Physically based neural networks for solving transport problems in porous media

*Saeid Khasi** and *Apostolos Kantzas*

Department of Chemical and Petroleum Engineering, University of Calgary, AB, Canada
PERM Inc., Calgary, AB, Canada

Abstract. Recent advances in deep learning have provided high performance frameworks to solve convolution and recursive operations. Based on available resources, this study proposes a neural network-based model for the solution of differential equations that govern the transport in porous media. Such a model replaces the finite difference method and the convergence procedure with pre-specified convolution and recurrent layers in a neural network. The kernels of the convolution neural networks (CNNs) are derived from the implicit discretized form of the governing equations. Such kernels have a window size of 2 and 3 for two- and three-dimensional simulations, respectively. When solving diffusion equation for a 500×500 grid space, the simulation runtime is about 47 % shorter than that of implicit finite difference methods. Considering that the computational complexity of CNNs does not increase quickly and they can be implemented seamlessly on parallel architecture, the proposed approach is promising for simulating large scale problems in porous media. Also, it can provide a generic platform for solving multiphysics problems and deals with irregular arbitrary shaped boundaries effectively which make it a good choice for pore-scale simulations too.

1 Introduction

Modeling transport processes in porous media is crucial for many practical applications including enhancing oil recovery, geothermal energy harvesting, remediation of groundwater, and geological carbon dioxide sequestration [1]. Describing these phenomena usually involves solving partial differential equations. When analytical solutions to the latter equations exist, they often possess inherent precision. However, their utility is limited to special conditions such as simple geometry and boundary conditions [2].

To address the issues with analytical solutions, there are various numerical methods for different types of complex problems such as shooting, finite difference, finite element, finite volume and spline-based methods. Implementing these numerical methods for problems in porous media often requires large space and time complexities. The latter issue exists for both macroscopic and microscopic modeling of porous media due to the large number of computing units in either case. Such complexities are caused by large sizes of geometries and irregular boundary conditions in the field and pore scales, respectively. Due to importance of simulations in porous media, efficient computational methods have been investigated for decades [3]. Recent high performance computing methods based on graphical user processors (GPUs) may help to overcome issues with time consuming simulations.

High performance computing solutions are based on the fact that the resultant matrix from a discretized problem is typically very sparse, meaning it contains a significant number of zeros that can be exploited algorithmically. Using parallel computing techniques to accelerate sparse linear algebra is likely the most popular method to deal with such large-scale scientific computing. However, designing parallel algorithms for sparse matrices that are both efficient and easy to implement is not trivial. Complex algorithms may require significant effort to optimize and may not always yield the desired performance improvements [4]. To avoid the latter issue, we propose a new method to achieve such parallelism in modern computer architectures using a simple and generalized method based on neural network encoding.
Neural networks can be used to encode and solve both ordinary and partial differential equations. In the latter method, computational complexity does not increase quickly when the number of computing points is increased while in the other standard numerical methods computational complexity increases rapidly as we increase the number of points in the interval. Not only the method can be implemented on parallel architectures, but also, it provides a solution with very good generalization properties which make it suitable for porous media where handling the irregular boundaries is challenging [2].

Here, we propose a method to use special types of neural network units to utilize the generalization and speed up properties for solving problems in porous media. Such a method replaces the finite difference discretization

* Corresponding author: saeid.khasi@ucalgary.ca

scheme and the convergence procedure with pre-specified convolution and recurrent layers in a neural network. The kernels of the convolution neural networks (CNNs) are derived from the implicit discretized form of the governing equations. Also, the recurrent units are specified using acceptable tolerances in common numerical methods. In the latter case, a long short-term memory (LSTM) network is suggested to retain most important information of previous convergence behaviors. Results of solving transport equation using traditional finite difference and new method are compared across different scales. Using recent advances in designing optimized parallel architectures for deep learning, the proposed method offers an opportunity to tackle difficult multiphysics differential equation problems arising in porous media in real time. It should be noted that the proposed method in this study is not based on the domain knowledge gained from physics to serve as a guideline for designing a deep learning model as done in previous studies [5]. Instead, it directly encodes physics to utilize available parallelled computational resources developed for deep learning. The proposed model in this study can be converted into a surrogate model by freezing the optimization variables. The latter models are important in many-query applications where computational cost causes a major limitation in utilizing simulation results [6].

The structure of this paper is as follows: In Section 2, methodology is presented by deriving CNN kernels relevant to pore and continuum scales transport equations in porous media. Section 3 describes simulation results obtained by different methods and examine the efficiency, correctness and scalability of the proposed method. Finally, conclusions are listed in Section 4.

2 Methodology

The finite-difference method (FDM) is a powerful technique to solve complex problems. FDM is based on approximating the derivatives by finite difference terms. Using these approximations, ordinary differential equations (ODE) or partial differential equations (PDE), which may be nonlinear, are converted into a system of linear equations that can be solved by matrix algebra techniques. Calculating the finite differences can also be seen as a linear operation and therefore be implemented by multiplication with a convolution kernel.

Without loss of generality, the diffusion equation is used in this section to demonstrate such an operation replacement. But as will be discussed in Section 3, results can be extended. First, the diffusion equation is discretized in both space and time and then CNNs are defined for solving a similar problem.

2.1 Finite difference

The diffusion equation commonly formulated as:

$$\frac{\partial c}{\partial t} = D\nabla^2 c \tag{1}$$

The latter equation models the spread of particle density, c, in a medium over time, where D denotes the diffusion coefficient. This equation is important in different fields such as physics, chemistry, and environmental science; by providing a theoretical basis for phenomena such as heat flow, mass transfer, and pollutant dispersion.

Numerical solutions to the diffusion equation are often based on the finite difference method, which discretizes the continuum of space and time into finite increments (grids). Such discretization is described in the following sub-sections. This discretization transforms the partial differential equation into a system of algebraic equations that can be solved using standard computational techniques.

2.1.1 Spatial discretization

In the discretization, the spatial domain is represented as grids, and the continuous derivatives in the diffusion equation are approximated by differences between neighboring grid points.

For a two-dimensional space with a uniform grid spacing dx in the x-direction and dy in the y-direction, the second derivatives in the Laplacian (∇^2) can be discretized using central difference formulas:

$$\frac{\partial^2 c}{\partial x^2} = \frac{c_{i+1,j} - 2c_{i,j} + c_{i-1,j}}{dx^2} \tag{2}$$

$$\frac{\partial^2 c}{\partial y^2} = \frac{c_{i,j+1} - 2c_{i,j} + c_{i,j-1}}{dy^2} \tag{3}$$

2.1.2 Time discretization

The time derivative can be discretized using methods like the explicit, implicit, or Crank-Nicolson schemes. The implicit approach, as implemented in this study, computes the concentration at the next time step by solving a linear equation system. This scheme is noted for its stability, allowing larger time steps without dependency on spatial discretization.

Accordingly, the explicit (Eq. 4) and implicit (Eq. 5) schemes based on finite difference for a two-dimensional problem with a pulse at center (see Figure 1) can be derived as follows:

$$\frac{c_{i,j}^{n+1} - c_{i,j}^{n}}{\Delta t} = D\left(\frac{c_{i+1,j}^{n} - 2c_{i,j}^{n} + c_{i-1,j}^{n}}{\Delta x^2} + \frac{c_{i,j+1}^{n} - 2c_{i,j}^{n} + c_{i,j-1}^{n}}{\Delta y^2}\right) \tag{4}$$

$$\frac{c_{i,j}^{n+1} - c_{i,j}^{n}}{\Delta t} = D\left(\frac{c_{i+1,j}^{n+1} - 2c_{i,j}^{n+1} + c_{i-1,j}^{n+1}}{\Delta x^2} + \frac{c_{i,j+1}^{n+1} - 2c_{i,j}^{n+1} + c_{i,j-1}^{n+1}}{\Delta y^2}\right) \tag{5}$$

Zero initial value and the Dirichlet boundary conditions are also set for simplicity as can be seen in Figure 1 where model geometry is shown.

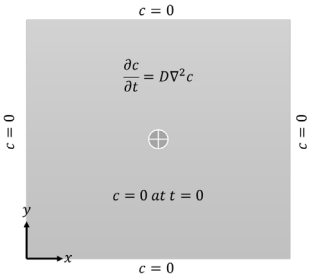

Fig. 1. Model geometry alongside governing equation and side conditions for solving the diffusion equation.

2.2 Neural network

Leveraging CNNs to simulate the diffusion equation may introduce a powerful computational paradigm that utilizes deep learning techniques and highly optimized platforms that are being developed in this area. Unlike traditional methods that require manual discretization of derivatives, CNNs can either learn or encode to solve the equation by minimizing a defined loss function related to the diffusion dynamics.

2.2.1 Convolutional kernel for Laplacian

The first and important step of applying CNNs in this context is the convolutional kernel designed to mimic the Laplacian operator. For a two-dimensional diffusion process, assuming that $\Delta x = \Delta y$, this can be represented by the following expression [7]:

$$K = \begin{bmatrix} 0 & \gamma & 0 \\ \gamma & -4\gamma & \gamma \\ 0 & \gamma & 0 \end{bmatrix} \quad (6)$$

where $\gamma = D\Delta t/\Delta x^2$. This kernel (also known as five-point stencil), when convolved with the concentration matrix, approximates the Laplacian's effect as follows:

$$c_{i,j}^{n+1} - c_{i,j}^{n} = K \cdot \begin{bmatrix} c_{i-1,j-1}^{n} & c_{i-1,j}^{n} & c_{i-1,j+1}^{n} \\ c_{i,j-1}^{n} & c_{i,j}^{n} & c_{i,j+1}^{n} \\ c_{i+1,j-1}^{n} & c_{i+1,j}^{n} & c_{i+1,j+1}^{n} \end{bmatrix} =$$

$$\gamma(c_{i-1,j}^{n} + c_{i,j-1}^{n} - 4c_{i,j}^{n} + c_{i,j+1}^{n} + c_{i+1,j}^{n}) \quad (7)$$

The latter equation can be written in an implicit form and be encoded by the same kernel:

$$c_{i,j}^{n+1} - c_{i,j}^{n} = K \cdot \begin{bmatrix} c_{i-1,j-1}^{n+1} & c_{i-1,j}^{n+1} & c_{i-1,j+1}^{n+1} \\ c_{i,j-1}^{n+1} & c_{i,j}^{n+1} & c_{i,j+1}^{n+1} \\ c_{i+1,j-1}^{n+1} & c_{i+1,j}^{n+1} & c_{i+1,j+1}^{n+1} \end{bmatrix} \quad (8)$$

Eq. (7) and (8) are used to update the concentration values according to the diffusion equation. In the CNN architecture, the implicit form in Eq. (8) is handled by either an optimization loop or introducing recurrent neural networks as it will be discussed in the following subsections. While the derived kernel in Eq. (6) is applicable to two-dimensional systems, a three-dimensional kernel can also be defined for three-dimensional simulations. The 3D Laplacian kernel is typically represented as a 3×3×3 tensor (also known as seven-point stencil). In either case, the input layer (with a shape of $N_x \times N_y \times N_z$ where N_x, N_y, and N_z are system sizes in 3D) is feed into a hidden later via the defined convolution kernel. Then, a loss is calculated and optimized to generate new inputs. Finally, to enforce non-negativity of the calculated concentrations, a rectified linear unit (ReLu) is used as an activation function.

2.2.2 Training and input optimization

Training of the physically informed neural networks (PINN) are usually based on calculating unknown weights of CNN kernels using gradient decent through feeding the simulation results from numerical modeling [8]. In the latter cases, a surrogate model will replace the numerical solver which is just an approximation of the solution function with limitations already imposed by the data from the numerical solution.

Here, the models' variables are defined on the input later of the CNN to predict the next time steps of the solution while the kernel weights are encoded by the derived matrix in Equation (4). In deep learning literature, such an approach is used for optimizing the input images by performing gradient ascent on image. Mainly, input optimization is used to generate new examples for data augmentation purposes. General case of the input optimization is selecting an arbitrary layer of the network as trainable variables which is the main idea behind the gradient based visualization such as saliency maps [9]. The latter maps provide another useful application for the proposed method in this study as it will be discussed in Section 3.

Here, the input layer of the CNN is trained through an iterative process where it learns to predict the evolution of the concentration field by continuously minimizing a loss function. For the loss calculation, the difference between the generated concentration by gradient ascent and the predicted concentration (using the diffusion equation) is calculated, then it is squared and summed to obtain the loss value. This function can also be defined to quantify and imposes the mass and energy balances on top of the discrepancy between the network's predictions and the desired outcomes. It should be noted that any terms (including convection and reaction terms) can be added to the computational graph of the neural network to obtain the predicted value and the outcomes in the optimization loop. A detailed description of the computational graph as well as steps taken to generate new outputs is provided in the appendix.

The boundary conditions are directly encoded during optimization by forcing the associated pixel to values corresponding to boundary conditions. This is done by padding some values to the edges of the matrix known as edge handling in computer vison. Some common techniques to deal with different boundary conditions are clip filter/black (sets the boundary values to zero or a specific "black" value), wrap around (boundary values are taken from the opposite edge), copy edge (values at the boundary are copied from the nearest interior values) and reflect across edge (boundary values are mirrored across the edge). Here, the Dirichlet boundary condition is used for simplicity and therefore the pixel values are set to an initial value of zero on boundaries (clip filter method).

Overall, this approach does away with the need for solving large systems of equations one by one, offering potential efficiencies and enhanced flexibility in handling complex boundary conditions or irregular domains. By integrating deep learning with traditional PDE modeling, this approach not only streamlines computations but also opens new avenues for analyzing and solving complex dynamic systems especially those involved with image type data such as digital rock lab analyses.

2.2.3 LSTM units for convergence

The latter approach based on CNN architecture, requires an optimization loop over each pixel of the input geometry

at each time step. Therefore, to predict the evolution of spatial data (like concentration profiles in the diffusion equation) over time, the choice of hyperparameters such as learning rate would be crucial for each time steps. Also, the convergence criteria in the numerical methods will be replaced by callback parameters for stopping algorithms in the learning process. The result of this work is based on the latter approach since we are only examining the feasibility of replacing numerical methods by deep learning architectures for solving problems in porous media.

To further utilize available deep learning architecture, one could construct a hybrid model that combines the spatial processing capabilities of CNNs with the temporal sequence handling of recurrent neural networks such as LSTM (short for long short-term memory) units. In such a hybrid model, CNN layers will first process the spatial information at each time step to capture spatial dependencies and morphological features. The outputs of these CNN layers will then be fed into LSTM units, which will process the temporal evolution of these features. This sequential processing mimics the propagation of diffusion over time, capturing both the spatial and temporal dynamics inherent in the diffusion equation.

Alternative to hybrid models, one could employ custom LSTM layers such as ConvLSTM where the convolution operation (including a Laplacian-like kernel) is integrated directly within the LSTM architecture. ConvLSTMs combine spatial convolution and LSTM's temporal processing in a single layer, making them ideal for spatio-temporal data [10].

An important consideration in the proposed hybrid model as well as custom LSTM layers (ConvLSTM) would be the need for encoding and modifying the optimized functions in the RNN units. If the operations are not as efficiently optimized as conventional matrix operations in deep learning frameworks, hybrid models could even lead to increased computational demands.

All of the mentioned approaches are dealing with calculating the concentration profiles. One could also use this network as a surrogate model. In the latter case, the optimization parameters in the computational graph can be frozen to predict (instead of calculating) the concentration profiles in future time steps. Again, ConvLSTM architecture should work the best for such an application due to sequential behavior of the concentration profiles over different time steps.

3 Results and Discussion

In this section, simulation results of the diffusion problem are presented using FDM and CNN methods. The input parameters are dimensionless for simplicity.

For FDM, the grid dimensions (N_x and N_y) are varied between 10 to 500, with a diffusion coefficient of $D = 1$. The time step for the simulation is $\Delta t = 0.1$, and the spatial step sizes are $\Delta x = 1$ and $\Delta y = 1$. These parameters determine the resolution and time scale of the simulation. The initial concentration array, c, is set to a concentration of 100 at the center of the grid and 0 everywhere else. The simulations are also run for $N_t = 10$ time steps.

For the CNN based method, the Adam optimizer with a learning rate of 0.01 is employed for the optimization. Again, the simulations are run for 10-time steps, with a tolerance of 0.001 on the loss function variations and a maximum of 500 iterations in the inner optimization loop.

Figures 2-4 show the simulation results of concentration profiles for explicit FDM, implicit FDM and implicit CNN based models, respectively. As can be seen in the latter figures, the simulation results of both implicit methods are similar, and the maximum concentrations are higher than that of obtained by the explicit method.

The reason for this difference in the concentration profiles can be found by checking the mass conservation in the simulations results as shown in Figure 5. As illustrated in the latter figure, the explicit scheme is less mass conservative (with an error of 0.7 % over 10 time-steps) as compared to the implicit schemes (0.26 % for FDM and 0.28 % CNN based method).

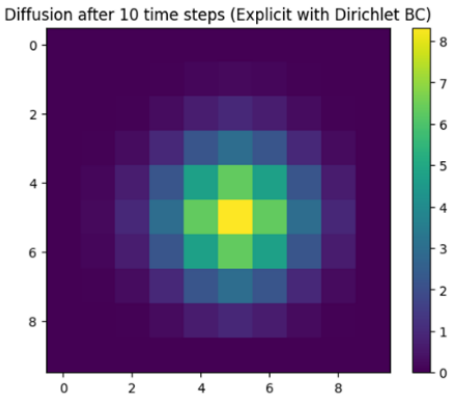

Fig. 2. Concentration profiles obtained by solving the diffusion problem using explicit scheme of finite difference on a 10×10 gird size.

Regardless of the error manganite in Figure 5, behavior of error in mass conservation between CNN based and FDM suggests a different computation algorithm in the latter two cases.

One of the reasons to use neural network-based formulations is possible speed up for large scale or high-resolution problems at it can be implemented on parallel computational architectures such as graphical processing units (GPUs). Unlike the central processing units (CPUs) where one operation at a time is handled, GPU can process thousands of threads simultaneously. Therefore, it is expected to observe significant speed up as number of grids (pixels) are increased when such a parallel architecture is used for the computations. To examine this hypothesis, experiments can be run at different grid sizes and the simulation runtimes be compared. For example, Figure 6 shows simulation results of the concentration profile when the size of the system is increased to a 20×20 pixel-size. It also shows the loss values obtained after 200 epoch in each time step of calculations. The latter loss

curves will be discussed in more detail in the following when convection terms are introduced.

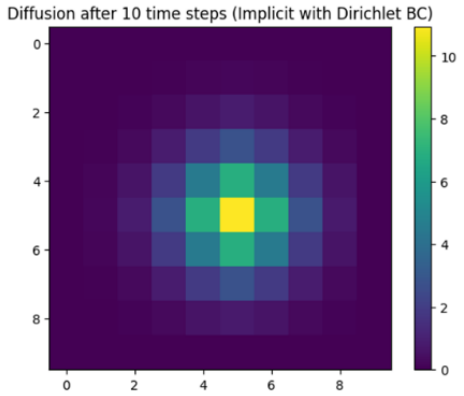

Fig. 3. Concentration profiles obtained by solving the diffusion problem using implicit scheme of finite difference on a 10×10 grid size.

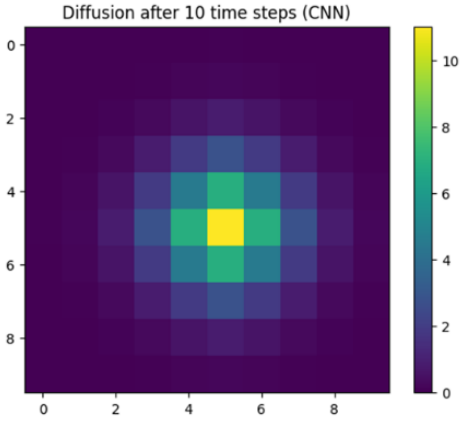

Fig. 4. Concentration profiles obtained by solving the diffusion problem using implicit scheme of CNN based model on a 10×10-pixel size.

Similar to the results in Figure 6, we run simulations using three different approaches on a Tesla T4 GPUs in Google Colab computer machine. The simulation runtimes are reported in Table 1 for system sizes of 10×10, 20×20, 50×50, 100×100, 200×200 and 500×500. Each reported value is an average of 5 different runs to minimize random errors caused by available online computational power at a given time.

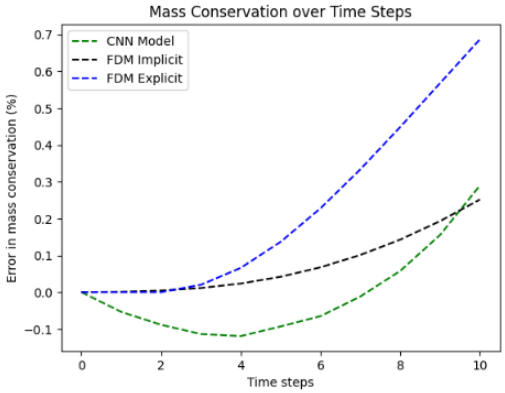

Fig. 5. Summation of total mass over all grids (pixels) in the simulations' models.

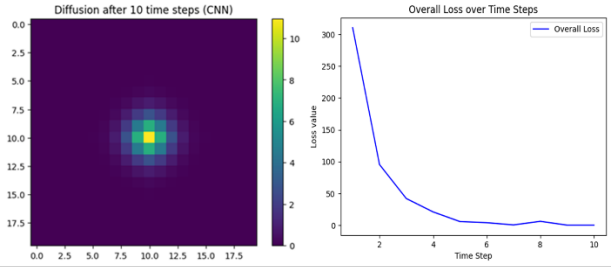

Fig. 6. (a) Concentration profiles obtained by solving the diffusion problem using implicit scheme of CNN based model on a 20×20-pixel size. (b) loss values obtained after 200 epoch in each time step of calculations.

Table 1 shows that as the grid sizes expand to 200×200, the runtime of the CNN-based model stabilizes due to the parallel processing capabilities of the GPU. Although the implicit FDM initially outperforms in terms of runtime for smaller system sizes, its advantage over the CNN-based model lessens when the system size reaches 500×500.

The key observation in Table 1 is that the computational complexity does not increase significantly in neural network methods when the number of pixels is increased. This contrasts sharply with standard numerical methods, where computational complexity escalates rapidly as more grids are added. This difference is critical in scenarios where high-resolution data are needed, making neural network methods particularly advantageous for handling large-scale problems efficiently.

In Table 1, a trivial observation would be that the explicit method outperforms both implicit approaches since the equations are not solved simultaneously in an explicit scheme, and the computing process only includes basic operations and assignments.

Table 1. Simulation runtime (in seconds) by adopting different approaches for solving the diffusion problem at different sizes.

Size	CNN Implicit	FDM Implicit	FDM Explicit
10×10	12	0.06	0.002
20×20	18	0.4	0.002
50×50	30	3.2	0.003
100×100	65	8	0.007
200×200	70	28	0.008
500×500	71	133	0.03

Another aspect of neural network-based methods is the property of generalization. As can be seen in derivation of the CNN kernels, they can be written for different governing equations. Coupling and extending different equations can be readily encoded by modifying the loss function and connecting the output layers of different neural networks which make the proposed method suitable for developing generic multiphysics modeling. For example, adding other transport terms to the solved diffusion equation in previous section only require modifying the loss function.

Figures 7-8 shows simulation results of concentration profiles obtained by solving the diffusion problem with convective term and velocity of 1 in x and y, respectively. As can be seen in the latter figures, diffusion propagations are affected by horizontal and vertical velocity directions through convolution operation on the input images.

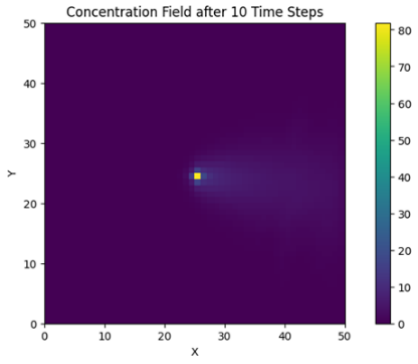

Fig. 7. Concentration profiles obtained by solving the diffusion problem with convective term and velocity of u=1 in x-direction using implicit scheme of CNN based model on a 50×50-pixel size.

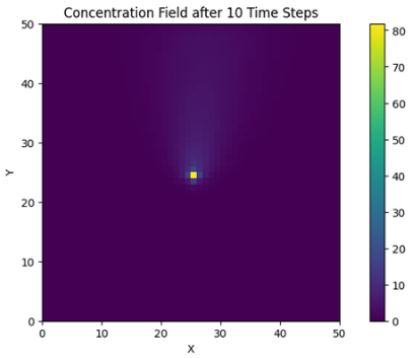

Fig. 8. Concentration profiles obtained by solving the diffusion problem with convective term and velocity of v=1 in y-direction using implicit scheme of CNN based model on a 50×50-pixel size.

For introducing the convection terms in Figures 7 and 8, only two additional kernels are needed to compute the first derivatives in the x and y directions. To do so, the kernel in Eq. (6) remains unchanged from the diffusion-only model and is used to compute the second spatial derivatives of the concentration field. Two one-dimensional gradient kernels can also be defined as follows:

$$G_x = \gamma \Delta x / D [-1 \quad 0 \quad 1] \tag{9}$$

$$G_y = \gamma \Delta y / D [-1 \quad 0 \quad 1]^T \tag{10}$$

The latter kernels in Eqs. (9) and (10) are used to approximate the first derivative with respect to x and y, respectively, in the convection term of the transport equation. In other words, we replace the prediction of each neural network layer in Eq. (11) by the one presented in Eq. (12)

$$prediction_{layer} = c^n + K \times c^{n+1} \tag{11}$$

$$prediction_{layer} = c^n + K \times c^{n+1} - u \times G_x \times c^{n+1} - v \times G_y \times c^{n+1} \tag{12}$$

Similar to other numerical methods, convergence issues should arise when the ratio of the convection to the diffusion (also known as Peclet number) is increased. Figure 9 shows the concentration profiles after 20-time steps when velocity in y-direction is increased from 1 (Figure 9(a)) to 3 (Figure 9(b)).

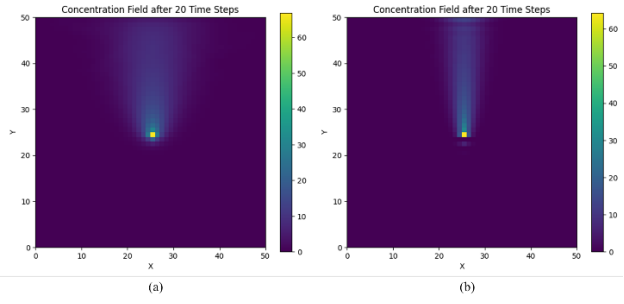

Fig. 9. Concentration profiles obtained by solving the diffusion problem with convective term and velocity of (a) v=1 and (b) v=3 in y-direction using implicit scheme of CNN based model on a 50×50-pixel size.

Figure 9 shows how increasing the convective term can change the concentration profiles. To show the impacts of a higher Peclet number on numerical

convergence, the corresponding overall loss values at the end of 200 epochs are plotted against the time steps Figure 10. The latter figure shows that a higher ratio of the convective term to the diffusion term generated a higher value of the loss. A higher loss indicates a higher convergence error. Therefore, the proposed method will need higher computational cost at higher Peclet number similar to previous numerical methods.

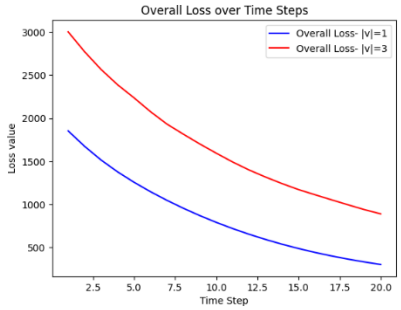

Fig. 10. Loss values at the end of 200 epochs during optimization of the concentration profiles using implicit scheme of CNN based model on a 50×50-pixel size.

In addition to developing a simulator for large scale or high-resolution problem, the suggested approach in this study can be used to interpret the results of pre-train model by finding their kernels and try to retrieve the underlying equations [11]. One example would be history matching in reservoir engineering where to use data-driven approaches for characterizations. The idea of such an application would be similar to saliency maps generated for visualizing the training process of neural networks.

Conclusions

In this work, the problem of solving the diffusion equation using an implicit finite difference scheme was mapped to the task of optimization a neural network. Simulation experiments were run to examine the accuracy, performance and scalability of the proposed approach.

The mass conservation analysis showed that the accuracy of the proposed method is comparable to that of achieved in the implicit finite differences approach. After a large enough gird size (500×500 in the simulations results in this study), neural network-based models could outperform conventional implicit finite differences method in terms of simulation runtime. In fact, after a gird size of 200×200, runtime of simulation by new approach seems to level off.

The simulation results were obtained based on the simplest form of the diffusion equations in a two-dimensional system with a simple boundary condition. But the proposed encoded neural networks can be extended into more complex equations to incorporate other terms in transport equations such as convection and reaction expressions. The model can also be upscaled to three-dimensional systems with irregular boundaries by merely changing the applied kernel and subsequent modifying the loss function.

As a future work, neural network with recurrent layers can be used to further utilize the computational efficiency of available frameworks for deep learning. The underlying concept for the latter task was provided in this work. Finally, to provide a comparison between the proposed method with traditional parallel computing techniques, future studies should focus on evaluating speedups and numerical performance measures such as compression efficiency. The numerical performance of a problem with parallel computing depends on number of processors and types of machines (whether it is a multicore/manycore shared memory machines or distributed memory machines like clusters and supercomputers).

Financial support from the Energi Simulation Centre for Geothermal Systems Research, The Energy Harvesitng Processes Program (ConocoPhillips, Ashaw Energy, Kalina Distributed Power, Telsec, Remedy Services, The Alberta Geological Survey and The Geological Survey of Canada) and the Fundamentals of Unconventional Resources program (NSERC, Alberta Innovates, Chevron, CNRL, ConocoPhillips and Enerplus) is gratefully acknowledged.

References

1. J. Bear, Y. Bachmat, *Introduction to modeling of transport phenomena in porous media* **4**, 263-285 (2012).
2. N. Yadav, A. Yadav, and M. Kumar, *An introduction to neural network methods for differential equations* **1**, 1-11 (2015).
3. M. Sahimi, D. Stauffer, Chem. Eng. Sci. **46**, 9 (1991).
4. G. Xiao, C. Yin, T. Zhou, X. Li, Y. Chen, K. Li, ACM Comput. Surv. **56**, 1 (2023).
5. E. De Bézenac, A. Pajot, P. Gallinari, J. Stat. Mech: Theory Exp. **2019**, 12 (2019).
6. N. R. Franco, S. Fresca, F. Tombari, A. Manzoni, Chaos **33**, 12 (2023).
7. J. A. Actor, D. T. Fuentes, and B. Rivière, *Proceedings of SPIE--the International Society of Optical Engineering* **11313**, 17 (2020).
8. C. G. Fraces, A. Papaioannou, H. Tchelepi, arXiv prepr. arXiv **2001**, 05172 (2020).
9. K. Simonyan, A. Vedaldi, A. Zisserman, arXiv prepr. arXiv **1312**, 6034 (2013).
10. X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, W. Woo, *Advances Neural Information Processing Systems* **28**, 802-810 (2015).
11. P. Guo, K. Huang, and Z. Xu, arXiv prepr. arXiv 2103, 08313 (2021).

Appendix

The textual representation of the computational graph for the basic developed neural network is shown in Figure A.1. In the latter figure, the computations steps include initialization, outer and inner loop calculations and loss and mass evaluations. In the initialization, initial concentration is reshaped into tensor. For each time step in the outer loop, a copy of the tensor is stored. In the inner

loop, the optimization is performed (using Adam optimizer) until the loss is below the tolerance value or the maximum number of iterations is reached. For the loss calculation, the difference between newly generated profile and the predicted concentration (using the diffusion equation) is calculated, then it is squared and summed to obtain the loss value. The gradients of the loss with respect to the tensor variable are used to generate new concentration profiles. To enforce non-negativity on the tensor variables, the ReLu activation function is applied after applying the Laplacian kernel. Finally, the current loss and mass deviations are appended to the losses and masses lists.

Fig. A1. Computational graph for the neural network used in place of finite difference scheme.