE for pressure averages 10^{-4}. This is a testament to the effectiveness of the program.

This problem was solved on a fully connected network [27]. Physics-informed radial basis function networks are as accurate as fully connected networks, but they are easier and faster to train.

3. CONCLUSIONS

Based on the approach proposed by the authors in the article [20], an algorithm for solving a system of differential equations for the problem of hydrodynamics was developed using the example of the Kovasznay flow model problem, a program for solving two-dimensional stationary Navier–Stokes equations using physics-informed radial basis function networks trained by the Nesterov method was developed. Our method allows achieving accuracy satisfactory for engineering applications in fewer iterations.

Further development of this work involves the improvement of learning algorithms for physics-informed radial basis function networks and the solution of various differential equations and systems of equations.

FUNDING

This work was supported by ongoing institutional funding. No additional grants to carry out or direct this particular research were obtained.

CONFLICT OF INTEREST

The authors of this work declare that they have no conflicts of interest.

REFERENCES

1. I. E. Lagaris, A. Likas, and D. I. Fotiadis, IEEE Trans. Neural Networks **9**, 987 (1998). https://doi.org/10.1109/72.712178
2. N. Yadav, A. Yadav, and M. Kumar, *An Introduction to Neural Network Methods for Differential Equations*, SpringerBriefs in Applied Sciences and Technology (Springer, Dordrecht, 2015). https://doi.org/10.1007/978-94-017-9816-7
3. G. Cybenko, Math. Control, Signals Syst. **2**, 303 (1989). https://doi.org/10.1007/BF02551274
4. K. Hornik, Neural Networks **4**, 251 (1999). https://doi.org/10.1016/0893-6080(91)90009-t
5. B. Hanin, Mathematics **7**, 992 (2019). https://doi.org/10.3390/math7100992
6. M. Raissi, P. Perdikaris, and G. E. Karniadakis, J. Comput. Phys. **378**, 686 (2019). https://doi.org/10.1016/j.jcp.2018.10.045

7. L. Lu, X. Meng, Zh. Mao, and G. E. Karniadakis, SIAM Rev. **63**, 208 (2021). https://doi.org/10.1137/19m1274067
8. S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli, J. Sci. Comput. **92**, 88 (2022). https://doi.org/10.1007/s10915-022-01939-z
9. A. A. Ramabathiran and P. Ramachandran, J. Comput. Phys. **445**, 110600 (2021). https://doi.org/10.1016/j.jcp.2021.110600
10. V. I. Gorbachenko and M. V. Zhukov, Comput. Math. Math. Phys. **57**, 145 (2017). https://doi.org/10.1134/s0965542517010079
11. A. Hryniowski and A. Wong, arXiv Preprint (2019). https://doi.org/10.48550/arXiv.1911.09257
12. M. D. Buhmann, *Radial Basis Functions: Theory and Implementations* (Cambridge University Press, Cambridge, 2004).
13. M. Alqezweeni and V. Gorbachenko, in *Proceedings of International Scientific Conference on Telecommunications, Computing and Control*, Ed. by N. Voinov, T. Schreck, and S. Khan, Smart Innovation, Systems and Technologies, Vol. 220 (Springer, Singapore, 2021), p. 475. https://doi.org/10.1007/978-981-33-6632-9_42
14. V. I. Gorbachenko and D. A. Stenkin, in *Advances in Neural Computation, Machine Learning, and Cognitive Research IV. NEUROINFORMATICS 2020*, Ed. by B. Kryzhanovsky, W. Dunin-Barkowski, V. Redko, and Y. Tiumentsev, Studies in Computational Intelligence, Vol. 925 (Springer, Cham, 2021), p. 412. https://doi.org/10.1007/978-3-030-60577-3_49
15. V. I. Gorbachenko and D. A. Stenkin, in *7th International Conference on Contemporary Information Technology and Mathematics* (2021), p. 267.
16. V. I. Gorbachenko and D. A. Stenkin, in *Advances in Neural Computation, Machine Learning, and Cognitive Research V*, Ed. by B. Kryzhanovsky, W. Dunin-Barkowski, V. Redko, Y. Tiumentsev, and V. V. Klimov, Studies in Computational Intelligence, Vol. 1008 (Springer, Cham, 2021), p. 230. https://doi.org/10.1007/978-3-030-91581-0_31
17. V. I. Gorbachenko and D. A. Stenkin, in *Cyber-Physical Systems and Control II*, Ed. by D. G. Arseniev and N. Aouf, Lecture Notes in Networks and Systems, Vol. 460 (Springer, Cham, 2021), p. 3. https://doi.org/10.1007/978-3-031-20875-1_1
18. V. I. Gorbachenko and D. A. Stenkin, Tech. Phys. **68**, 151 (2023). https://doi.org/10.1134/s1063784223050018
19. J. Bai, G.-R. Liu, A. Gupta, L. Alzubaidi, X.-Q. Feng, and Yu. Gu, Comput. Methods Appl. Mech. Eng. **415**, 116290 (2023). https://doi.org/10.1016/j.cma.2023.116290
20. D. Stenkin and V. Gorbachenko, Mathematics **12**, 241 (2024). https://doi.org/10.3390/math12020241
21. Z. Li, arXiv Preprint (2024). https://doi.org/10.32657/10356/4626
22. Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljacic, T. Y. Hou, and M. Tegmark, arXiv Preprint (2024). https://doi.org/10.48550/arXiv.2404.19756
23. Y. Wang, J. Sun, J. Bai, C. Anitescu, M. S. Eshaghi, X. Zhuang, T. Rabczuk, and Yi. Liu, arXiv Preprint (2024). https://doi.org/10.48550/arXiv.2406.11045
24. PINN (physics-informed neural networks) on Navier–Stokes equations. https://github.com/chenyingfa/pinn-torch.
25. B. Hu and D. Mcdaniel, Mathematical and Computational Applications **28**, 102 (2023). https://doi.org/10.3390/mca28050102
26. A. Farkane, M. Ghogho, M. Oudani, and M. Boutayeb, Int. J. Numer. Methods Fluids **96**, 381 (2024). https://doi.org/10.1002/fld.5250
27. K. B. Koshelev and S. V. Strijhak, Trudy Instituta Sistemnogo Programmirovaniya Rossiiskoi Akademii Nauk **35**, 245 (2023). https://doi.org/10.15514/ispras-2022-35(5)-16
28. N. Mai-duy and T. Tran-cong, Int. J. Numer. Methods Fluids **37**, 65 (2001). https://doi.org/10.1002/fld.165
29. R. Farber, AI produces data-driven OpenFOAM speedup. https://www.hpcwire.com/2024/07/23/ai-produces-data-driven-openfoam-speedup/.
30. L. S. G. Kovasznay, Math. Proc. Cambridge Philos. Soc. **44**, 58 (1948). https://doi.org/10.1017/s0305004100023999
31. N. A. Slezkin, *Dynamics of Viscous Incompressible Fluid* (Gosudarstvennoe Izdatel'stvo Tekhniko-Teoreticheskoi Literatury, Moscow, 1955).
32. I. Sutskever, J. Martens, and G. Dahl, in *ICML'13 Proceedings of the 30th International Conference on International Conference on Machine Learning* (2013), p. 28.
33. Yu. E. Nesterov, *Introduction to Convex Optimization* (Moskovskii Tsentr Nepreryvnogo Matematicheskogo Obrazovaniya, Moscow, 2010).
34. P. Sadovnikov, Methods for optimizing neural networks. https://habrahabr.ru/post/318970/.
35. S. Ruder, arXiv Preprint (2016). https://doi.org/10.48550/arXiv.1609.04747