UDC 004.93

SOLVING POISSON EQUATION WITH CONVOLUTIONAL NEURAL NETWORKS

Kuzmych V. A. – PhD, Student of Department of Computer Engineering, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine.

Novotarskyi M. A. – Dr. Sc., Professor of Department of Computer Engineering, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine.

Nesterenko O. B. – PhD, Head of the Department of Applied Physics and Higher Mathematics, Kyiv National University of Technologies and Design, Kyiv, Ukraine.

ABSTRACT

Context. The Poisson equation is the one of fundamental differential equations, which used to simulate complex physical processes, such as fluid motion, heat transfer problems, electrodynamics, etc. Existing methods for solving boundary value problems based on the Poisson equation require an increase in computational time to achieve high accuracy. The proposed method allows solving the boundary value problem with significant acceleration under the condition of acceptable loss of accuracy.

Objective. The aim of our work is to develop artificial neural network architecture for solving a boundary value problem based on the Poisson equation with arbitrary Dirichlet and Neumann boundary conditions.

Method. The method of solving boundary value problems based on the Poisson equation using convolutional neural network is proposed. The network architecture, structure of input and output data are developed. In addition, the method of training dataset generation is described.

Results. The performance of the developed artificial neural network is compared with the performance of the numerical finite difference method for solving the boundary value problem. The results showed an acceleration of the computational speed in x10–700 times depending on the number of sampling nodes.

Conclusions. The proposed method significantly accelerated speed of solving a boundary value problem based on the Poisson equation in comparison with the numerical method. In addition, the developed approach to the design of neural network architecture allows to improve the proposed method to achieve higher accuracy in modeling the process of pressure distribution in areas of arbitrary size.

KEYWORDS: machine learning, Poisson equation, convolutional neural network.

ABBREVIATIONS

CFD is a computational fluid dynamics;
ANN is an artificial neural network;
FPN is a Feature Pyramid Network;
MSE is mean squared error;
GPU is a graphics-processing unit;
PDE is a partial differential equation;
RHS is a right hand side;
BC is a boundary condition.

NOMENCLATURE

Δ is the Laplace operator;

$F(x,y)$ is the right-hand side function;

$u(x,y)$ is the unknown function;

$u_{i,j}$ is the discrete value of the unknown function at the node with (i,j) coordinates;

h is an area sampling step;

n,m are the number of steps in x and y coordinates, respectively;

φ_i is the Dirichlet boundary condition.

INTRODUCTION

Traditionally, we use CFD modeling to determine the distribution of pressure and other parameters of the movement of liquids or gases. Finite Difference Method, Finite Element Method and Finite Volume Method are key techniques for solving aerodynamic problems, weather forecasting, life sciences and many other fields.

However, all these methods have a number of significant disadvantages. The main common disadvantage of CFD methods is a significant increase in the amount of computation and memory used with increasing number of sampling nodes or domain size.

When using the finite differences method, the solution of such a system can be simplified by using the fact that in this case we obtain a three-diagonal matrix. This structure of the matrix allows the use of parallel calculations, which significantly reduces the time to solve the problem. However, this approach makes sense to apply in the case of an area with simple geometry, because the finite difference method may lose convergence.

These factors interfere the widespread use of numerical methods in real-time applications and stimulate active research into alternative methods for solving PDEs to overcome these limitations.

The use of ANN has great prospects for solving these problems. This opportunity appeared through significant progress in the field of deep machine learning. The use of ANN of various types to solve boundary value problems is developing rapidly. This progress is based on a real opportunity to overcome those objective limitations that numerical methods have.

One of the fields of study, where deep learning algorithms can be applied is a biomedical engineering. Accuracy of modelling of biological objects has less crucial role, than speed of modelling. Moreover, GPU-accelerated neural networks can be a new efficient solution for many biomedical problems.

DOI 10.15588/1607-3274-2022-1-6

Example of such problem is the mathematical models of anastomoses of human stomach. Determination of the pressure field in the zone, which can be described by the Poisson equation, of the reconstruction-recovery operation in the real time, doesn't require extremely high precision. That's why we consider application of artificial neural networks as alternative to numerical methods to solve mentioned problem.

The object of study is the process of computational hydrodynamics.

The subject of study is the methods and means of using ANN to solve the Poisson equation.

The aim of this work is to develop and train an artificial neural network for modeling pressure field changes in areas with complex geometry and boundary conditions, which allows to reduce the simulation time with acceptable accuracy.

1 PROBLEM STATEMENT

Poisson equation is an elliptic partial differential equation. The solution of this equation is presented as a boundary value problem in a rectangular domain Ω with parameters $(x,y)\in[0,n]\times[0,m]$. This problem can be mathematically represented by next formulas:

$$\begin{cases} \Delta u(x,y)=F(x,y), \\ u(x,0)=\varphi_1(x), \\ u(0,y)=\varphi_2(y), \\ u(x,m)=\varphi_3(x), \\ u(n,y)=\varphi_4(y). \end{cases} \qquad (1)$$

To construct a difference scheme for the boundary value problem (1), we introduce a uniform grid: $\omega_h=\{x_i=ih, y_j=jh, i=\overline{1,n}, j=\overline{jm}\}$, where h=1.

After discretization of the boundary value problem (1), using the symbolic notation of difference operators, we obtain the difference boundary value problem on the ω_h grid:

$$\begin{cases} -(\Delta_h)u_{i,j}=f_{i,j}, \\ u_{i,0}=\varphi_1(i), \\ u_{0,j}=\varphi_2(j), \\ u_{i,m}=\varphi_3(i), \\ u_{n,j}=\varphi_4(j). \end{cases} \qquad (2)$$

We use the second central difference to approximate the Laplace operator at an arbitrary interior cell with coordinates (x_i,y_j) and perform calculations on a five-point template. Thus, we obtain an approximation of the Poisson equation for interior cells:

$$-(\Delta_h u)_{i,j}=\frac{4u_{i,j}-u_{i-1,j}-u_{i+1,j}-u_{i,j-1}-u_{i,j+1}}{h^2}=f_{i,j} \qquad (3)$$

Since we use a step size equal to one, equation (3) has a simplified form:

$$f_{i,j}=4u_{i,j}-u_{i-1,j}-u_{i+1,j}-u_{i,j-1}-u_{i,j+1}. \qquad (4)$$

In reality, we do not use the whole domain Ω to get useful results, but only a certain part of it. The object in the domain Ω has a complex shape bounded by obstacles. To represent difference between regions, the $(m\times n)$-matrix $dist$ introduced by its elements:

$$dist_{ij}=\begin{cases} 1, (i,j) - \text{node of the obstacle region}, \\ 0, (i,j) - \text{node of the object region}. \end{cases} \qquad (5)$$

For all (i,j)-cells with $dist_{ij}=1$ we use $f_{i,j}=0$. Otherwise, function $f_{i,j}$ defined in tabular form. The values of the $f_{i,j}$ function can vary in the range [–1, 1]. Example of domain, divided into obstacle and object, presented in Fig. 1. Part of such domain presented more detail in Fig. 2.

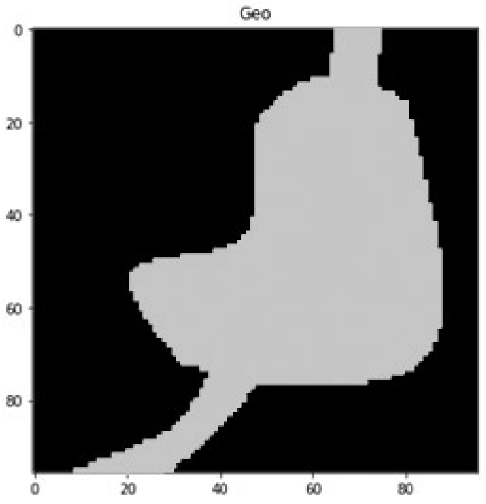

Figure 1 – The black region is an obstacle and the gray region is an object

We carry out calculations in accordance with equation (4) only for those (*i*, *j*)-cells for which the following conditions are satisfied:

$$\begin{cases} dist_{i,j+1}=0, \\ dist_{i,j-1}=0, \\ dist_{i+1,j}=0, \\ dist_{i-1,j}=0. \end{cases} (1<i<n, 1<j<m).$$

DOI 10.15588/1607-3274-2022-1-6

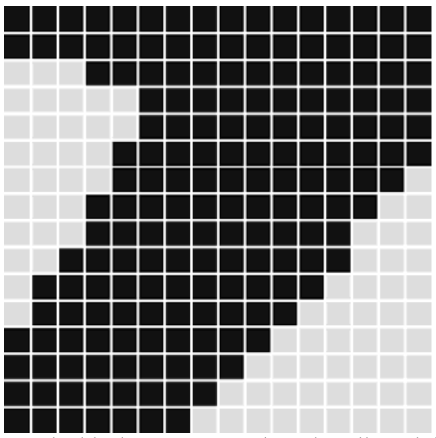
Figure 2 – The black squares are obstacle cells and the gray squares are object cells

The remaining cells belonging to the object may have 1, 2 or 3 neighbors located in the obstacle region. There may be various combinations of obstacle and object neighbor cells. All of them can be obtained by rotating and/or mirroring the configurations shown in Fig.3.

We use different approximation schemes for each of the following configurations:

– $4u_{i,j} - u_{i-1,j} - u_{i+1,j} - 2u_{i,j-1} = f_{i,j}$ for the cell with one obstacle (Fig. 3a),

– $4u_{i,j} - u_{i,j+1} - u_{i,j-1} = f_{i,j}$ for two opposite obstacles (Fig. 3b),

– $4u_{i,j} - 2u_{i,j-1} - 2u_{i+1,j} = f_{i,j}$ for two non-opposite obstacles (Fig. 3c),

– $4u_{i,j} - 2u_{i,j+1} = f_{i,j}$ for three obstacles (Fig. 3d),

We assume that $f_{i,j} = 0$ if i and j are outside the domain.

To solve this boundary value problem, we transform (2) into a system of linear algebraic equations and calculate the unknown value for each cell.

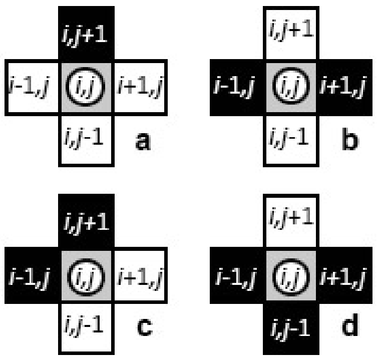

Figure 3 – Basic combinations of obstacle and object neighbor cells

2 REVIEW OF THE LITERATURE

The dynamics of fluid motion is an extremely important task that underlies research and engineering solutions in various fields of science and technology. CFD is widely used in many fields of study and industries [1]. At the same time, there is a problem associated with the fact that all numerical methods have a number of significant disadvantages [2].These disadvantages can be partially overcome by parallelizing the finite difference method [3]. However, to achieve convergence of this method, we must only use simple domain geometry or adaptive meshes [4].

Thanks to the successful development of machine learning and deep neural networks, it has become possible to make a breakthrough in the field of modeling physical processes [5]. The first investigations into solving PDEs using machine learning was carried out in the early 90s. Dissanayake and Phan-Thien [6] proposed multilayer perceptron to solve non-linear problems, such as Poisson equation and thermal conduction with non-linear heat generation. The main limitation of this method is a relatively small size of modeled domain [7]. Lee and Kang [8] introduced the general concept of developing neural algorithms for solving differential equations. Main limitations of studies of mentioned period are specific range of modelled RHS functions, a narrow set of BC and small domain sizes.

In the beginning of 2000s, with the improvement of computational efficiency, much more studies were published, which presented more complicated and robust models. In [9] the authors investigated Multilayer perceptrons to predict the orthogonal decomposition of the 2D Navier-Stokes equation and the 1D Kuramoto-Sivashinsky equation. A more in-depth and complex review of methods and techniques was conducted by Yadav and Kumar [10].

Xiao et al. [11] and Tompson et al. [12] were one of the first to research the use of convolutional neural networks to solve the Poisson equation. Both works proposed similar approaches to solve a boundary value problem based on the Poisson equation with a given RHS function. It is reported that in both cases there are problems with the accuracy of the results with reduced simulation time.

3 MATERIALS AND METHODS

The structure of the developed ANN was adapted and modified from the FPN architecture [13]. The presented version of the Poisson solution-oriented ANN allowed reducing the number of trained parameters from 23534592 in the basic network to 337447.

The presented ANN includes two main parts, called “bottom-up pathway” and “top-down pathway”. The general structure of ANN is shown in Fig. 4.

The input data for the “bottom-up pathway” is presented as a tensor of size 96×96×2. We obtain such data by combining two 96×96 matrices, where the first matrix is the RHS of the Poisson equation and the second matrix encodes the geometric space. The values of the elements of the second matrix correspond to the following rules: if the grid cell is located in the obstacle area, then the corresponding matrix element is zero, otherwise the value of the matrix element is equal to the distance to the nearest obstacle cell divided by 96√2. Example of geometry and respective encoding is shown in Fig. 5.

The "bottom-up pathway" part consists of 4 blocks. There are three modified "residual" blocks and one "pre_conv" block in it. The structure of the "pre_conv" block is shown in Fig. 6. It contains three layers, namely 2D convolution layer, batch normalization layer and relu activation layer. We included this block in the ANN structure to increase the number of channels from 1 to 64 and pass the tensor forward to the "residual" blocks.

Each subsequent "residual" block halves the tensor obtained from the previous block and transmits the result to the corresponding upsampling layer, which is located in the "top-down pathway" part. The proposed structure of a convolutional neural network uses the main property of neural networks of this type, which combines the characteristics of the fluid at different scales. Thus, we combine each cell with the rest of the considered domain. The data outputs of each "residual" block and the "pre_conv" block are transmitted to a 2D convolution layer with a core size of 1 × 1 to reduce the number of channels.

Next, convolution outputs pass to upsampling layers, which form "top-down pathway", where size of first "residual" block output multiplies by 2, second output by 4, and third output by 8. After this operation, all outputs combine into single tensor by concatenating them along last axis. Finally, this tensor moves to a two-dimensional block "output_conv", which consists of 2 layers of convolution: the first layer consists of 7 filters and core size 3, and the second contains 1 filter, core size 3 and the function of activating the hyperbolic tangent in the range from –1 to 1. Output tensor of the model is 2D array, with shape 96×96. To obtain solution of Poisson equation with input RHS, values of the output array must be rescaled into the original training data distribution.

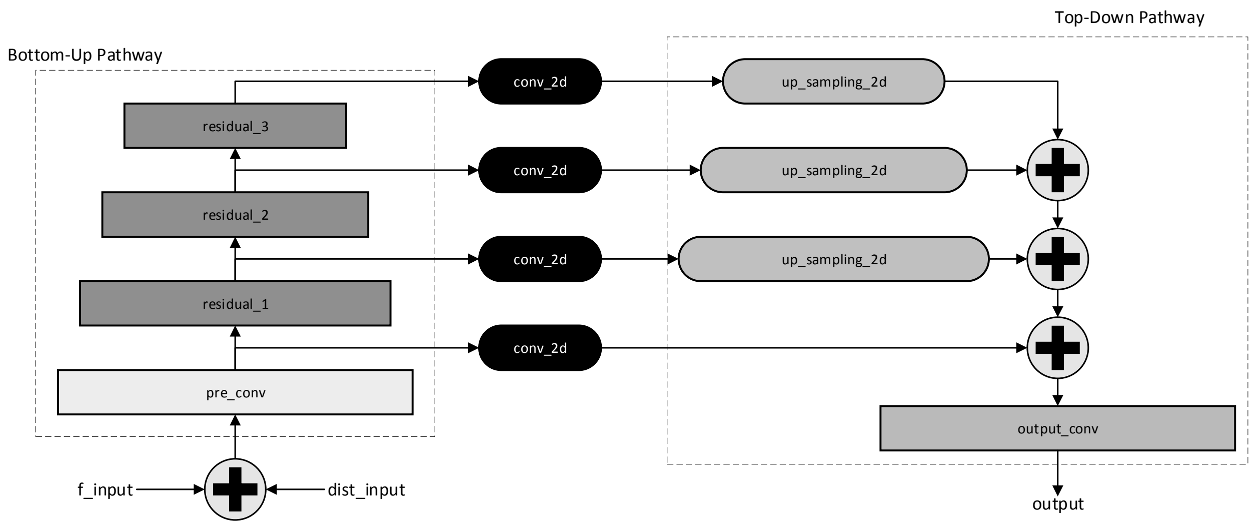

Figure 4 – General structure of the ANN

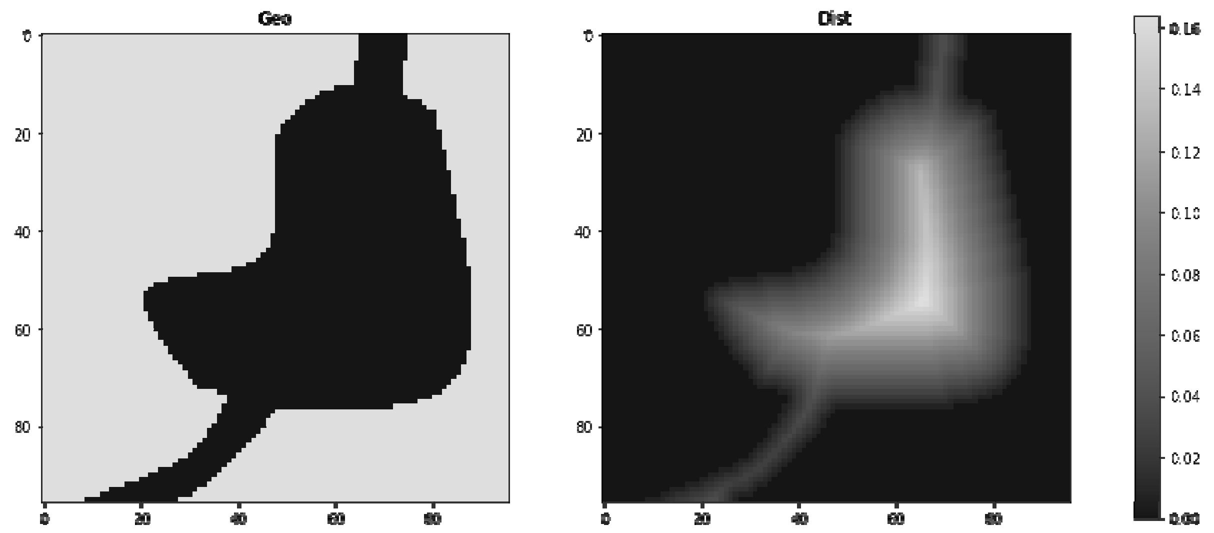

Figure 5 – The left shows the geometry of the zone where the light shade represents the obstacle zone; the light shades in the right figure correspond to the values of the elements of the distance matrix to the nearest obstacle

© Kuzmych V. A., Novotarskyi M. A., Nesterenko O. B., 2022
DOI 10.15588/1607-3274-2022-1-6

The training data set contains 40,000 pairs of samples, where the features are represented by 2 matrices of size 96 × 96. One matrix contains a tabular representation of RHS, and the other specifies the geometry of the space. The target matrix is a solution of the Poisson equation of 96 × 96 size. The solution is computed using the algebraic multigrid Python PyAMG package [14].

We prepared 140 geometries, and for each geometry, we generated 250 RHS matrices and corresponding solutions. Thus, 35 000 samples were processed.

Examples of such a pairs presented in fig. 8. Additionally, we generated 5000 samples without any geometries. An example of such a pair we can see in Fig. 7. In this case, all values of the distance matrix were equal to 1. RHS matrices were created by generating 8 × 8, 12 × 12 and 16 × 16 low-resolution grids with random values from –1 to 1, and then increasing the generated grids to the target size of 96 × 96 by cubic interpolation.

Each pair of samples contains an array with the solution of the equation, i.e. the target value. However, those values constrained in an interval, that highly bigger, that [–1, 1]. To provide stability of network training, all values of generated solution were normalized to distribution with 0 mean and 1 standard deviation. After normalization, those values were rescaled to range [–1, 1]. Data normalization and scaling was performed by scikit-learn library [15]. Those values are final target variable of training process.

The model was implemented using TensorFlow 2.4.1, with Adam weight optimizer [16], with following parameters: learning_rate=0.001, beta_1 = 0.9, beta_2 = 0.999, epsilon = 1e–7. We take mean square error as a loss function during training.

Training was conducted on GPU MSI GeForce GTX 1660 Super Ventus OC 6GB GDDR6, during 300 epochs and with batch size of 32.

4 EXPERIMENTS AND RESULTS

The described ANN is used in modeling the distribution of pressure in the human stomach. We modeled the stomach in 3 states-normal state and 2 different types of anastomosis (Fig. 9).

MSE in the first case is 0.000185, in the second – 0.000161, in the third – 0.000344.

Results demonstrated in Fig. 10–12.

We obtained an increase in the simulation speed compared to the numerical method. The PyAmg package was used to implement the numerical method. We measured the time of solving the boundary value problem on 1, 10, 50, 100, 200, 500 and 1000 samples. Due to the ability of ANN to process many samples simultaneously using a graphics processor, the highest increase in acceleration speed was achieved in 500 samples (Table 1 and Fig. 13).

5 DISCUSSION

Figures 10 and 11 show that the trained neural network selects the same regions with extreme pressure values as the numerical method chosen as the ground truth. Despite the ability to distinguish areas with high or low values, the trained ANN needs to be improved to make better predictions in the direction of smoothly varying values.

The experiments showed an increase in the simulation speed compared to the chosen numerical method. This fact can also be explained in particular by conducting experiments using GPU, which accelerate the matrix operations that are basic for CNN. Moreover, used GPU – nVidia GTX 1660 Super isn't the fastest GPU nowadays. Using newer and powerful GPU will achieve even bigger gain in computational speed. In respect of this fact and relatively small accuracy loss, in comparison with numerical method, the developed method can be applied in fields, where speed of modelling is more crucial, than accuracy, like some parts of biomedical engineering etc.

Further research will be aimed on improving neural network architecture to make predictions smoother. In addition, we believe that the main disadvantage of this result is the fixed size of the domain of 96 × 96. Therefore, we will focus on developing methods of deep learning paradigms that will allow us to work with arbitrary domain sizes. This approach involves improving the generation of training data sets, including the choice of a numerical method for solving a boundary value problem that directly affects the accuracy of the neural network.

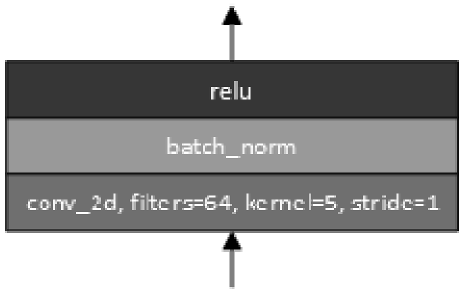

Figure 6 – "Pre_conv" block

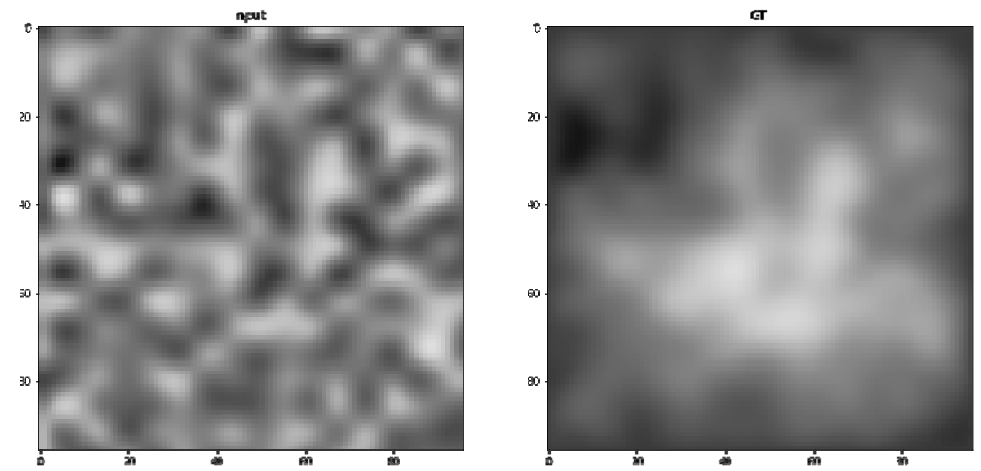

Figure 7 – The graphical representation of the RHS function is presented on the left, the corresponding solution is shown on the right

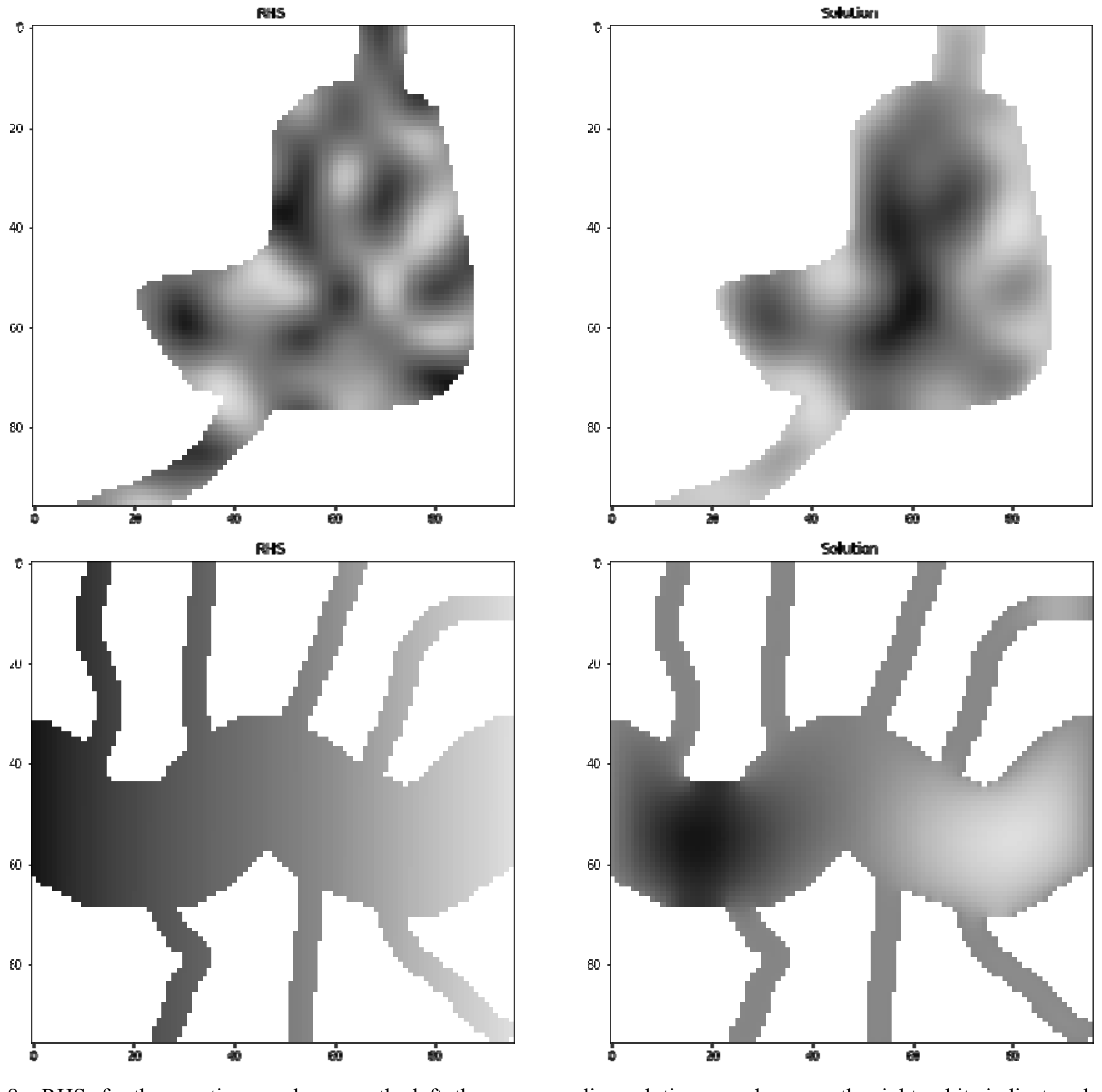

Figure 8 – RHSs for the equation are shown on the left, the corresponding solutions are shown on the right; white indicates obstacles

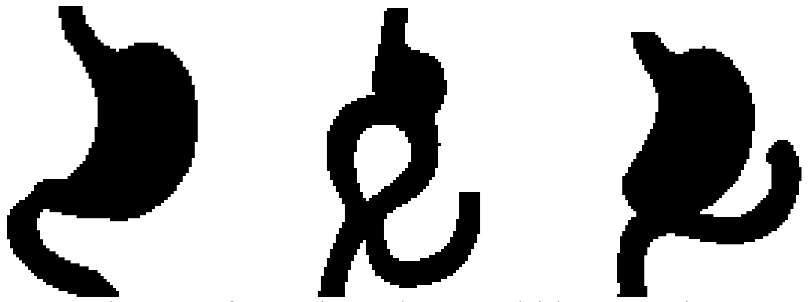

Figure 9 – Left –normal stomach, center and right-anastomosis

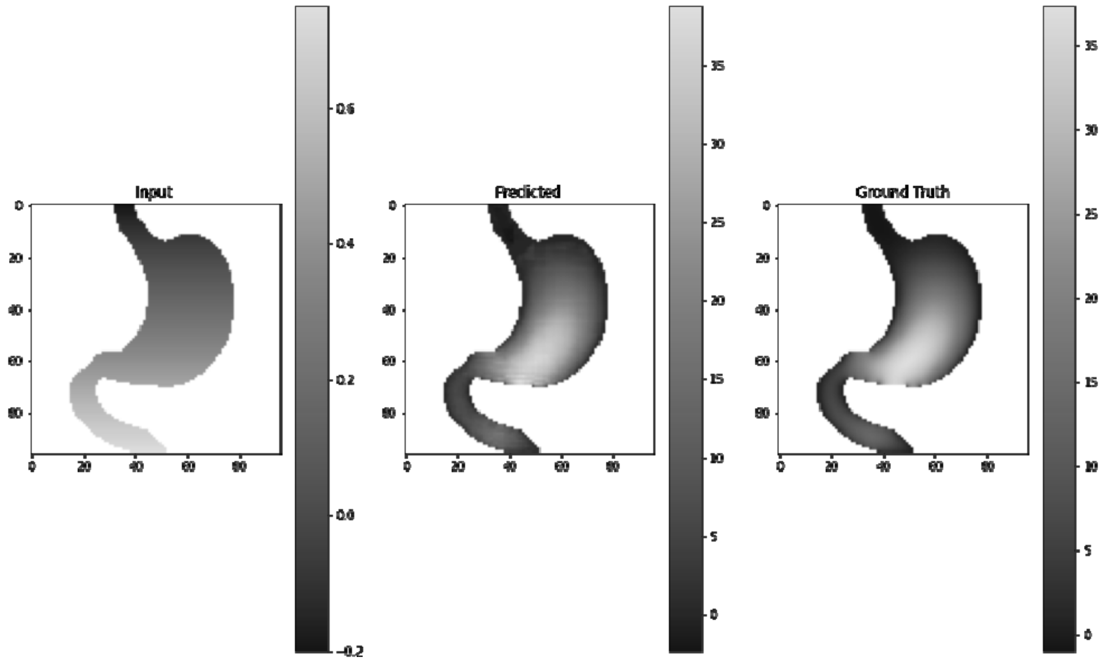

Figure 10 – Normal stomach; left – RHS of equation, center – network prediction, right – ground truth; white color denotes obstacles

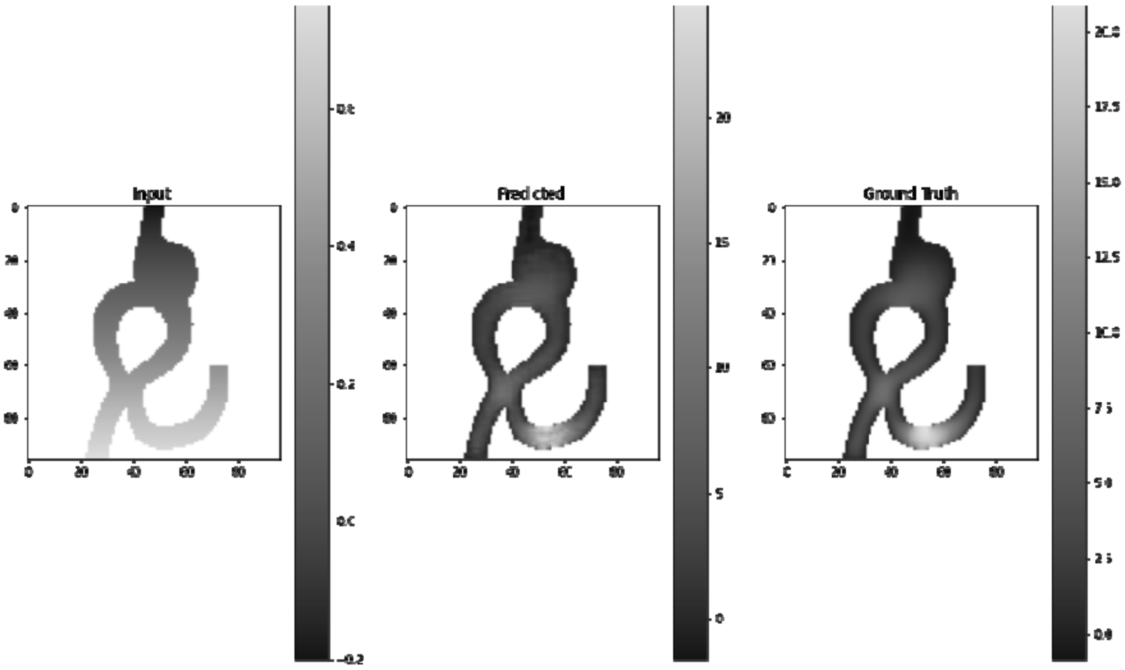

Figure 11 – Anastomosis; on the left – the right equation, the center – the network forecast, on the right – the basic truth; white indicates obstacles

Table 1 – Comparison of ANN and numerical method speed performance

Number of samples	Total time – ANN, ms	Time per sample – ANN, ms	Total time – numerical method, ms	Time per sample – numerical method, ms
1	23.80	23.80	237.00	237.00
10	29.00	2.90	7460.00	746.00
50	55.80	1.12	31400.00	628.00
100	91.20	0.91	52600.00	526.00
200	159.00	0.80	99000.00	495.00
500	334.00	0.67	261000.00	522.00
1000	725.00	0.73	536000.00	536.00

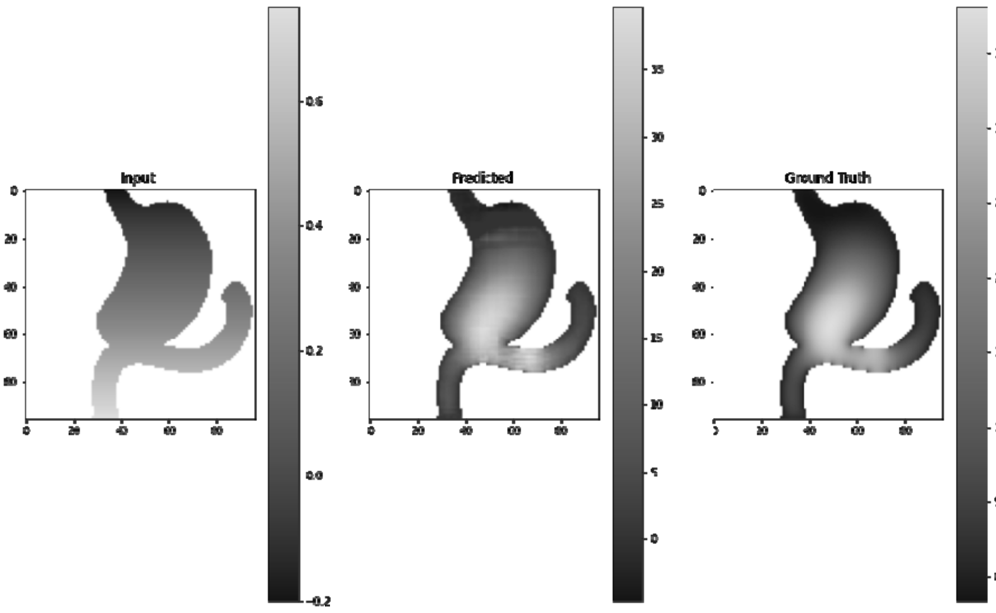

Figure 12 – Anastomosis; on the left – the right equation, the center – the network forecast, on the right – the basic truth; white indicates obstacles

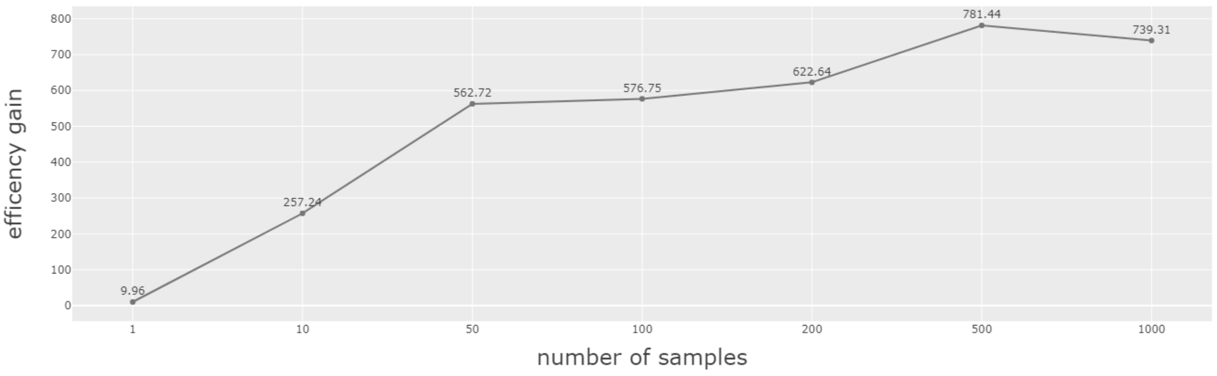

Figure 13 – Dependence of efficiency gain on the number of samples

CONCLUSIONS

This paper has investigated application of convolutional neural networks in solving boundary value problem based on the Poisson equation. We propose novel approach to solve Poisson equation with Dirichlet and Neumann boundary conditions, in domain with fixed size – 96×96.

Experiments with the developed method shown big gain of computational speed, in comparison with numerical method.

Scientific novelty is represented by the method of solving a boundary value problem based on the Poisson equation, which allowed to significantly accelerate the speed of its solution in comparison with the numerical method under the condition of a given acceptable accuracy.

The practical value of this achievement and insignificant loss in prediction accuracy is that convolutional neural network based technique can be applied in different field of engineering and science, where modelling speed matters.

Future work will focus on improvement of generalizability of CNN architecture, in order to handle various domain shapes and sizes. In addition, improving the speed of simulations using GPU for the tasks under consideration has not been widely studied. Therefore, continued research in this area is extremely important.

ACKNOWLEDGEMENTS

The results of these studies were used in scientific research projects “Artificial Intelligence Platform for Distant Computer-Aided Detection and Computer-Aided Diagnosis of Human Diseases” (state registration number 0120U105119) and “Application of mathematical methods in the problems of natural and technical sciences” (state registration number 0120U100979).

REFERENCES

1. Tu J. Computational Fluid Dynamics: A Practical Approach. Oxford: Butterworth-Heinemann, 2018, 495 p.
2. Wu C. Y., Ferng Y. M., Ciieng C. C., Liu C. C. Investigating the advantages and disadvantages of realistic approach and porous approach for closely packed pebbles in CFD simulation, *Nuclear Engineering and Design*, 2010, Vol. 240, pp. 1151–1159.
3. Simon H. D., Gropp W., Lusk E. Parallel Computational Fluid Dynamics: Implementations and Results (Scientific and Engineering Computation). Cambridge, The MIT Press, 1992, 362 p.
4. Runnels B., Agrawal V., Zhang W., Almgren A. Massively parallel finite difference elasticity using block-structured adaptive mesh refinement with a geometric multigrid solver, ArXiv e-prints, 2020, https://arxiv.org/abs/2001.04789v2, DOI: https://doi.org/10.1016/j.jcp.2020.110065
5. Raghu M., Schmidt E. A Survey of Deep Learning for Scientific Discovery, *ArXiv e-prints*, 2020, https://arxiv.org/abs/2003.11755v1.
6. Dissanayake M. W. M. G., Phan-Thien N. Neural-network-based approximations for solving partial differential equations, *Communications in Numerical Methods in Engineering,* 1994, Vol. 10, No. 3, pp. 195–201.
7. Sirignano J., Spiliopoulos K. DGM: A deep learning algorithm for solving partial differential equations, *Journal of Computational Physics*, 2018, Vol. 375, pp. 1339–1364.
8. Lee H., Kang I. S. Neural algorithm for solving differential equations, *Journal of Computational Physics*, 1990, Vol. 91, No. 1, pp. 110–131.
9. Smaoui N., Al-Enezi S. Modelling the dynamics of nonlinear partial differential equations using neural networks, *Journal of Computational and Applied Mathematics*, 2004, Vol. 170, No. 1, pp. 27–58.
10. Kumar M. and Yadav N. Multilayer perceptrons and radial basis function neural network methods for the solution of differential equations: A survey, *Computers & Mathematics with Applications*, 2011, Vol. 62, №10, pp. 3796–3811.
11. Xiao X., Zhou Y., Wang H., and Yang X. A novel cnn-based poisson solver for fluid simulation, *IEEE Transactions on Visualization and Computer Graphics,* 2020, Vol. 26, No. 3, pp. 1454–1465. DOI: 10.1109/TVCG.2018.2873375
12. Tompson J., Schlachter K., Sprechmann P., and Perlin K. Accelerating eulerian fluid simulation with convolutional networks, *ArXiv e-prints,* 2017, https://arxiv.org/abs/1607.03597.
13. Lin T.-Y., Dollar P., Girshick R., He K., Hariharan B., Belongie S. Feature pyramid networks for object detection, *ArXiv e-prints*, 2017, https://arxiv.org/abs/1612.03144v2
14. Olson L. N., Schroder J. B., PyAMG: Algebraic multigrid solvers in Python v4.0.0, 2018, https://github.com/pyamg
15. Pedregosa F. et al. Scikit-learn: machine learning in Python, *Journal of Machine Learning Research*, 2011, Vol. 12, pp. 2825–2830.
16. Duchi J., Hazan E., Singer Y. Adaptive subgradient methods for online learning and stochastic optimization, *Journal of Machine Learning Research*, 2011, Vol. 12, pp. 2121–2159.

Received 16.11.2021.
Accepted 16.01.2022.

УДК 004.93

РОЗВ’ЯЗУВАННЯ РІВНЯННЯ ПУАССОНА З ЗАСТОСУВАННЯМ ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ

Кузьмич В. А. – PhD, студент кафедри обчислювальної техніки, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського».

Новотарський М. А. – д-р техн. наук, професор кафедри обчислювальної техніки, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського».

Нестеренко О. Б. – канд. фіз.-мат. наук, завідувач кафедри прикладної фізики та вищої математики, Київський національний університет технологій та дизайну.

АНОТАЦІЯ

Актуальність. Рівняння Пуассона – це одне з фундаментальних диференціальних рівнянь, яке використовується для моделювання складних фізичних процесів, таких як рух рідини, проблеми теплообміну, електродинаміки тощо. Існуючі методи розв’язування крайових задач на основі рівняння Пуассона для досягнення високої точності, вимагають збільшення часу обчислень. Запропонований метод дозволяє розв’язувати крайову задачу зі значним прискоренням, за умови незначної втрати точності.

Мета. Метою нашої роботи є розробка архітектури штучної нейронної мережі для розв’язування крайової задачі на основі рівняння Пуассона з довільними крайовими умовами Діріхле та Неймана.

Метод. Запропоновано метод розв’язування крайових задач на основі рівняння Пуассона за допомогою згорткової нейронної мережі. Розроблено архітектуру мережі, структуру вхідних та вихідних даних. Також описано метод формування навчального набору даних.

Результати. Результати роботи розробленої нейронної мережі були порівняні з продуктивністю чисельного методу скінченних різниць для вирішення крайової задачі. Результати продемонстрували прискорення обчислювальної швидкості у x10–700 разів, в залежності від кількості вузлів дискретизації.

DOI 10.15588/1607-3274-2022-1-6

Висновки. Запропонований метод значно прискорив швидкість вирішення крайової задачі на основі рівняння Пуассона в порівнянні з чисельним методом. Також розроблений підхід до проектування архітектури нейронної мережі дозволяє вдосконалити запропонований метод для досягнення більш високої точності при моделюванні процесу розподілу тиску у областях довільного розміру.

КЛЮЧОВІ СЛОВА: машинне навчання, рівняння Пуассона, згорткова нейронна мережа.

УДК 004.93

РЕШЕНИЕ УРАВНЕНИЯ ПУАССОНА С ПРИМЕНЕНИЕМ СВЕРТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ

Кузьмич В. А. – PhD студент кафедры вычислительной техники, Национальный технический университет Украины «Киевский политехнический институт имени Игоря Сикорского».

Новотарский М. А. – д-р техн. наук, профессор кафедры вычислительной техники, Национальный технический университет Украины «Киевский политехнический институт имени Игоря Сикорского».

Нестеренко О. Б. – канд. физ.-мат. наук, заведующая кафедрой прикладной физики и высшей математики, Киевский национальный университет технологий и дизайна.

АННОТАЦИЯ

Актуальность. Уравнение Пуассона – это одно из фундаментальных дифференциальных уравнений, которое используется для моделирования сложных физических процессов, таких как движение жидкости, проблемы теплообмена, электродинамики и тому подобное. Существующие методы решения краевых задач на основе уравнения Пуассона для достижения высокой точности, требуют увеличения времени вычислений. Предложенный метод позволяет решать краевую задачу со значительным ускорением при условии незначительной потери точности.

Цель. Целью нашей работы является разработка архитектуры искусственной нейронной сети для решения краевой задачи на основе уравнения Пуассона с произвольными граничными условиями Дирихле и Неймана.

Метод. Предложен метод решения краевых задач на основе уравнения Пуассона с помощью сверточной нейронной сети. Разработана архитектура сети, структура входных и выходных данных. Также описан метод формирования учебного набора данных.

Результаты. Результаты работы разработанной нейронной сети были сравнены с производительностью численного метода конечных разностей для решения краевой задачи. Результаты продемонстрировали ускорение вычислительной скорости в x10–700 раз, в зависимости от количества узлов дискретизации.

Выводы. Предлагаемый метод значительно увеличил скорость решения краевой задачи на основе уравнения Пуассона по сравнению с численным методом. Также разработанный подход к проектированию архитектуры нейронной сети позволяет улучшить предложенный метод для достижения большей точности при моделировании процесса распределения давления в областях произвольного размера.

КЛЮЧЕВЫЕ СЛОВА: машинное обучение, уравнение Пуассона, сверточная нейронная сеть.

ЛІТЕРАТУРА / ЛИТЕРАТУРА

1. Tu J. Computational Fluid Dynamics: A Practical Approach / J. Tu. – Oxford: Butterworth-Heinemann, 2018. – 495 p.
2. Investigating the advantages and disadvantages of realistic approach and porous approach for closely packed pebbles in CFD simulation / [Wu C. Y., Ferng Y. M., C. C. Ciieng, C. C. Liu] // Nuclear Engineering and Design. – 2010. – Vol. 240. – P. 1151–1159.
3. Simon H. D. Parallel Computational Fluid Dynamics: Implementations and Results (Scientific and Engineering Computation) / H. D. Simon, W. Gropp, E. Lusk. – Cambridge : The MIT Press, 1992. – 362 p.
4. Massively parallel finite difference elasticity using block-structured adaptive mesh refinement with a geometric multigrid solver / [B. Rennels, V. Agrawal, W. Zhang, A. Almgren] // ArXiv e-prints. – 2020. https://arxiv.org/abs/2001.04789v2, DOI:https://doi.org/10.1016/j.jcp.2020.110065
5. Raghu M. A Survey of Deep Learning for Scientific Discovery / M. Raghu, E. Schmidt // ArXiv e-prints. – 2020, https://arxiv.org/abs/2003.11755v1.
6. Dissanayake M. W. M. G. Neural-network-based approximations for solving partial differential equations / M. W. M. G. Dissanayake N. Phan-Thien // Communications in Numerical Methods in Engineering. – 1994. – Vol. 10, № 3. – P. 195–201.
7. Sirignano J. DGM: A deep learning algorithm for solving partial differential equations / J. Sirignano, K. Spiliopoulos // Journal of Computational Physics, 2018. – Vol. 375. – P. 1339–1364.
8. Lee H. Neural algorithm for solving differential equations / H. Lee, I. S. Kang // Journal of Computational Physics, 1990. – Vol. 91, №1. – P. 110–131.
9. Smaoui N. Modelling the dynamics of nonlinear partial differential equations using neural networks / N. Smaoui, S. Al-Enezi // Journal of Computational and Applied Mathematics. – 2004. – Vol. 170, №1. – P. 27–58.
10. Kumar M. Multilayer perceptrons and radial basis function neural network methods for the solution of differential equations: A survey / M. Kumar and N. Yadav // Computers & Mathematics with Applications, 2011. – Vol. 62, № 10. – P. 3796–3811.
11. A novel cnn-based poisson solver for fluid simulation / [X. Xiao, Y. Zhou, H. Wang, and X. Yang] // IEEE Transactions on Visualization and Computer Graphics. – 2020. – Vol. 26, № 3. – P. 1454–1465. DOI: 10.1109/TVCG.2018.2873375
12. Accelerating eulerian fluid simulation with convolutional networks / [J. Tompson, K. Schlachter, P. Sprechmann, and K. Perlin] // ArXiv e-prints. – 2017. https://arxiv.org/abs/1607.03597.
13. Feature pyramid networks for object detection / [P. Dollar, R. Girshick, K. He et al] // ArXiv e-prints. – 2017, https://arxiv.org/abs/1612.03144v2
14. Olson L. N. PyAMG: Algebraic multigrid solvers in Python v4.0.0 / L. N. Olson, J. B. Schroder. – 2018. https://github.com/pyamg
15. Pedregosa F. Scikit-learn: machine learning in Python / F. Pedregosa et al. // Journal of Machine Learning Research. – 2011. – Vol. 12. – P. 2825–2830.
16. Duchi J. Adaptive subgradient methods for online learning and stochastic optimization / J. Duchi, E. Hazan, Y. Singer // Journal of Machine Learning Research. – 2011. – Vol. 12. – P. 2121–2159.

DOI 10.15588/1607-3274-2022-1-6