ISSN 0027-1349, Moscow University Physics Bulletin, 2024, Vol. 79, Suppl. 2, pp. S922–S934. © Allerton Press, Inc., 2024.

Neural Operators for Hydrodynamic Modeling of Underground Gas Storages

D. D. Sirota[1*], K. A. Gushchin[1], S. A. Khan[1], S. L. Kostikov[1], and K. A. Butov[1]

[1] *PJSC Gazprom, St. Petersburg, 197229 Russia*

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

Abstract—Hydrodynamic modeling via numerical simulators of underground gas storages (UGSs) is an integral part of planning and decision-making in various aspects of UGS operation. Although numerical simulators can provide accurate predictions of numerous parameters in UGS reservoirs, in many cases this process can be computationally expensive. In particular, calculation time is one of the major constraints affecting decisions related to optimal well control and distribution of gas injection or withdrawal over the reservoir area. Novel deep learning methods that can provide a faster alternative to traditional numerical reservoir simulators with acceptable loss of accuracy are investigated in this paper. Hydrodynamic processes of gas flow in UGS reservoirs are described by partial differential equations (PDEs). Since PDEs involve approximating solutions in infinite-dimensional function spaces, this distinguishes such problems from traditional ones. Currently, one of the most promising machine learning approaches in scientific computing (scientific machine learning) is the training of neural operators that represent mappings between function spaces. In this paper, a deep learning method for hydrodynamic modeling of UGS is proposed. A modified Fourier neural operator for hydrodynamic modeling of UGS is developed, in which the model parameters in the spectral domain are represented as factorized low-rank tensors. We trained the model on data obtained from a numerical model of UGS with nonuniform discretization grid, more than 100 wells and complex geometry. Our method demonstrates superior performance compared to the original Fourier neural operator (FNO), with an order of magnitude (50 times) fewer parameters. Tensor decomposition not only greatly reduced the number of parameters, but also increased the generalization ability of the model. Developed neural operator simulates a given scenario in a fraction of a second, which is at least 10^6 times faster than a traditional numerical solver.

Keywords: mathematical modeling, deep learning, artificial intelligence, neural networks, neural operators, Fourier neural operators, hydrodynamic modeling, underground gas storage

DOI: 10.3103/S0027134924702382

1. INTRODUCTION

Underground gas storage (UGS) is a technological complex designed for the injection, storage, and withdrawal of gas, which includes surface engineering and technical facilities; a section of the subsoil limited by a mining allotment; a gas storage facility; control layers; a buffer gas volume; and a fund of wells for various purposes.

Modeling of gas flow in UGS reservoirs plays an crucial role in improving the accuracy and reliability of forecasting. Consequently, hydrodynamic modeling of UGS is an integral part of planning and decision-making in various aspects of UGS operation, such as optimal well utilization, pressure control and gas inventory management.

In the field of modeling underground gas storage processes, two approaches are common: simplified balance models and more accurate numerical hydrodynamic models (HDMs). Balance models are often used due to insufficient data for building three-dimensional numerical models or due to a lack of computing power. Balance models obtain a solution to the gas flow equation in a porous medium using simplified dependences and do not take into account complex geological and hydrodynamic peculiar properties that can have a significant impact on the behavior of UGS.

In turn, modern numerical simulators for hydrodynamic modeling of gas flow in reservoirs allow obtaining significantly more accurate information about the distribution of parameters in UGS reservoirs and calculating the influence of various factors on gas

*E-mail: D.Sirota@adm.gazprom.ru

storage processes. Numerical simulators typically use a finite volume method to approximate a system of differential equations in space and an implicit scheme to approximate in time, which can be a computationally expensive procedure.

Currently, numerical HDMs are mainly used to solve problems of hydrodynamic modeling of UGS. Due to the fact that such models can be adapted to the accumulated history of field development (in the case of creating UGS in depleted fields) and the actual history of UGS operation, the modeling and adaptation horizon of models can be more than 60 years. Moreover, given the significant amount of geological and production data supplied to hydrodynamic models as initial data (results of geophysical surveys, measurements of pressure, gas flow rates, etc.), the time of one calculation can reach several hours.

Thus, the calculation time is one of the main factors influencing the processes related to optimal well control the distribution of gas injection or withdrawal across wells and over the reservoir area. One of the promising approaches to solving the issue of accelerating hydrodynamic calculations is the use of modern deep learning methods.

A significant part of research in deep learning methods is focused on the study of mappings between finite-dimensional vector spaces [1, 2]. In turn, the physical processes of gas flow in UGS reservoirs are described by partial differential equations (PDEs), which require the study of mappings between functional spaces of infinite dimension, which distinguishes this problem from traditional ones [3].

According to the universal approximation theorem [4, 5], a fully connected network with a sufficient number of parameters can potentially approximate any continuous function defined on a compact set with any accuracy. In [6–8] the theoretical possibilities of approximating nonlinear mappings between functional spaces are shown. Moreover, in [9], estimates of the complexity bounds of the approximation error of neural networks are given, linking the number of model parameters and the dimension of the problem with the value of the approximation error.

In addition to the theoretical possibility of approximating mappings between infinite-dimensional spaces, efficient algorithms for such problems are needed for successful practical application. It is known that different neural network architectures show different performance in solving specific problems. For example, fully connected neural networks demonstrate significantly lower quality in computer vision problems compared to widely used convolutional architectures [2]. To study the issue of efficient training of neural network algorithms, [10] proposes to decompose the total model error into three components: approximation error, optimization error, and generalization error. The approximation error depends on the number of network parameters and the problem dimension, the optimization error is associated with the loss function, and the generalization error depends on the size of the training sample [11].

A crucial point in constructing mappings that generalize the dependences described by PDEs using neural networks is the curse of dimensionality [12, 13]. Especially for objects with complex geometry, such as UGS and for equations with multidimensional parameter spaces, such as the three-dimensional unsteady gas flow equation [3]. It is known that neural network algorithms may require a large training dataset to generalize the main dependences and relationships. According to [11], the upper bound on the generalization error is: $E_{\text{gen}} \sim \frac{1}{\sqrt{N}}$, where N is the number of training samples. In this case, to obtain a relative generalization error of 1%, a dataset of size $\sim \mathcal{O}\left(10^4\right)$ is required.

In the case of modeling hydrodynamic processes in reservoir systems, obtaining a large training data set can be a difficult task, since the data set is formed from calculations on a numerical simulator, which is a computationally expensive process. Therefore, taking into account the previously mentioned points, the development of neural network algorithms for effectively solving problems of this type is a nontrivial and relevant issue.

In recent years, deep learning methods have become widely used in scientific computing tasks, becoming a new paradigm [14, 15]. In particular, many algorithms have been developed that offer faster alternatives to numerical modeling.

Surely, there are many studies based on traditional deep learning approaches in the form of constructing finite-dimensional operators, using the results of numerical simulations as a training data. For example, in [16–18], convolutional, recurrent, and generative-adversarial architectures for solving hydrodynamic problems were investigated. But since the presented methods do not use knowledge about the structure of the modeled dependences, they are sensitive to the geometry of the object, the discretization grid, and require a large amount of data.

The group of methods [19–21] belongs to a specific class of algorithms, the so-called physics-informed neural networks (PINNs). This approach is also based on finite-dimensional mappings, but takes into account the PDE as a part of loss function, using the mechanism of automatic differentiation [22]. Thus, physics is taken into account during learning process, since the model tries to minimize the residuals between the left and right parts of

the equation, the initial and boundary conditions. The disadvantage of this approach is the ability to approximate only a specific instance of the PDE. Consequently, PINNs do not provide significant acceleration relative to traditional numerical methods for many applied problems.

A more promising approach is to train neural operators that represent mappings between function spaces [23–25]. Neural operators can approximate any nonlinear continuous operators, are grid-independent, and require only one-shot training, so they can be trained and evaluated on different discretization grids and PDE instances. These methods have been shown to have better performance in PDE approximation problems than other existing deep learning-based approaches, including for hydrodynamic modeling problems.

The construction and training of neural operators for approximating mappings in infinite-dimensional spaces is an active area of research, and work is currently ongoing to improve the efficiency and applicability of this approach to various problems.

2. PROBLEM SETTING

2.1. Mathematical Model of Gas Flow in Porous Media

This paper investigates the process of hydrodynamic modeling of UGS with porous reservoirs. For such objects, various parameters included in the equation of gas flow in a porous media (filtration) have a significant dependence on time [3]. Such processes are called unsteady (nonstationary).

The general equation (Eq. (1)) of a three-dimensional unsteady single-phase flow of a compressible fluid (natural gas) in a porous media is obtained by substituting the law of conservation of momentum (Darcy's filtration law) into the law of conservation of mass [26]:

$$\frac{\partial}{\partial x}\left(\frac{A_x k_x}{\mu_g B_g}\frac{\partial p}{\partial x}\right)\Delta \mathrm{x} + \frac{\partial}{\partial y}\left(\frac{A_y k_y}{\mu_g B_g}\frac{\partial p}{\partial y}\right)\Delta \mathrm{y} + \frac{\partial}{\partial z}\left(\frac{A_z k_z}{\mu_g B_g}\frac{\partial p}{\partial z}\right)\Delta \mathrm{z} = \frac{V_b \phi T_{\mathrm{sc}}}{p_{\mathrm{sc}} T}\frac{\partial}{\partial t}\left(\frac{p}{Z}\right) - q_{\mathrm{gsc}}, \tag{1}$$

where p is the pressure, q_{gsc} is the gas flow rate at standard conditions, $B_g = \frac{p_{\mathrm{sc}} T Z}{T_{\mathrm{sc}} p}$ is the gas formation volume factor, Z is the compressibility correction factor (z-factor) of a real gas, μ_g is the gas viscosity, T_{sc} is the temperature at standard conditions, p_{sc} is the pressure at standard conditions, ϕ is the porosity, k is the permeability, A is the cross-sectional area of fluid flow, V_b is the reservoir bulk volume, Δx, Δy, Δz are the length, width and height of the rock volume (control volume), respectively.

The partial differential equation (Eq. (1)) is nonlinear due to the dependence of μ_g, B_g, and Z on pressure and is similar to the diffusion equation, however, in terms of its dynamic characteristics, the flow described by this relationship is not diffusion but filtration.

2.2. Operator Learning

The purpose of this work is to approximate the gas flow equation in reservoirs of UGS (Eq. (1)) by constructing a neural operator that maps between two infinite-dimensional spaces from a finite set of pairs of initial, boundary conditions and solutions of PDE. In the case of implementing such an approach, it is possible to obtain a significant acceleration of the calculation speed with an acceptable loss of accuracy relative to numerical simulations.

Let the problem dimension $d \in \mathbb{N}$ and denote by $D \subset R^d$ the domain in R^d. Then we can consider a mapping that is essentially a solution operator for a PDE (Eq. (2)):

$$\mathcal{G} : \mathcal{A}\left(D; R^{d_a}\right) \to \mathcal{U}\left(D; R^{d_u}\right), \quad a \mapsto u := \mathcal{G}(a), \tag{2}$$

where $a \in \mathcal{A}\left(D; R^{d_a}\right)$ is an input function of the form $a : D \mapsto R^{d_a}$, $u \in \mathcal{U}\left(D; R^{d_u}\right)$ is an output function of the form $u : D \mapsto R^{d_u}$. $\mathcal{A}\left(D; R^{d_a}\right)$ and $\mathcal{U}\left(D; R^{d_u}\right)$ are Banach spaces.

For operator learning, there is a finite collection of pairs of initial, boundary conditions and solutions of the PDE $\{a_j, u_j\}_{j=1}^N$, where $a_j \sim \mu$ is an i.i.d. sequence from probability measure μ defined on $\mathcal{A}$ and $u_j = \mathcal{G}(a_j)$. In our case these pairs can be obtained using historical data or by simulations on a numerical hydrodynamic model of the UGS. This numerical model uses the finite volume method to approximate the system of partial differential equations in space and an implicit scheme for approximation in time. Thus, the training of the neural operator can be formulated as follows.

The initial data generated from the numerical simulator are essentially the result of a nonlinear mapping satisfying the gas flow equation: $\mathcal{G}(a_j) = u_j$. As a result, it is possible to construct a neural operator $\mathcal{N}_{\theta^*}$ by selecting the parameters $\theta \in \Theta$ in such a way as to approximate the original mapping $\mathcal{N}_{\theta^*} \approx \mathcal{G}$. Then the learning process is reduced to the problem of minimizing the loss function $C : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ and has the form (Eq. (3)):

$$\min_{\theta} \mathbb{E}_{a\sim\mu}\left[C\left(\mathcal{N}_\theta(a), \mathcal{G}(a)\right)\right]. \tag{3}$$

3. NEURAL OPERATORS

In accordance with the goals of this work, it is planned to train a neural operator that approximates the PDE solution of gas flow in a UGS. To develop such methods, it is convenient to adhere to the following sequence of steps in constructing the model architecture [25]:

1. $\mathcal{P}$—Operator which lifts input data to higher dimensional latent space;
2. Iterative application of the kernel integration operator ($\mathcal{L}$);
3. $\mathcal{Q}$—Operator of projection from latent space into original output space.

Thus, the structure of the neural operator has the form (Eq. (4)):

$$\mathcal{N}(a) = \mathcal{Q} \circ \mathcal{L}_L \circ \mathcal{L}_{L-1} \circ \cdots \circ \mathcal{L}_1 \circ \mathcal{P}(a), \quad (4)$$

where the layers depth is $L \in \mathbb{N}$, $\mathcal{P} : \mathcal{A}\left(D; R^{d_a}\right) \to \mathcal{U}\left(D; R^{d_v}\right)$, $d_v \geq d_a$, $\mathcal{Q} : \mathcal{U}\left(D; R^{d_v}\right) \to \mathcal{U}\left(D; R^{d_u}\right)$.

By analogy with classical finite-dimensional neural networks $\mathcal{L}_1, \ldots, \mathcal{L}_L$ are nonlinear layers of the operator, $\mathcal{L}_l : \mathcal{U}\left(D; R^{d_v}\right) \to \mathcal{U}\left(D; R^{d_v}\right)$, $v \mapsto \mathcal{L}_l(v)$, which can be written as (Eq. (5)):

$$\mathcal{L}_l(v)(x) = \sigma\Big(W_l v(x) + (\mathcal{K}(a;\theta_l)\, v)(x)\Big), \quad \forall x \in D \quad (5)$$

where W_l is a linear transformation, $\mathcal{K} : \mathcal{A} \times \Theta \to \mathcal{L}(\mathcal{U}(D;\mathbb{R}^{d_v}), \mathcal{U}(D;\mathbb{R}^{d_v}))$.

The operator $\mathcal{K}(a;\theta_l)$ [24] is an integral operator of the form (Eq. (6)):

$$(\mathcal{K}(a;\theta_l)v)(x) = \int_D \kappa_\theta(x, y; a(x), a(y))v(y)dy, \quad \forall x \in D. \quad (6)$$

The kernel κ_θ is a neural network with parameters $\theta \in \Theta$ and can have different structures. Due to this, there are various types of neural operators. For example, graph neural operators (GNO, MGNO) [24], low-rank neural operators (LNO), Fourier neural operators (FNO).

Currently, one of the promising methods for approximating solutions of the gas flow equation is FNO (Fig. 1), which parameterizes the kernel of the integral operator in the Fourier space [27]. This method is one of the most studied and shows better efficiency in problems of fluid filtration in a porous media compared to traditional neural network algorithms and other operator architectures (GNO, MGNO, LNO, DeepONet) [25]. Moreover, in [28] it is shown that to achieve a given error, the complexity of FNO grows logarithmically, for the case of approximating the diffusion equation. In contrast to the alternative architecture DeepONet [23], for which the complexity grows quadratically.

The Fourier neural operator [27] belongs to a class of neural operators in which the kernel can be written as a convolution (Eq. (7)):

$$(\mathcal{K}(a;\theta_l)v)(x) = \int_D \kappa_\theta(x-y)v(y)dy, \quad \forall x \in D. \quad (7)$$

To efficiently parameterize the kernel, according to the convolution theorem, this method considers the image of v in Fourier space using the fast Fourier transform (FFT) algorithm $\mathcal{F}$ and the inverse FFT $\mathcal{F}^{-1}$ (Eq. (8)):

$$(\mathcal{K}(\theta)v)(x) = \mathcal{F}^{-1}\Big(R_\theta(k) \cdot \mathcal{F}(v)(k)\Big)(x), \quad \forall x \in D, \quad (8)$$

where $R_\theta(k) = \mathcal{F}(\kappa_\theta)(k)$ is the tensor of Fourier transform coefficients of κ_θ.

Thus, the layers of the Fourier neural operator will have the form:

$$\mathcal{L}_l(v)(x) = \sigma\Big(W_l v(x) + \mathcal{F}^{-1}(R_l(k) \cdot \mathcal{F}(v)(k))(x)\Big). \quad (9)$$

The key difference between (Eq. (9)) and traditional neural network architectures is that all operations are defined directly in the function space and hence do not depend on the data discretization.

To develop an efficient algorithm for a neural operator that approximates a nonlinear equation of gas flow in porous media, it is crucial to take into account the multidimensional spatiotemporal structure of the underlying PDE solution operator. This takes the problem beyond the scope of matrix linear algebra into the field of tensor methods. These methods are widely used in many applications. As shown in [29], parameterization of multidimensional convolutional operators using tensor decompositions allows for a significant increase in the efficiency of convolutional neural networks in problems with significant spatiotemporal structure.

Tensor decompositions (canonical polyadic, Tucker, tensor-train) have also shown high performance in the field of neural operator learning. The authors [30] have deeply investigated this approach on the Navier–Stokes and Burgers equations, showing its effectiveness.

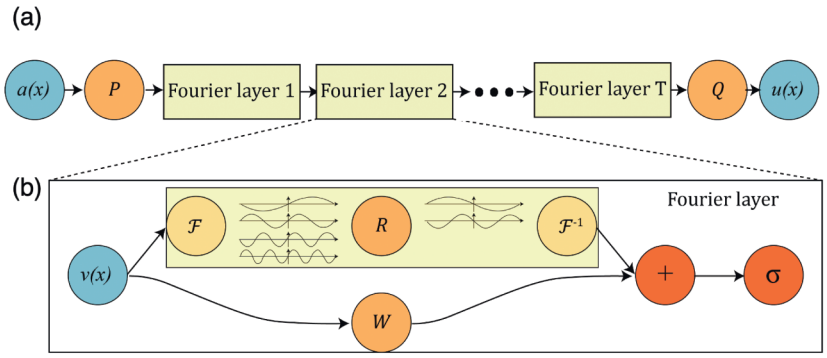

Fig. 1. (a) FNO model architecture: $a(x)$ is the model input, P is a fully connected lifting layer into a higher-dimensional latent space, Q is a fully connected projection layer into the physical space, $u(x)$ is the model output; (b) Fourier layer: $\mathcal{F}, \mathcal{F}^{-1}$ are the Fourier transform and the inverse Fourier transform, R is a linear transform in the Fourier space, W is a linear transform.

Consider the Tucker [31] decomposition (for its flexibility) in a three-dimensional formulation (Eq. (10)):

$$\mathbf{W} = \mathbf{G} \times_1 \mathbf{U^{(1)}} \cdots \times_d \mathbf{U^{(d)}} \times_{d+1} \mathbf{U^{(I)}} \times_{d+2} \mathbf{U^{(O)}} \times \ {}_{d+3}\mathbf{U^{(L)}}, \quad (10)$$

where $\mathbf{W}$ is the tensor of model parameters, $\mathbf{G}$ is the core tensor, $\mathbf{U^{(1)}}, \cdots, \mathbf{U^{(d)}}, \mathbf{U^{(I)}}, \mathbf{U^{(O)}}, \mathbf{U^{(L)}}$ — factor matrices.

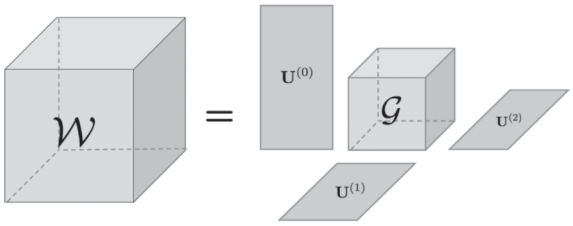

Fig. 2. Illustration of a Tucker decomposition for a three-dimensional model parameters tensor.

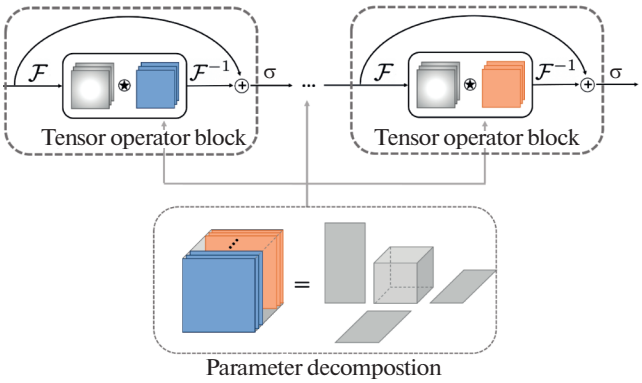

Fig. 3. Factorized Fourier layer with parameters represented in the spectral domain via a low-rank tensor decomposition.

A visualization of the Tucker decomposition of a third-order tensor can be seen in Fig. 2.

In this paper, we present a method for hydrodynamic modeling of UGS using a modified Fourier neural operator, where the model parameters in the spectral domain are represented as factorized low-rank tensors (Fig. 3).

Tensor decompositions significantly reduce the number of model parameters, enabling the implementation of more complex operators that enhance modeling accuracy.

4. NUMERICAL EXPERIMENTS

4.1. Data Configuration

The dataset in this paper is formed using data from a hydrodynamic model of an UGS. The approximation period is chosen to be equal to the gas withdrawal season. However, to increase amount of training data, the dataset also includes injection seasons data. The entire dataset is formed from historical data and simulations of various withdrawal and injection scenarios with a time step of 10 days (decades). The UGS has more than 100 operating wells and complex geometry.

The dataset consists of 620 input-output pairs. There are 540 samples for the training dataset and 40 each for the validation and test datasets.

Note that the training set consists of the withdrawal and injection seasons data, while the validation and testing sets consist only of the withdrawal seasons data.

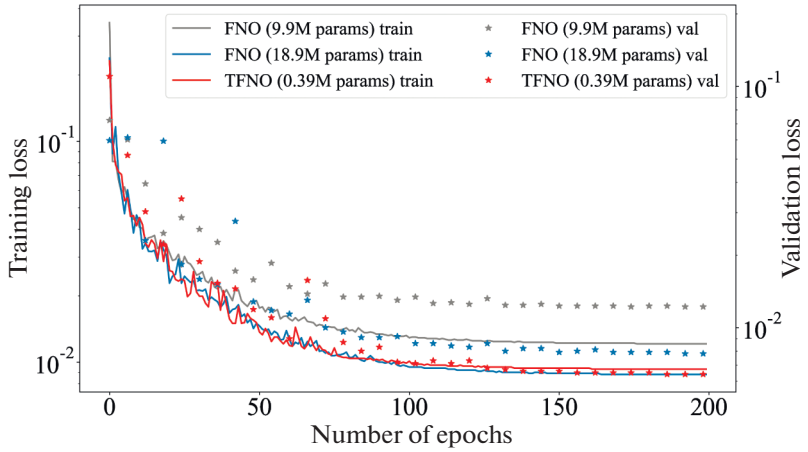

Fig. 4. Training and validation losses evolution vs epochs.

4.2. Loss Functions and Training

We use relative H^1 Sobolev norm as loss function [32] to train the model (Eq. (11)) because it has a good normalization and regularization effect since the reservoir pressure in the UGS has different scales at different spatial and temporal points. Moreover, by matching derivatives, such loss function allows the neural operator to better capture the smoothness of a physical process:

$$L(y,\hat{y}) = \left(\sum_{i=0}^{k} \frac{||D^i y - D^i \hat{y}||_p^p}{||D^i y||_p^p}\right)^{\frac{1}{p}}, \quad k=1, \quad p=2 \tag{11}$$

where $\hat{y}$ is the pointwise predicted reservoir pressure, y is the pointwise ground truth reservoir pressure, D^i is a differential operator of order i, p is the order of norm. Computing derivatives is implemented via finite-difference method.

Relative L_2 error function is used for validation and testing (Eq. (12)):

$$L_2(y,\hat{y}) = \frac{||y-\hat{y}||_2}{||y||_2}. \tag{12}$$

The model predicts the evolution of the reservoir pressure field over the entire period of time.

During training, the initial learning rate is specified to be 0.001 and gradually decreases with the number of epochs passed. Training stops when the loss on the validation set no longer decreases.

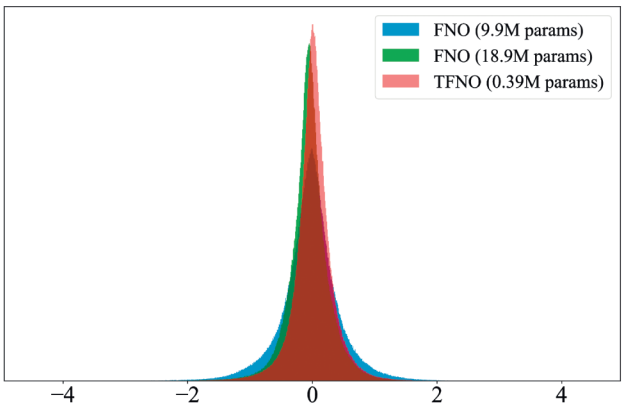

Fig. 5. Histograms of the residuals on test dataset (bar).

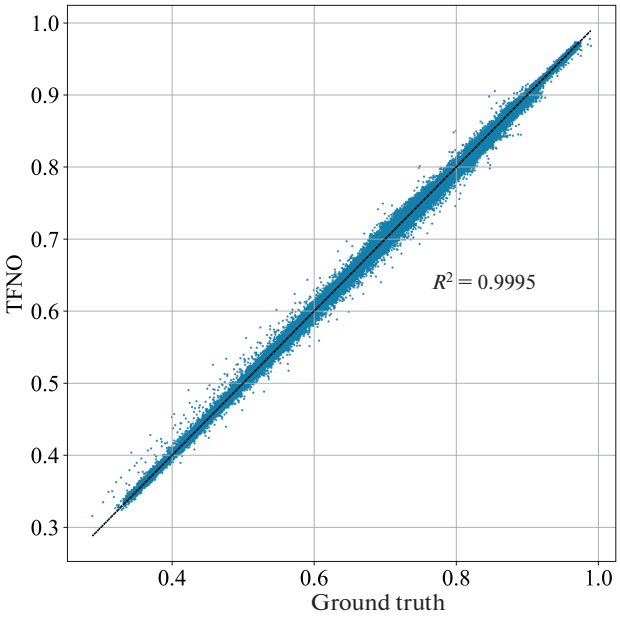

Fig. 6. Scatter plot of scaled reservoir pressure.

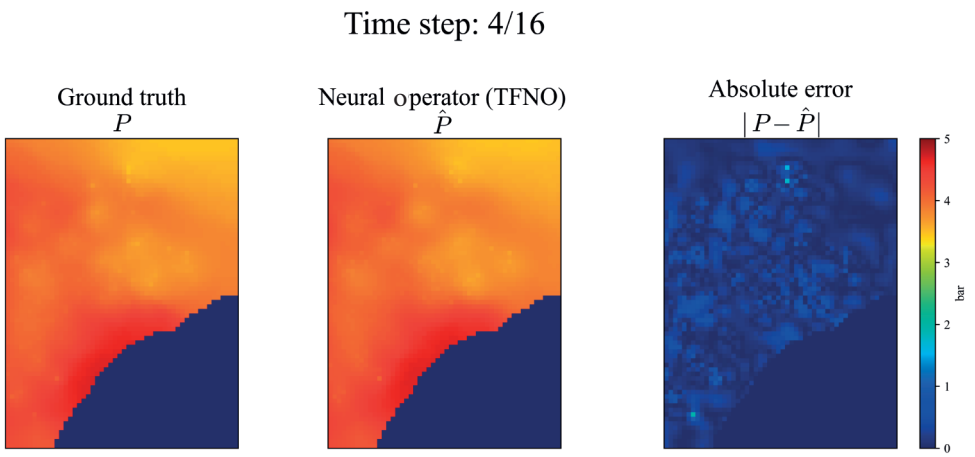

Fig. 7. Visualization of reservoir pressure from numerical model, TFNO model and absolute error on the test sample (time step 4/16).

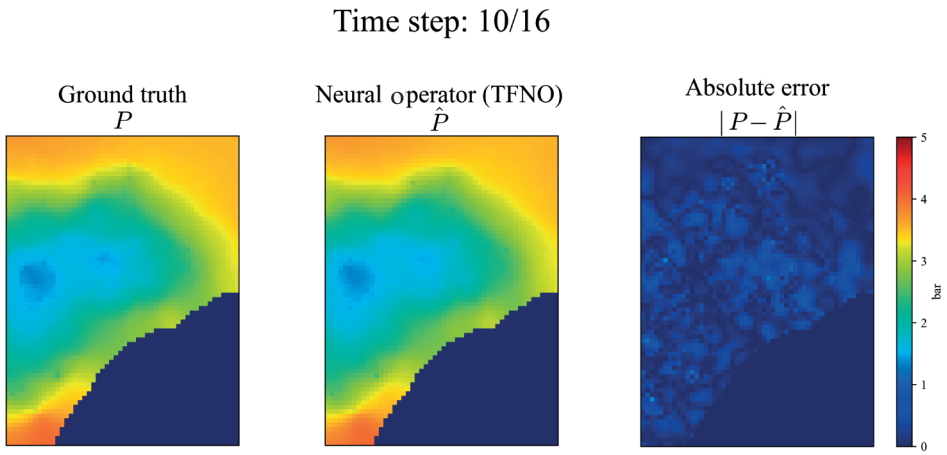

Fig. 8. Visualization of reservoir pressure from numerical model, TFNO model and absolute error on the test sample (time step 10/16).

4.3. Results

In numerical experiments we designed and trained 3 types of neural operator architectures:

1. FNO with the number of Fourier modes limited to 5 and the number of parameters 9.9M.

2. FNO with the number of Fourier modes limited to 16 and the number of parameters 18.9M.

3. TFNO (this paper)—the same as in 2, but with a factorized parameter tensor, having only 2% of the parameters (0.39M).

The models are trained using the H^1 loss function and tested using the relative L_2 error function. Training and validation performance are shown at Fig. 4.

Among trained models, developed in this paper modified Fourier neural operator, in which the model parameters in the spectral domain are represented in the form of factorized low-rank tensors (TFNO) shown superior performance with an order of magnitude lower parameters.

Interestingly, tensor decomposition of the model parameters not only greatly reduced the number of parameters, but also increased the generalization ability of the model. FNO with 18.9 million parameters and TFNO have very similar training errors, but TFNO shows a significantly lower validation error.

The statistical characteristics and histograms of the residuals $(y - \hat{y})$ on the test dataset vs the number of model parameters can be seen in Table 1 and Fig. 5, respectively.

Based on Table 1 and Fig. 5, a comparison of the statistical characteristics of the models residuals

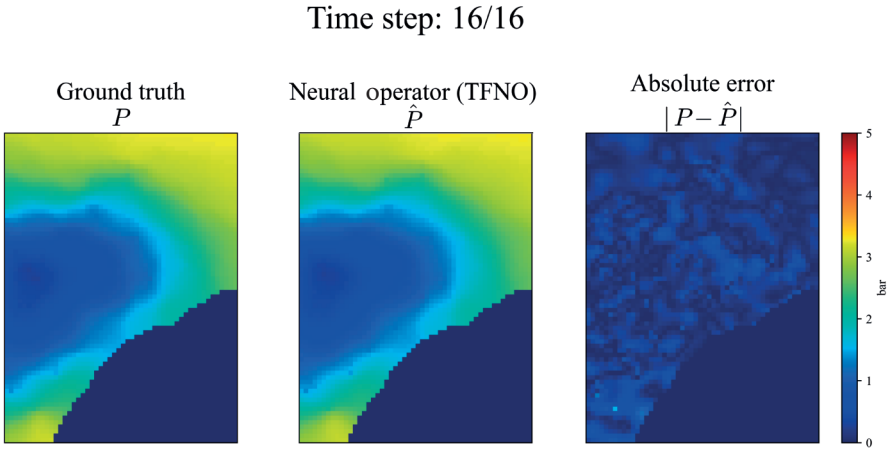

Fig. 9. Visualization of reservoir pressure from numerical model, TFNO model and absolute error on the test sample (time step 16/16).

shows that the neural operator with tensor decomposition of parameters has the best performance on the test data and at the same time has 50 times fewer parameters.

The trained model is capable of quite accurately reproducing the dynamics of reservoir pressure during the withdrawal seasons. Figure 6 shows a scatter plot of normalized (scaled to range from 0 to 1) reservoir pressure between modified Fourier neural operator (TFNO) and the results of numerical simulations on test data.

The scatter plot shows that the distribution generated by the neural operator on the test data in each point of the reservoir is very close to the distribution from the hydrodynamic model. The coefficient of determination $R^2 = 0.9995$. Scatter plot visualizations for FNO (18.9M parameters) and FNO (9.9M parameters) can be seen in the Appendix A.

Comparison of the reservoir pressure field dynamics predicted by neural operator and the numerical simulations (on test data) is presented in Figs. 7–9. The time step denotes the ordinal number of the ten-day period (decade) within the gas extraction season. A fragment of an underground gas storage reservoir is shown (in the i, j plane). The absolute prediction error of the model also has acceptable values. The trained model's output looks smooth and reproduces global and local peculiarities of reservoir pressure distribution.

Note, developed neural operator simulates a given scenario in a fraction of a second, which is at least 10^6 times faster than a traditional numerical solver.

Visualizations of the reservoir pressure field dynamics predicted by FNO (18.9M parameters) and FNO (9.9M parameters) can be seen in the Appendices B, C.

Table 1. Statistical characteristics of residuals on test dataset vs number of trained model parameters

Method	μ (bar)	σ (bar)	No. of params
FNO	−0.03	0.5	9 965 441
FNO	−0.05	0.36	18 898 337
TFNO	**0.02**	**0.35**	**386 529**

5. CONCLUSIONS

In this paper, a deep learning method for hydrodynamic modeling of underground gas storages (UGS) is proposed. A modified Fourier neural operator for hydrodynamic modeling of UGS is developed, in which the model parameters in the spectral domain are represented as factorized low-rank tensors. We trained the model on data obtained from a numerical model of UGS with nonuniform discretization grid, more than 100 wells and complex geometry.

Through comparisons with original Fourier neural operator model it is shown that our method demonstrates superior performance with an order of magnitude (50 times) fewer parameters. Tensor decomposition not only greatly reduced the number of parameters, but also increased the generalization ability of the model. Our numerical experiments show, that developed neural operator simulates a given scenario in a fraction of a second, which is at least 10^6 times faster than a traditional numerical solver, enabling its application in planning and decision-making tasks related to various aspects of UGS operation, such as optimal well utilization, pressure control, and gas inventory management.

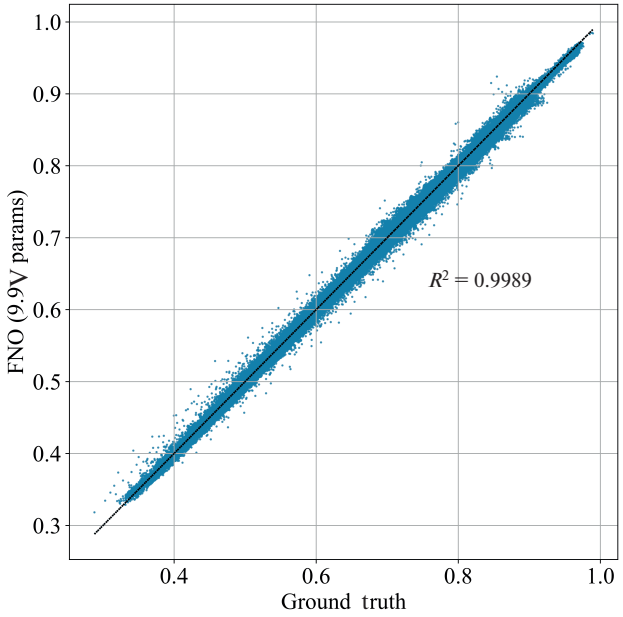

Fig. 10. Scatter plot of scaled reservoir pressure for FNO with 9.9M parameters.

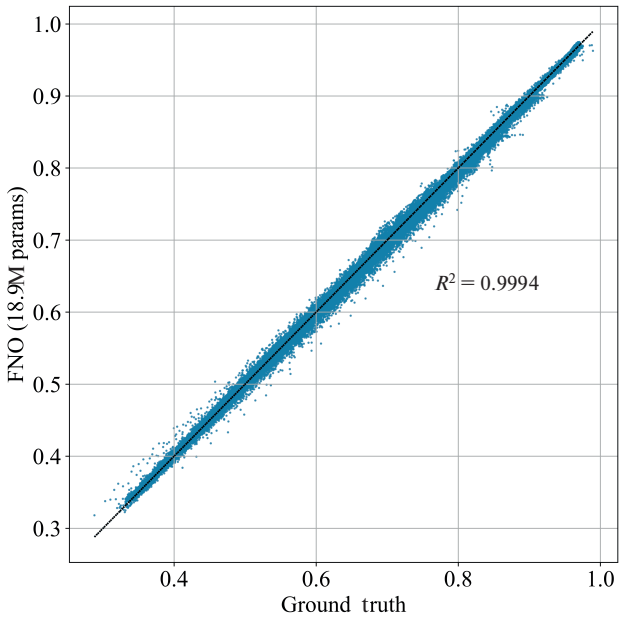

Fig. 11. Scatter plot of scaled reservoir pressure for FNO with 18.9M parameters.

APPENDIX A

COMPARISON OF SCATTERPLOTS BETWEEN NEURAL OPERATORS AND NUMERICAL SIMULATIONS

Scatter plots for trained FNO models with 9.9M parameters and 18.9M parameters can be seen in Figs. 10 and 11, respectively.

APPENDIX B

RESERVOIR PRESSURE COMPARISON (FNO WITH 9.9M PARAMETERS)

Comparison of the reservoir pressure field dynamics predicted by neural operator (FNO with 9.9M parameters) and the numerical simulations (on test data) is presented in Figs. 12–14.

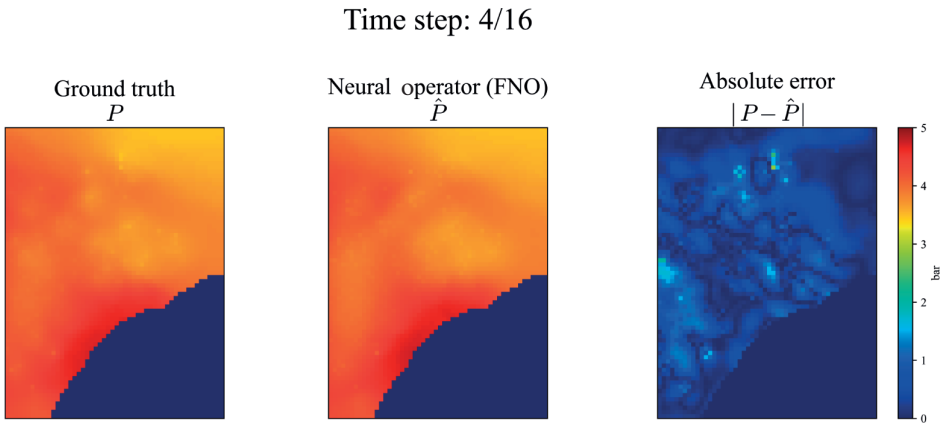

Fig. 12. Reservoir pressure from numerical model, FNO model (9.9M parameters), and absolute error (time step 4/16).

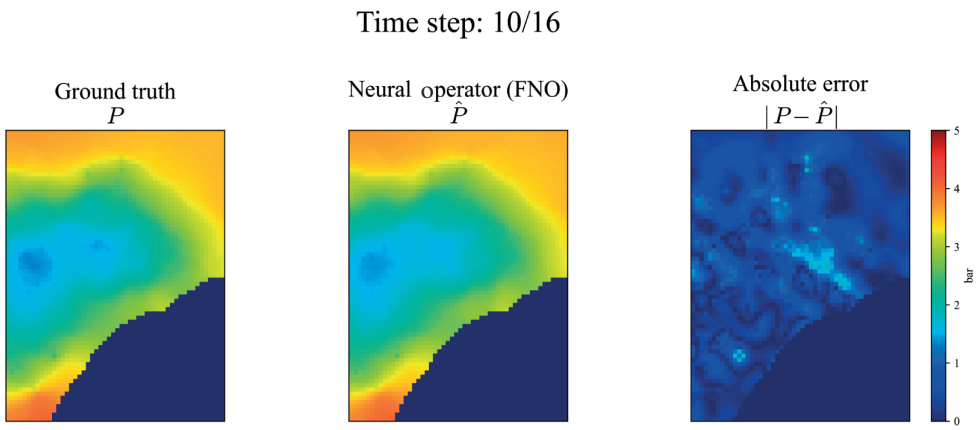

Fig. 13. Reservoir pressure from numerical model, FNO model (9.9M parameters), and absolute error (time step 10/16).

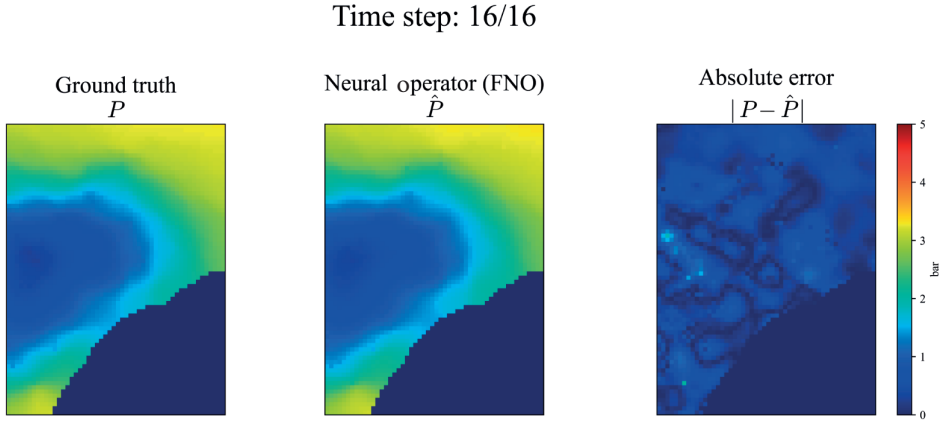

Fig. 14. Reservoir pressure from numerical model, FNO model (9.9M parameters), and absolute error (time step 16/16).

APPENDIX C

RESERVOIR PRESSURE COMPARISON (FNO WITH 18.9M PARAMETERS)

Comparison of the reservoir pressure field dynamics predicted by neural operator (FNO with 18.9M parameters) and the numerical simulations (on test data) is presented in Figs. 15–17.

FUNDING

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

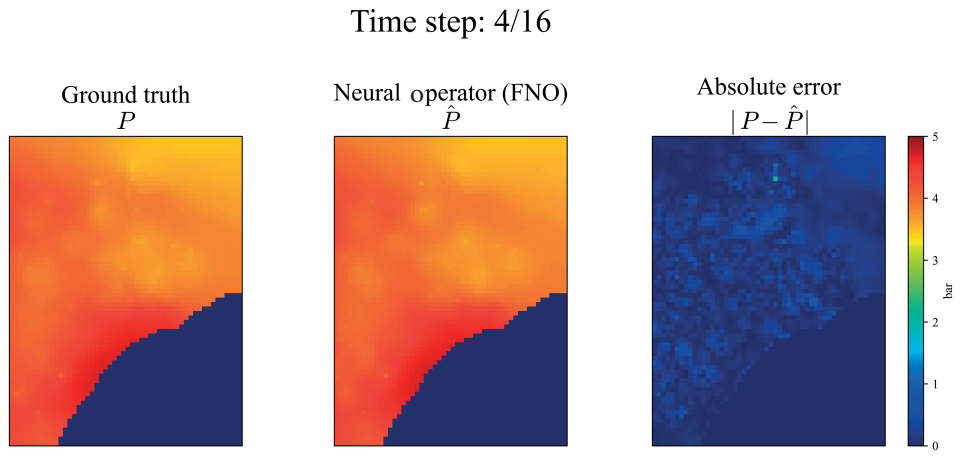

Fig. 15. Reservoir pressure from numerical model, FNO model (18.9M parameters) and absolute error (time step 4/16).

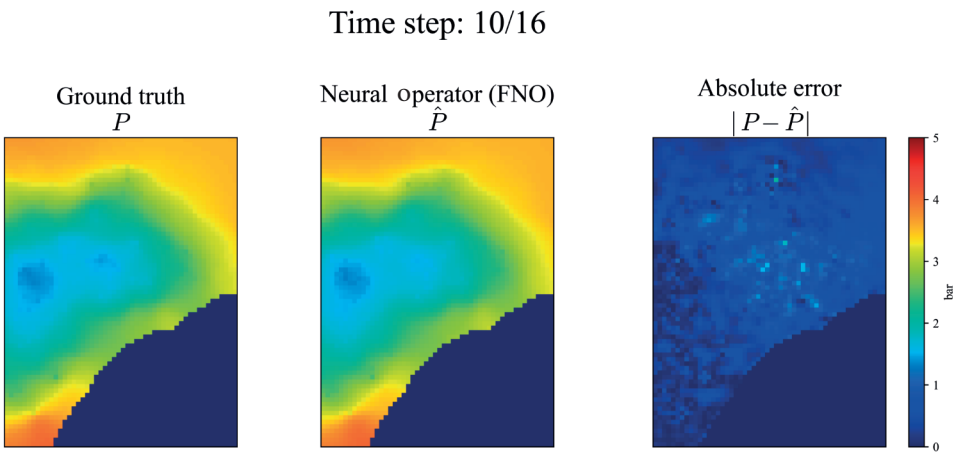

Fig. 16. Reservoir pressure from numerical model, FNO model (18.9M params) and absolute error (time step 10/16).

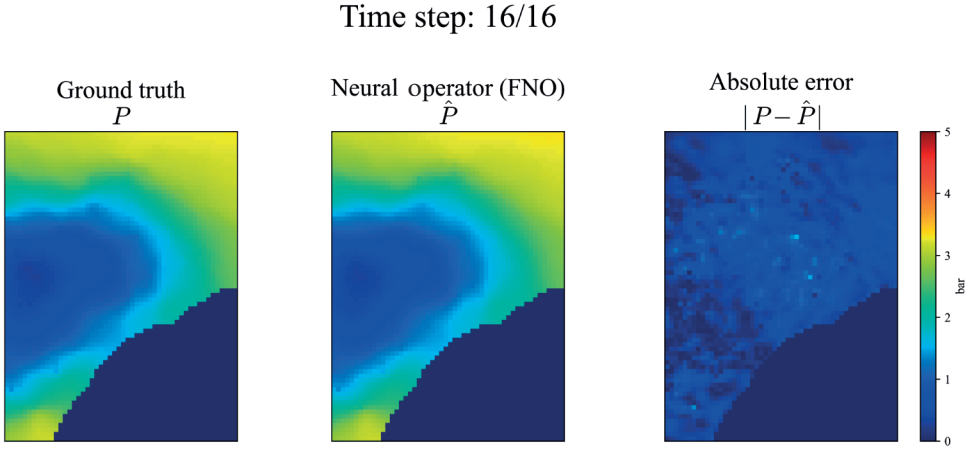

Fig. 17. Reservoir pressure from numerical model, FNO model (18.9M params) and absolute error (time step 16/16).

CONFLICT OF INTEREST

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

REFERENCES

1. Ya. Lecun, Yo. Bengio, and G. Hinton, Nature **521**, 436 (2015). https://doi.org/10.1038/nature14539

2. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (The MIT Press, Cambridge, MA, 2016).
3. K. Aziz and A. Settari, *Petroleum Reservoir Simulation* (Applied Science Publishing, London, 1979).
4. G. Cybenko, Math. Control, Signals Syst. **2**, 303 (1989). https://doi.org/10.1007/bf02551274
5. K. Hornik, M. Stinchcombe, and H. White, Neural Networks **2**, 359 (1989). https://doi.org/10.1016/0893-6080(89)90020-8
6. T. Chen and H. Chen, IEEE Trans. Neural Networks **4**, 910 (1993). https://doi.org/10.1109/72.286886
7. T. Chen and H. Chen, IEEE Trans. Neural Networks **6**, 904 (1995). https://doi.org/10.1109/72.392252
8. T. Chen and H. Chen, IEEE Trans. Neural Networks **6**, 911 (1995). https://doi.org/10.1109/72.392253
9. D. Yarotsky, Neural Networks **94**, 103 (2017). https://doi.org/10.1016/j.neunet.2017.07.002
10. P. Jin, L. Lu, Yi. Tang, and G. E. Karniadakis, Neural Networks **130**, 85 (2020). https://doi.org/10.1016/j.neunet.2020.06.024
11. D. Jakubovitz, R. Giryes, and M. R. D. Rodrigues, in *Compressed Sensing and Its Applications*, Ed. by H. Boche, G. Caire, R. Calderbank, G. Kutyniok, R. Mathar, and P. Petersen, Applied and Numerical Harmonic Analysis (Birkhäuser, Cham, 2019), p. 153. https://doi.org/10.1007/978-3-319-73074-5_5
12. R. Bellman, *Dynamic Programming* (Princeton University Press, Princeton, NJ, 1984).
13. J. Han, A. Jentzen, and W. E, Proc. Natl. Acad. Sci. U. S. A. **115**, 8505 (2018). https://doi.org/10.1073/pnas.1718942115
14. A. Lavin, D. Krakauer, H. Zenil, J. Gottschlich, T. Mattson, J. Brehmer, A. Anandkumar, S. Choudry, K. Rocki, A. G. Baydin, C. Prunkl, B. Paige, O. Isayev, E. Peterson, P. L. McMahon, J. Macke, K. Cranmer, J. Zhang, H. Wainwright, A. Hanuka, M. Veloso, S. Assefa, S. Zheng, and A. Pfeffer, arXiv Preprint (2022). https://doi.org/10.48550/arXiv.2112.03235
15. 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
16. X. Guo, W. Li, and F. Iorio, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, 2016* (Association for Computing Machinery, New York, 2016), p. 481. https://doi.org/10.1145/2939672.2939738
17. M. Tang, Yi. Liu, and L. J. Durlofsky, J. Comput. Phys. **413**, 109456 (2020). https://doi.org/10.1016/j.jcp.2020.109456
18. Zh. Zhong, A. Y. Sun, and H. Jeong, Water Resour. Res. **55**, 5830 (2019). https://doi.org/10.1029/2018wr024592
19. J. Berg and K. Nyström, Neurocomputing **317**, 28 (2018). https://doi.org/10.1016/j.neucom.2018.06.056
20. M. Raissi, P. Perdikaris, and G. E. Karniadakis, arXiv Preprint (2017). https://doi.org/10.48550/arXiv.1711.10561
21. M. Raissi, P. Perdikaris, and G. E. Karniadakis, J. Comput. Phys. **378**, 686 (2019). https://doi.org/10.1016/j.jcp.2018.10.045
22. A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, J. Mach. Learn. Res. **18** (153), 1 (2018). https://www.jmlr.org/papers/v18/17-468.html.
23. L. Lu, P. Jin, G. Pang, Zh. Zhang, and G. E. Karniadakis, Nat. Mach. Intell. **3**, 218 (2021). https://doi.org/10.1038/s42256-021-00302-5
24. Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, arXiv Preprint (2020). https://doi.org/10.48550/arXiv.2003.03485
25. N. Kovachki, Z. Li, B. Liu, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, and A. Anandkumar, J. Mach. Learn. Res. **24** (89), 1 (2023). https://www.jmlr.org/papers/v24/21-1524.html.
26. T. Ertekin, J. H. Abou-Kassem, and G. R. King, *Basic Applied Reservoir Simulation* (Society of Petroleum Engineers, Richardson, TX, 2001).
27. Z. Li, N. B. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, in *International Conference on Learning Representations* (2021). https://openreview.net/forum?id=c8P9NQVtmnO.
28. S. Lanthaler, R. Molinaro, P. Hadorn, and S. Mishra, in *The Eleventh International Conference on Learning Representations* (2023). https://openreview.net/forum?id=CrfhZAsJDsZ.
29. J. Kossaifi, A. Toisoul, A. Bulat, Ya. Panagakis, T. M. Hospedales, and M. Pantic, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020). https://openaccess.thecvf.com/content_CVPR_2020/html/Kossaifi_Factorized_Higher-Order_CNNs_With_an_Application_to_Spatio-Temporal_Emotion_Estimation_CVPR_2020_paper.html.
30. J. Kossaifi, N. B. Kovachki, K. Azizzadenesheli, and A. Anandkumar, in *The Eleventh International Conference on Learning Representations, Kigali, Rwanda, 2023* (2023). https://openreview.net/forum?id=po-oqRst4Xm.

31. L. R. Tucker, Psychometrika **31**, 279 (1966). https://doi.org/10.1007/bf02289464
32. W. M. Czarnecki, S. Osindero, M. Jaderberg, G. Swirszcz, and R. Pascanu, in *Advances in Neural Information Processing Systems*, Ed. by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Curran Associates, 2017), Vol. 30. https://proceedings.neurips.cc/paper_files/paper/2017/file/758a06618c69880a6cee5314ee42d52f-Paper.pdf.