Physics-informed neural networks for solving coupled flow and transport system

Sanghyun Lee [1] **Teeratorn Kadeethum,**[2]

[1]Department of Mathematics, Florida State University
1017 Academic Way
Tallahassee, FL 32304
[2]Sibley School of Mechanical and Aerospace Engineering, Cornell University
130 Upson Hall
Ithaca, NY 14853
lee@math.fsu.edu, tk658@cornell.edu

Abstract

In this paper, a direct application of the physics-informed neural networks to consider the coupled flow and transport system as a forward solver is presented. We address the classical challenge of solving the coupled system with multiple variables involving the convection-dominated regime of the transport. The comparisons of the approximated solutions from neural networks and the exact solutions are presented with the sensitivity analyses regarding the hyper-parameters and number of the training data.

Introduction

Since pioneering work in 1943 on neural networks (NN) based on a brain model (McCulloch and Pitts 1943), the development of NN and deep learning (DL) in the form of deep neural networks (DNNs) have been accelerated with the help of big data analytic and the advancement of powerful computational resources. Despite the numerous successes obtained with DL, limitations remain concerning applications in scientific problems. In many scientific and engineering problems, collecting a large amount of data to guarantee model accuracy is expensive and often not possible (Ahmed, Jones, and Marks 2015). In addition, the training of the DL model is only based on the available data, and no physical laws are involved during the training, which may lead to physically unreasonable predictions (Xiao et al. 2016; Wang, Wu, and Xiao 2017; Raissi, Perdikaris, and Karniadakis 2019; Kutz 2017; Wang et al. 2018; Zhang et al. 2019b).

To overcome the above limitations of NN and DL, scientific machine learning (SciML) based on physical sciences incorporates scientific knowledge and physics-based partial differential equations (PDEs) into DL architectures. New approaches that solve PDEs based on DL include a deep Ritz (Weinan 2017; Weinan and Yu 2018), PDE-Net (Long, Lu, and Dong 2019), a deep Galerkin (Sirignano and Spiliopoulos 2018), variational Galerkin (Kharazmi, Zhang, and Karniadakis 2019; Khodayi-Mehr and Zavlanos 2019), Bayesian deep convolutional networks (Zhu et al. 2019), a deep domain decomposition (Li et al. 2019), theory-guided data science (Karpatne et al. 2017a), a physics-guided NN (Karpatne et al. 2017b), a theory-guided NN (Wang et al. 2020), physics-informed NN (Raissi, Perdikaris, and Karniadakis 2019; Kadeethum, Jørgensen, and Nick 2020b,a), and others (Owhadi 2015; Jagtap, Kharazmi, and Karniadakis 2020; Lu et al. 2019; Raissi, Yazdani, and Karniadakis 2020; D'Elia et al. 2020; Kadeethum, Jørgensen, and Nick 2020b; Fraces, Papaioannou, and Tchelepi 2020). The requirement of large amount of training examples is mitigated using these new research idea, and the physical laws are naturally embedded through systems of PDEs (Lu et al. 2019; Raissi, Yazdani, and Karniadakis 2020; D'Elia et al. 2020; Raissi, Perdikaris, and Karniadakis 2019; Kadeethum, Jørgensen, and Nick 2020a).

In this paper, we utilize the idea of physics-informed NN (PINN) to solve a coupled flow and transport system in porous media (Fraces, Papaioannou, and Tchelepi 2020; Cai et al. 2020; He et al. 2020). Sensitivity analyses in respect to the hyper parameters (number of layers and neurons) and the number of training data points are discussed. Moreover, we present the performance of the PINN for solving the advection-dominated transport system when the flux is also computed by PINN.

Computational Algorithms

In this section, we introduce the governing system and the main ingredients of PINN with a rationale based on previous studies (Raissi, Perdikaris, and Karniadakis 2019).

Governing equations

We consider an initial-boundary value problem of the coupled flow and transport problem in the computational domain $\Omega \subset \mathbb{R}^2$ where the time domain is denoted by $\mathbb{T} = (0, \mathrm{T}]$ with a given final time $T > 0$. The coupled system derived from the conservation of mass (volume) is defined as the following

$$\begin{cases} -\nabla \cdot (\kappa \nabla p) &= f, \\ \mathbf{v} &:= -\kappa \nabla p, \\ \dfrac{\partial}{\partial t} c + \nabla \cdot (\mathbf{v} c) &= g, \end{cases} \tag{1}$$

where the unknown variables are the scalar pressure function (p) and the scalar transport function (c). Here, f and g are the body forces for each equations, and $\mathbf{v}$ is the velocity vector defined with the given coefficient κ, which is assumed to be a constant in this paper for the simplicity.

The boundary condition on $\partial\Omega$ for the pressure equation can be decomposed to pressure (Dirichlet) and flux (Neumann) boundaries, $\partial\Omega_p$ and $\partial\Omega_q$, respectively. Also, the transport equation is supplemented by both Dirichlet and Neumann boundaries, $\partial\Omega_c$ and $\partial\Omega_r$, respectively. Moreover the initial condition for the concentration $c(\cdot, t = 0) = c_0$ is given.

Physics-Informed Neural Networks

Recently developed Physics-Informed Neural Networks (PINN) (Raissi, Perdikaris, and Karniadakis 2019) seek the solutions satisfying PDEs by utilizing the residuals of each equation in the governing system and boundary/initial conditions as part of the training. Here, the solution of PDEs is formulated as the solution to a constrained optimization problem. A couple of main advantages to this approach include i) the size of the training set is considerably reduced by utilizing the governing equations as an implicit regularization term in an objective function to be minimized (D'Elia et al. 2020), and ii) it is a mesh-free approach, where the NN is trained on batches of randomly sampled time and space points. In particular, PINNs have been further extended to solve fractional PDEs (Pang, Lu, and Karniadakis 2019), stochastic PDEs (Nabian and Meidani 2018; Yang, Zhang, and Karniadakis 2020; Zhang, Guo, and Karniadakis 2020; Zhang et al. 2019a), and nonlocal models (D'Elia et al. 2020).

Neural network architecture An example of a neural network architecture is presented in Figure 1 (Rumelhart, Hinton, and Williams 1986; LeCun, Bengio, and Hinton 2015; Hinton and Salakhutdinov 2006). The number of input and output nodes in the NNs shown in this figure are determined from the given formulation of the problem. For example, if the problem is solving a time dependent PDE, we have three input nodes (x, y and t), where t is time, x and y are coordinates in x- and y-directions, respectively (i.e $i = 3$ in Figure 1). The output nodes will be the solution functions satisfying the given system of PDEs. In the coupled flow and transport problem as defined in (1), we have two output nodes c and p, where c represents the concentration, and p is the pressure. Thus, we have $k = 2$ in Figure 1.

The number of hidden layers (N_{hl}) and the number of neurons (N_n) act as hyper-parameters, which means they are problem-specific and needed to be adjusted according to the natures of each problem (Goodfellow, Bengio, and Courville 2016). Each neuron (e.g., $\mathrm{H}_{1,1}$ ... $\mathrm{H}_{1,\mathrm{N_n}}$) is connected to the nodes of the previous layer with adjustable weights (W) and also has an adjustable bias (b). These variables are learned during a training phase (Hinton and Salakhutdinov 2006; Goodfellow, Bengio, and Courville 2016) by utilizing the hyperbolic tangent ($\tanh$) as an activation function, and second-order Limited-memory BFGS as an optimizer by minimizing loss functions (LOSS).

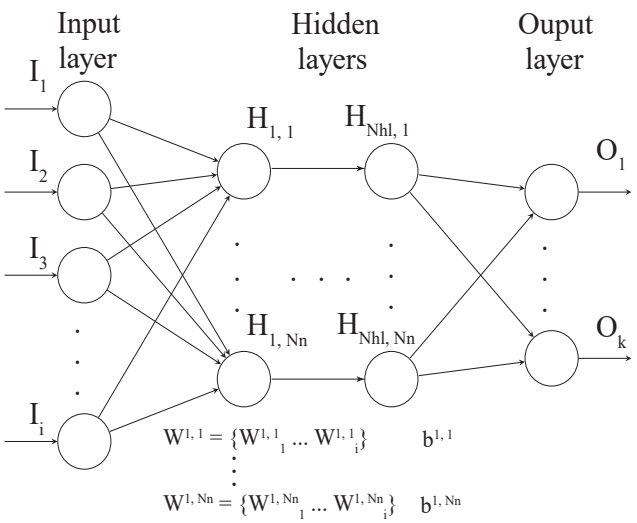


Figure 1: An example of general neural network architecture (Rumelhart, Hinton, and Williams 1986; LeCun, Bengio, and Hinton 2015; Hinton and Salakhutdinov 2006). The input layer contains up to i input nodes, and the output layer is composed of $1, ..., k$ output nodes. N_{hl} refers to the number of hidden layers, and each hidden layer is composed of N_n neurons. Each neuron (e.g., $\mathrm{H}_{1,1}$ ... $\mathrm{H}_{1,\mathrm{N_n}}$) is connected to the nodes of the previous layer with adjustable weights and also has an adjustable bias. This figure is adapted from (Kadeethum, Jørgensen, and Nick 2020b,a)

Based on the algorithm established in (Raissi, Perdikaris, and Karniadakis 2019; Lu et al. 2019), the PINN utilizes two NNs. One NN is employed to impose the boundary/initial conditions of the system with the adjustable W and b, and the other NN is for the information given by the differential operators as regularizing terms of the loss functions for each PDEs. The latter loss functions could be defined by the residual of the PDEs. Thus, the training examples that are used to evaluate these extra regularizing terms are different from those used to train the network with the boundary/initial conditions (BC/ICs). In other words, BC/ICs are imposed using training data and the equation residuals are imposed in a data-free manner by automatic differentiation of the predicted states at randomly chosen interior collocation points. This additional NN with the residuals as loss functions are known as the PINN (Raissi, Perdikaris, and Karniadakis 2019). In order to minimize PINN, we adjust all the W and b from the first NN. Throughout this paper, the NNs are built on the Tensorflow platform (Abadi et al. 2015) and the numerical code is built on DeepXDE (Lu et al. 2019).

Loss functions The loss functions (LOSS), which we minimize through the machine learning process, is composed of two parts as discussed in the previous section. One is the error on the training data (MSE_{tr}), which includes the boundary and initial conditions for the first NN. The other one is the mean square value of the regularization term given

by the physics-informed functions (MSE_Π).

We encode the underlying physical information to the NNs through the so-called physics-informed functions (Π) as following,

$$\Pi_p := -\nabla \cdot (\kappa \nabla p) - f, \tag{2}$$

$$\Pi_c := \frac{\partial}{\partial t} c + \nabla \cdot ((-\kappa \nabla p) c) - g, \tag{3}$$

which are the residuals of the given PDEs. These residuals (Π_p, Π_c) will act as the additional regularizing terms in the loss function defined below.

Thus, the loss function of our problem is

$$\text{LOSS} := \text{MSE}_{tr} + \text{MSE}_{\Pi_p} + \text{MSE}_{\Pi_c}, \tag{4}$$

where the boundary and initial conditions are incorporated into the MSE_{tr}. Here, the value of the mean squared error (MSE) is defined as $\text{MSE} := \frac{1}{n}\sum_{i=1}^{n}(\phi(\mathbf{x}_i, t_i) - \phi_h(\mathbf{x}_i, t_i))^2$ for a given function ϕ and an approximated function ϕ_h.

Numerical Results

In this final section, we present and discuss the capabilities and the performance of the algorithm by solving the the coupled flow and transport system shown in (1).

Example 1

First, to test the accuracy of the proposed algorithm, we set the exact solutions as

$$c(x, y, t) := \sin(t + x + y), \tag{5}$$

and

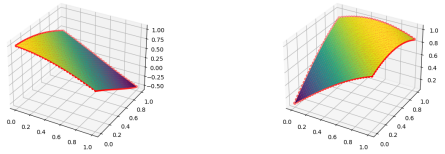
$$p(x, y, t) := \cos(t + x + y), \tag{6}$$

for the transport and the pressure, respectively, in the computational domain $\Omega \times \mathbb{T} = (0, 1)^2 \times (0, 1]$. Here, the body forces f and g are computed with the given solutions where κ is chosen to be $\kappa = 1$, and the Dirichlet boundary conditions are given on $\partial\Omega$ for both pressure and transport.

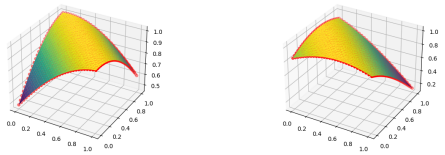
Figure 2 illustrates the approximated solutions of pressure (p) at time $t = 0.1$ and transport (c) at time $t = 0.1, 0.5$ and $t = 1$ by utilizing PINN. In addition, Figure 3 presents the comparisons with the exact solution over the line for each given time steps. We observe that the PINN with the given loss functions provide accurate approximations to the multi-variable coupled flow and transport system. We note that the algorithm does not have the time marching steps.

Here, the number of the training data points for approximating the initial condition ($N_{\Omega(t=0)}$), the boundary conditions ($N_{\partial\Omega}$), and inside the domain (N_Ω) are all set to be 1000. The number of hidden layers is 4, and the number of neurons for each layer is 10. The second-order Limited-memory BFGS method is employed for the optimization, and the number of *epochs* (iterations of training) is set to be $10,000$. Moreover, $\tanh$ function is utilized for the activation function.

Next, Table 1 illustrates the sensitivity test regarding to the choice of the hyper-parameters (number of hidden layers

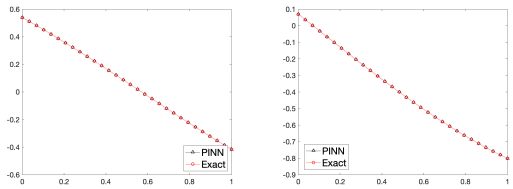


(a) p at $t = 0.1$. (b) c at $t = 0.1$.

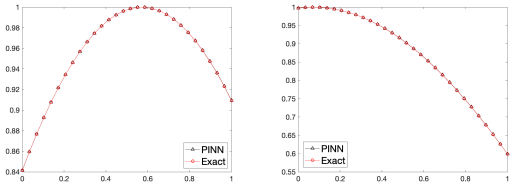


(c) c at $t = 0.5$. (d) c at $t = 1$.

Figure 2: (a) illustrates the approximated pressure value p at $t = 0.1$, and (b)-(d) are the approximated transport values c for each time $t = 0.1, 0.5$ and $t = 1$.



(a) p at $t = 0.5$ (b) p at $t = 1$.



(c) c at $t = 0.5$ (d) c at $t = 1$.

Figure 3: The comparison of the approximated pressure and transport solutions, p, c, with the exact solution over the line $(0, 0.5)$-$(1, 0.5)$.

(N_{hl}) and neurons (N_n)). Here, the error is computed by the following definition,

$$\|\phi(\mathbf{x}, t) - \phi_h(\mathbf{x}, t)\|_{L_\infty} := \max_i \left(|\phi(\mathbf{x}_i, t_i) - \phi_h(\mathbf{x}_i, t_i)|\right),$$

where ϕ denotes the exact solution, and ϕ_h is the approximated solution, which is either pressure or transport in this case. In addition, for this test, the number of training points for the initial/boundary conditions $(N_{\Omega(t=0)}, N_{\partial\Omega})$ and inside the domain for the residual (N_Ω) are fixed to be 1000. We observe that the algorithm does depend on these hyper-parameters, but the results do not vary too much.

Moreover, Table 2 presents the algorithm's sensitivity test regarding the number of training points for the initial/boundary conditions and inside the domain for the residual. Here the hyper-parameters are fixed to be $N_{hl} = 4$ and

$N_n \backslash N_{hl}$	4	8	16
10	2.65e-03	2.19e-03	7.79e-04
20	6.23e-04	1.86e-03	7.61e-04
40	3.32e-03	8.89e-04	1.75e-03
$N_n \backslash N_{hl}$	4	8	16
10	1.47e-03	3.11e-03	1.08e-03
20	6.10e-04	1.05e-03	9.60e-04
40	5.00e-03	2.56e-03	2.89e-03

Table 1: Comparison of the error depending on number of the hidden layers and the neurons. The top table is for the pressure and the bottom one is for the transport. The row indicates different number of hidden layers (N_{hl}) and the columns are for different number of neurons N_n.

$N_n = 20$. To provide a general result, the average of 5 different realizations are shown in the table.

$(N_{\Omega(t=0)}, N_{\partial\Omega}) \backslash N_\Omega$	10	100	1000
(10,10)	1.25e-03	1.04e-03	1.44e-03
(100,100)	1.04e-03	9.99e-04	6.40e-04
(1000,1000)	1.56e-03	8.02e-04	1.08e-03
$(N_{\Omega(t=0)}, N_{\partial\Omega}) \backslash N_\Omega$	10	100	1000
(10,10)	4.73e-03	2.06e-03	2.32e-03
(100,100)	1.23e-03	1.94e-03	1.08e-03
(1000,1000)	1.48e-03	1.20e-03	1.20e-03

Table 2: Comparison of the error depending on number of the training points. The top table is for the pressure and the bottom one is for the transport. The column indicates different $(N_{\Omega(t=0)}, N_{\partial\Omega})$ and the rows are for N_Ω.

Example 2

In the final example, we solve a simple flow and transport problem in $\Omega \times \mathbb{T} = (0,1)^2 \times (0,1]$ by setting $f = g = 0$. The boundary conditions for the transport and the pressure are given as

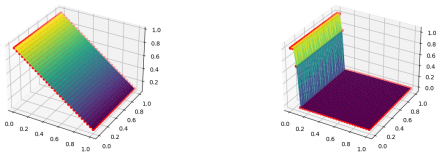
$$\begin{cases} c & = 1, \text{ if } x = 0, \\ \nabla c \cdot \mathbf{n} & = 0, \text{ else where,} \end{cases}$$

and

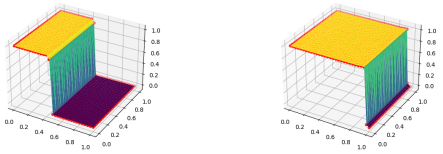
$$\begin{cases} p & = 1, \text{ if } x = 0, \\ p & = 0, \text{ if } x = 1, \\ \kappa \nabla p \cdot \mathbf{n} & = 0, \text{ else where .} \end{cases}$$

The initial condition for the transport equation is set to be $c(\cdot, t = 0) = 0$. Thus, we are transporting the $c = 1$ from the left boundary to the right boundary with the computed velocity. Here, the hyper-parameters are fixed to be $N_{hl} = 4$ and $N_n = 20$, and $N_{\Omega(t=0)} = N_{\partial\Omega} = N_\Omega = 1000$.

Figure 4 illustrates the solution of the pressure p and the transport c for each given time. We note that the pressure and the velocity are constant over the time domain, as shown in Figure 4(a). However the value of c is transported from left to the right as expected (Figure 4(b)-(d)). The solutions of c over the line $(0, 0.5) - (1, 0.5)$ for each time $t = 0.1, 0.3, 0.5, 0.7$ and 0.9 are plotted in the same domain in Figure 5. We do not observe any spurious oscillations or over/under shooting (violation of the maximum principle) in this advection dominated case (Wang, Teng, and Perdikaris 2020; Fuks and Tchelepi 2020).



(a) p at $t = 0.1$ (b) c at $t = 0.1$



(c) c at $t = 0.5$ (d) c at $t = 1$.

Figure 4: The solution of the pressure p and the transport c for each given time step.

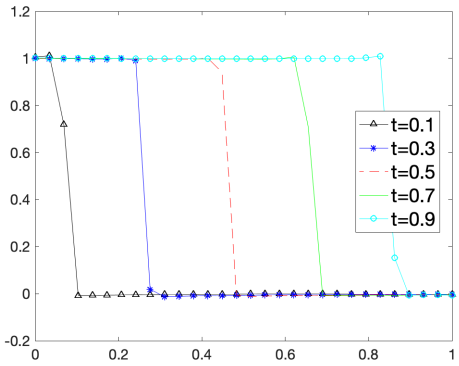


Figure 5: Solution of c over the line $(0, 0.5) - (1, 0.5)$ is plotted for each time $t = 0.1, 0.3, 0.5, 0.7$ and 0.9. We do observe the moving step functions as expected.

Conclusion

This paper utilizes PINN to solve one of the multi-physics problems, coupled flow and transport systems. The sensitivity tests regarding to the parameters for training NN and the loss functions for PINN are presented. The numerical experiments illustrate the accuracy and the capabilities of the algorithm. Detailed comparison with the existing numerical methods such as finite element method, and extension to consider nonlinear problems in heterogeneous media are ongoing work.

References

Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.; Davis, A.; Dean, J.; Devin, M.; et al. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems .

Ahmed, E.; Jones, M.; and Marks, T. 2015. An improved deep learning architecture for person re-identification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3908–3916.

Cai, S.; Wang, Z.; Lu, L.; Zaki, T. A.; and Karniadakis, G. E. 2020. DeepM&Mnet: Inferring the electroconvection multiphysics fields based on operator approximation by neural networks. *arXiv preprint arXiv:2009.12935* .

D'Elia, M.; Parks, M. L.; Pang, G.; and Karniadakis, G. 2020. nPINNs: nonlocal Physics-Informed Neural Networks for a parametrized nonlocal universal Laplacian operator. Algorithms and Applications. Technical report, Sandia National Lab.(SNL-NM), Albuquerque, NM (United States).

Fraces, C. G.; Papaioannou, A.; and Tchelepi, H. 2020. Physics Informed Deep Learning for Transport in Porous Media. Buckley Leverett Problem. *arXiv preprint arXiv:2001.05172* .

Fuks, O.; and Tchelepi, H. A. 2020. Limitations of physics informed machine learning for nonlinear two-phase transport in porous media. *Journal of Machine Learning for Modeling and Computing* 1(1).

Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. *Deep learning*. MIT press.

He, Q.; Barajas-Solano, D.; Tartakovsky, G.; and Tartakovsky, A. M. 2020. Physics-informed neural networks for multiphysics data assimilation with application to subsurface transport. *Advances in Water Resources* 141: 103610.

Hinton, G.; and Salakhutdinov, R. 2006. Reducing the dimensionality of data with neural networks. *science* 313(5786): 504–507.

Jagtap, A. D.; Kharazmi, E.; and Karniadakis, G. E. 2020. Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering* 365: 113028.

Kadeethum, T.; Jørgensen, T.; and Nick, H. 2020a. Physics-informed Neural Networks for Solving Inverse Problems of Nonlinear Biot's Equations: Batch Training. In *54th US Rock Mechanics/Geomechanics Symposium*. Golden, CO, USA: American Rock Mechanics Association.

Kadeethum, T.; Jørgensen, T.; and Nick, H. 2020b. Physics-informed neural networks for solving nonlinear diffusivity and Biot's equations. *PLoS ONE* 15(5):e0232683.

Karpatne, A.; Atluri, G.; Faghmous, J. H.; Steinbach, M.; Banerjee, A.; Ganguly, A.; Shekhar, S.; Samatova, N.; and Kumar, V. 2017a. Theory-guided data science: A new paradigm for scientific discovery from data. *IEEE Transactions on knowledge and data engineering* 29(10): 2318–2331.

Karpatne, A.; Watkins, W.; Read, J.; and Kumar, V. 2017b. Physics-guided neural networks (pgnn): An application in lake temperature modeling. *arXiv preprint arXiv:1710.11431* .

Kharazmi, E.; Zhang, Z.; and Karniadakis, G. 2019. Variational Physics-Informed Neural Networks For Solving Partial Differential Equations. *arXiv preprint arXiv:1912.00873* .

Khodayi-Mehr, R.; and Zavlanos, M. M. 2019. VarNet: Variational neural networks for the solution of partial differential equations. *arXiv preprint arXiv:1912.07443* .

Kutz, N. 2017. Deep learning in fluid dynamics. *Journal of Fluid Mechanics* 814: 1–4.

LeCun, Y.; Bengio, Y.; and Hinton, G. 2015. Deep learning. *nature* 521(7553): 436.

Li, K.; Tang, K.; Wu, T.; and Liao, Q. 2019. D3M: A deep domain decomposition method for partial differential equations. *IEEE Access* .

Long, Z.; Lu, Y.; and Dong, B. 2019. PDE-Net 2.0: Learning PDEs from data with a numeric-symbolic hybrid deep network. *Journal of Computational Physics* 399: 108925.

Lu, L.; Meng, X.; Mao, Z.; and Karniadakis, G. E. 2019. DeepXDE: A deep learning library for solving differential equations. *arXiv preprint arXiv:1907.04502* .

McCulloch, W.; and Pitts, W. 1943. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics* 5(4): 115–133.

Nabian, M. A.; and Meidani, H. 2018. A deep neural network surrogate for high-dimensional random partial differential equations. *arXiv preprint arXiv:1806.02957* .

Owhadi, H. 2015. Bayesian numerical homogenization. *Multiscale Modeling & Simulation* 13(3): 812–828.

Pang, G.; Lu, L.; and Karniadakis, G. E. 2019. fpinns: Fractional physics-informed neural networks. *SIAM Journal on Scientific Computing* 41(4): A2603–A2626.

Raissi, M.; Perdikaris, P.; and Karniadakis, G. 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics* 378: 686–707.

Raissi, M.; Yazdani, A.; and Karniadakis, G. E. 2020. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science* 367(6481): 1026–1030.

Rumelhart, D.; Hinton, G.; and Williams, R. 1986. Learning representations by back-propagating errors. *nature* 323(6088): 533–536.

Sirignano, J.; and Spiliopoulos, K. 2018. DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics* 375: 1339–1364.

Wang, J.; Wu, J.; and Xiao, H. 2017. Physics-informed machine learning approach for reconstructing Reynolds stress modeling discrepancies based on DNS data. *Physical Review Fluids* 2(3).

Wang, N.; Zhang, D.; Chang, H.; and Li, H. 2020. Deep learning of subsurface flow via theory-guided neural network. *Journal of Hydrology* 584: 124700.

Wang, S.; Teng, Y.; and Perdikaris, P. 2020. Understanding and mitigating gradient pathologies in physics-informed neural networks. *arXiv preprint arXiv:2001.04536* .

Wang, Z.; Xiao, D.; Fang, F.; Govindan, R.; Pain, C.; and Guo, Y. 2018. Model identification of reduced order fluid dynamics systems using deep learning. *International Journal for Numerical Methods in Fluids* 86(4): 255–268.

Weinan, E. 2017. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics* 5(1): 1–11.

Weinan, E.; and Yu, B. 2018. The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics* 6(1): 1–12.

Xiao, H.; Wu, J.; Wang, J.; Sun, R.; and Roy, C. 2016. Quantifying and reducing model-form uncertainties in Reynolds-averaged Navier–Stokes simulations: A data-driven, physics-informed Bayesian approach. *Journal of Computational Physics* 324: 115–136.

Yang, L.; Zhang, D.; and Karniadakis, G. E. 2020. Physics-informed generative adversarial networks for stochastic differential equations. *SIAM Journal on Scientific Computing* 42(1): A292–A317.

Zhang, D.; Guo, L.; and Karniadakis, G. E. 2020. Learning in modal space: Solving time-dependent stochastic PDEs using physics-informed neural networks. *SIAM Journal on Scientific Computing* 42(2): A639–A665.

Zhang, D.; Lu, L.; Guo, L.; and Karniadakis, G. 2019a. Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems. *Journal of Computational Physics* 397: 108850.

Zhang, Z.; Song, X.; Ye, S.; Wang, Y.; Huang, C.; An, Y.; and Chen, Y. 2019b. Application of deep learning method to Reynolds stress models of channel flow based on reduced-order modeling of DNS data. *Journal of Hydrodynamics* 31(1): 58–65.

Zhu, Y.; Zabaras, N.; Koutsourelakis, P.-S.; and Perdikaris, P. 2019. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *Journal of Computational Physics* 394: 56–81.