Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 September 2020 doi:10.20944/preprints202006.0258.v2

Data-driven solutions and discoveries in mechanics using physics informed neural network

Qi Zhang[a,*], Yilin Chen[a], Ziyi Yang[b]

[a]*Department of Civil and Environmental Engineering, Stanford University, Stanford, CA 94305, USA*
[b]*Department of Mechanical Engineering, Stanford University, Stanford, CA 94305, USA*

Abstract

Deep learning has achieved remarkable success in diverse computer science applications, however, its use in other traditional engineering fields has emerged only recently. In this project, we solved several mechanics problems governed by differential equations, using physics informed neural networks (PINN). The PINN embeds the differential equations into the loss of the neural network using automatic differentiation. We present our developments in the context of solving two main classes of problems: data-driven solutions and data-driven discoveries, and we compare the results with either analytical solutions or numerical solutions using the finite element method. The remarkable achievements of the PINN model shown in this report suggest the bright prospect of the physics-informed surrogate models that are fully differentiable with respect to all input coordinates and free parameters. More broadly, this study shows that PINN provides an attractive alternative to solve traditional engineering problems.

Keywords: Conservation laws, Data inference, Data discovery, Dimensionless form, PINN

1. Introduction

It is well-known that many mechanics phenomenons are governed by conservation laws which can be converted to differential equations for most of the applications. Solving these differential equations and obtaining parameter estimations from given observations are always intriguing topics, and significant research has been done to develop many advanced (semi-)analytical or numerical algorithms. While in the last decade, machine learning especially neural networks have yielded revolutionary results across diverse disciplines, including image and pattern recognition, natural language processing, genomics, and material constitutive modeling [15, 23, 26, 34], among which a fair amount of research has also been done related to differential equations [1, 9, 12, 13, 17, 18, 19, 22, 24, 25, 27, 28, 29, 30, 31, 32, 33, 35, 36, 38], owing to the dramatic increase in the computing resources. Therefore, it will be interesting to adopt neural networks as an important alternative to traditional mathematical methods to approximate the solutions to differential equations through iterative update of the network weights and biases.

*Corresponding author
Email address: qzhang94@stanford.edu (Qi Zhang)

© 2020 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 September 2020 doi:10.20944/preprints202006.0258.v2

2. Related work

Early researches that used neural network in scientific computing could date back to [12, 22], in which a single hidden layer neural network was adopted

$$N = \tilde{u} + \sum_{i=1}^{H} v_i \sigma_i \left(\sum_{j=1}^{n} w_{ij} x_j + u_i \right), \tag{1}$$

where N is the network output which is assumed to be a scalar for illustration purposes, H is the width of the hidden layer, and n is the dimension of the input vector $\boldsymbol{x} \in \mathbf{R}^n$, w_{ij} and v_i are called weights while u_i and $\tilde{u}$ are always denoted as the biases, $\sigma_i(\cdot)$ is called activation function. More recently, the development of automatic differentiation [2] has led to emerging literature on the use of deep neural networks to solve differential equations [17, 33, 38, 39] even in complex geometries, and Berg et al. [3] provided details of this back propagation algorithm for advection and diffusion equations. Unlike traditional machine learning methods, deep neural networks sometimes can overcome the curse of dimensionality [17]. In the work done by Raissi et al [30, 31, 32], they named such strong form approach for differential equation as the physics-informed neural network (PINN) for the first time. Today, the PINN becomes more and more popular in many engineering fields such as hydrogeology [7, 18, 35], geomechanics [19], cardiovascular system [21] and so on. One attractive feature of PINN is that it is fully differentiable with respect to all the input coordinates and free parameters, in other words, the trial solution (via the trained PINN) represents a smooth approximation that can be evaluated and differentiated continuously on the domain [9]. Another important feature of PINN is that it can be used to solve inverse (data discovery) problems [16] with minimum change of the code from the forward problems [24].

It is also worth to mention that there are also other machine learning methods that didn't rely on the neural network to solve differential equations [28, 29], but in this report, we will focus on the PINN.

3. Methods

In this section, we will give a very brief overview of PINN in order to demonstrate how the loss function $\mathcal{L}$ is defined in such context. We refer interested readers to look at [32] for more details. Consider a system of differential equations defined on the domain Ω with boundary $\partial\Omega$:

$$\mathcal{D}(\boldsymbol{u}(\boldsymbol{x})) = \mathbf{0} \qquad \boldsymbol{x} \in \Omega, \tag{2}$$

$$\mathcal{B}(\boldsymbol{u}(\boldsymbol{x})) = \mathbf{0} \qquad \boldsymbol{x} \in \partial\Omega, \tag{3}$$

where $\boldsymbol{u}$ is the unknown solution vector, $\mathcal{D}$ denotes the abstract (non-linear) differential operator (*e.g.*, $\partial/\partial\boldsymbol{x}$, $\boldsymbol{u} \circ \partial/\partial\boldsymbol{x}$, $\boldsymbol{x} \circ \partial/\partial\boldsymbol{x}$, higher order derivatives, non-linear functions of $\boldsymbol{u}$ or $\boldsymbol{x}$, their combinations, *etc.*), and the operator $\mathcal{B}$ expresses arbitrary boundary conditions (*e.g.*, Dirichlet, Neumann, Robin, *etc.*) associated with the problem. Note here for time-dependent problems, we consider t as a special component of $\boldsymbol{x}$, *i.e.*, the Ω contains the temporal domain. The initial condition can be simply treated as a special type of Dirichlet boundary condition on the spatio-temporal domain [24].

Now we construct a neural network with L layers, or $L-1$ hidden layers to approximate $\boldsymbol{u}(\boldsymbol{x})$, the input to neural network is $\boldsymbol{x}$, and we will denote the output of the neural network as $\hat{\boldsymbol{u}}(\boldsymbol{x};\theta)$ (we want $\hat{\boldsymbol{u}}(\boldsymbol{x};\theta) \approx \boldsymbol{u}(\boldsymbol{x})$), where θ represents the collection of weight matrix $W^{[l]} \in \mathbf{R}^{n_l \times n_{l-1}}$ and bias vector $b^{[l]} \in \mathbf{R}^{n_l}$ for each layer l with n_l neurons. In the inverse or data discovery problems, there are some unknown parameters γ in

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 September 2020 doi:10.20944/preprints202006.0258.v2

the original differential equations, thus we will rewrite our approximation as $\hat{\boldsymbol{u}}(\boldsymbol{x};\theta,\gamma)$. Now we can define the physics-informed loss $\mathcal{L}$ as

$$\mathcal{L}(\theta,\gamma) = w_f\mathcal{L}_f(\theta,\gamma) + w_b\mathcal{L}_b(\theta,\gamma) + w_d\mathcal{L}_d(\theta,\gamma)\ , \tag{4}$$

where w_f is the weight for the loss due to mismatch with the governing equations

$$\mathcal{L}_f(\theta,\gamma) = \frac{1}{|\Gamma_f|}\sum_{\boldsymbol{x}\in\Gamma_f\subset\Omega} \|\mathcal{D}(\hat{\boldsymbol{u}}(\boldsymbol{x};\theta,\gamma))\|_2^2\ , \tag{5}$$

and w_b is the weight for the loss due to mismatch with the boundary conditions

$$\mathcal{L}_b(\theta,\gamma) = \frac{1}{|\Gamma_b|}\sum_{\boldsymbol{x}\in\Gamma_b\subset\partial\Omega} \|\mathcal{B}(\hat{\boldsymbol{u}}(\boldsymbol{x};\theta,\gamma))\|_2^2\ , \tag{6}$$

and w_d is the weight for the loss due to mismatch with the given data observations $\boldsymbol{u}^*(\boldsymbol{x})$

$$\mathcal{L}_d(\theta,\gamma) = \frac{1}{|\Gamma_d|}\sum_{\boldsymbol{x}\in\Gamma_d\subset\Omega} \left\|\hat{\boldsymbol{u}}(\boldsymbol{x};\theta,\gamma) - \boldsymbol{u}^*(\boldsymbol{x})\right\|_2^2\ . \tag{7}$$

In practice, Γ_f and Γ_b are always known as the set of locations of the "residual" points, and Γ_d is known as the set of measurement locations. We then optimize θ and γ together, and our solution is

$$\theta^*,\gamma^* = \underset{\theta,\gamma}{\operatorname{argmin}}\,\mathcal{L}(\theta,\gamma)\ . \tag{8}$$

Above process is approximately summarized in Fig. 1. Due to the network architecture and incorporation of differential equations, the loss will be highly non-linear and also non-convex, thus we will need some gradient-based optimizers, such as steepest descent [4], Newton's method [4], Adam [20], and L-BFGS (limited-memory quasi-Newton method) [5]. In this report, we will adopt purely Adam or Adam combined with L-BFGS optimizers (*i.e.*, we first use Adam for a certain number of iterations, and then switch to L-BFGS until convergence).

For estimation of the test error, we will evaluate the loss $\mathcal{L}_f$ on a given test set $\Gamma_{\text{test}} \subset \Omega$. In addition, if the true solution is available, we will also calculate the mean square error (similar to the form of $\mathcal{L}_d$) on the same test set.

4. Experiments, results and discussions

4.1. One dimensional consolidation problem

In soil mechanics, one of the most important processes is the consolidation process, while under some specific assumptions made by Prof. Terzaghi and Prof. Biot [6, 8, 37], we can mathematically quantify this process as a one dimensional diffusion problem [14] with specific boundary conditions and initial conditions. Suppose our problem domain is $0 \le x \le L$ where L can be understood as the thickness of the aquifer that rests on a rigid impermeable base, at time $t = 0$, surface load p_0 is applied on the top of the aquifer which will lead to an initial pore pressure field $p^{\text{u}} = \gamma_l p_0$ at $t = 0^+$ where γ_l is called loading efficiency [37]. As time goes on, the pressure p will gradually decrease due to the effect of the drainage boundary at $x = 0$, and it satisfies following partial differential equation

$$\frac{\partial p}{\partial t} - c\frac{\partial^2 p}{\partial x^2} = 0\,, \tag{9}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 September 2020 doi:10.20944/preprints202006.0258.v2

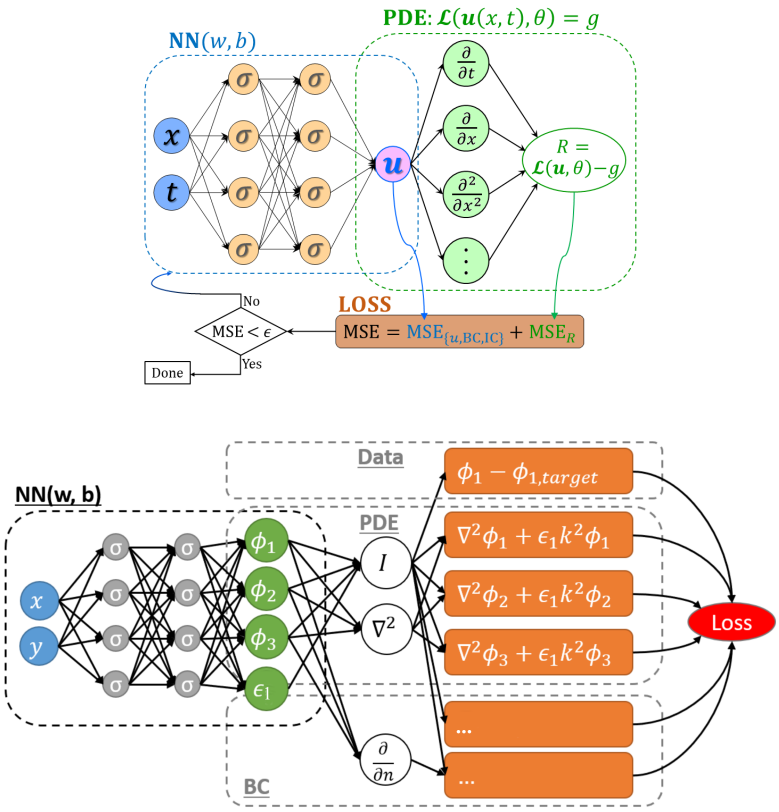

Figure 1: Two schematics of the physics-informed neural network (PINN) [24, 25].

with boundary conditions $p = 0$ at $x = 0$ and $\partial p/\partial x = 0$ at $x = L$. The initial condition is just $p = p^{\mathrm{u}}$ as mentioned before. In above Eq. (9), the c is called generalized consolidation coefficient [11] which has the unit of length squared divided by time.

However, the order of magnitudes of different physical quantities x (order of meters), t (order of months) and p (order of kPa) could cast great difficulty on the training of the neural networks [21], and to overcome this problem, we employ a non-dimensionalization technique with the purpose of scaling the input and the output of the neural network in proper scales (*e.g.*, $O(1)$). As a result, we will introduce characteristic values for x and p, and for this problem an obvious choice would be characteristic length L and characteristic pressure p^{u}. At this point we define the dimensionless quantities as

$$\hat{x} = \frac{x}{L} \qquad \hat{p} = \frac{p}{p^{\mathrm{u}}}\,. \tag{10}$$

For the time t, the dimensionless time $\hat{t}$ is given as

$$\hat{t} = \frac{ct}{L^2}\,. \tag{11}$$

Now all the variables take values with $O(1)$ magnitude and we will rewrite Eq. (9) using dimensionless

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 September 2020 doi:10.20944/preprints202006.0258.v2

variables, the result is shown as follows

$$\begin{aligned} \frac{\partial \hat{p}}{\partial \hat{t}} - \frac{\partial^2 \hat{p}}{\partial \hat{x}^2} &= 0 \qquad 0 < \hat{x} < 1 \qquad \hat{t} > 0 \\ \hat{p} &= 0 \qquad \hat{x} = 0 \qquad \hat{t} \geq 0 \\ \frac{\partial \hat{p}}{\partial \hat{x}} &= 0 \qquad \hat{x} = 1 \qquad \hat{t} > 0 \\ \hat{p} &= 1 \qquad 0 < \hat{x} \leq 1 \qquad \hat{t} = 0\,. \end{aligned} \tag{12}$$

The analytical solution for Eq. (12) is given as

$$\hat{p} = \sum_{m=1,3,5,7,\ldots}^{\infty} \frac{4}{m\pi} \sin\left(\frac{m\pi\hat{x}}{2}\right) \mathrm{e}^{-\frac{m^2\pi^2\hat{t}}{4}}\,. \tag{13}$$

For the PINN hyper-parameters, we assume $\hat{t}_{\max} = 0.5$, and we draw 500, 250, 125 "residual" points randomly from $(0,1) \times (0, \hat{t}_{\max})$, boundary of $\hat{x}$ and boundary of $\hat{t}$ respectively to form Γ_f and Γ_b. The test set Γ_{test} will have 10011 equispaced points. The loss weights are assigned as $w_f = w_b = 1/4$ (w_d is not used in this problem). For the architecture of the neural network, it will contain 5 hidden layers with 50 neurons each, and we will use hyperbolic tangent `tanh` as our activation function in hidden layers, the output activation will be linear activation but we will multiply the network output with $\hat{x}$ as our final output, because in this way our output will automatically satisfy the second expression in Eq. (12). For the optimization procedure, we will run Adam for 50000 epochs with learning rate 0.001 and then run L-BFGS until convergence. For the network initialization, we will use the Glorot normal initializer.

The convergence history, reference solution, PINN solution, and the absolute error is shown in the Fig. 2, from which we can see a perfect match is achieved, and the locations with highest error is close to the origin due to sharp gradient. Note in Fig. 2, we have enforced all the output values are between 0 and 1, and initial condition is strictly satisfied.

4.2. Steady state heat conduction problem

Our second example is a time-independent problem on 2D. In the steady state heat transfer problem with constant heat conduction coefficient, constant heat source and constant boundary temperature (Dirichlet boundary condition), the T will satisfy Poisson's equation [14]. Without loss of generality and to avoid the scaling differences, we will solve following PDE on $\Omega = [-1,1] \times [-1,1]$

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + 1 = 0 \qquad -1 < x < 1 \qquad -1 < y < 1\,, \tag{14}$$

$$T = 0 \qquad (x,y) \in \partial\Omega\,. \tag{15}$$

Even though the analytical solution for this problem doesn't exist, the finite element solution with high accuracy is available in deal.ii library, thus it will be interesting to compare these two numerical solutions.

For the PINN hyper-parameters, we choose 1500 and 500 random points to form Γ_f and Γ_b, respectively. The test set Γ_{test} will have 10000 equispaced points. The loss weights are assigned as $w_f = w_b = 1/2$ (w_d is not used in this problem). For the architecture of the neural network, it will contain 4 hidden layers with 50 neurons each, and we will not apply any transform for the network output. The learning rate for this problem is 0.0005. For the network initialization, we will use the Glorot uniform initializer. The remaining settings are the same as the first example.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 September 2020 doi:10.20944/preprints202006.0258.v2

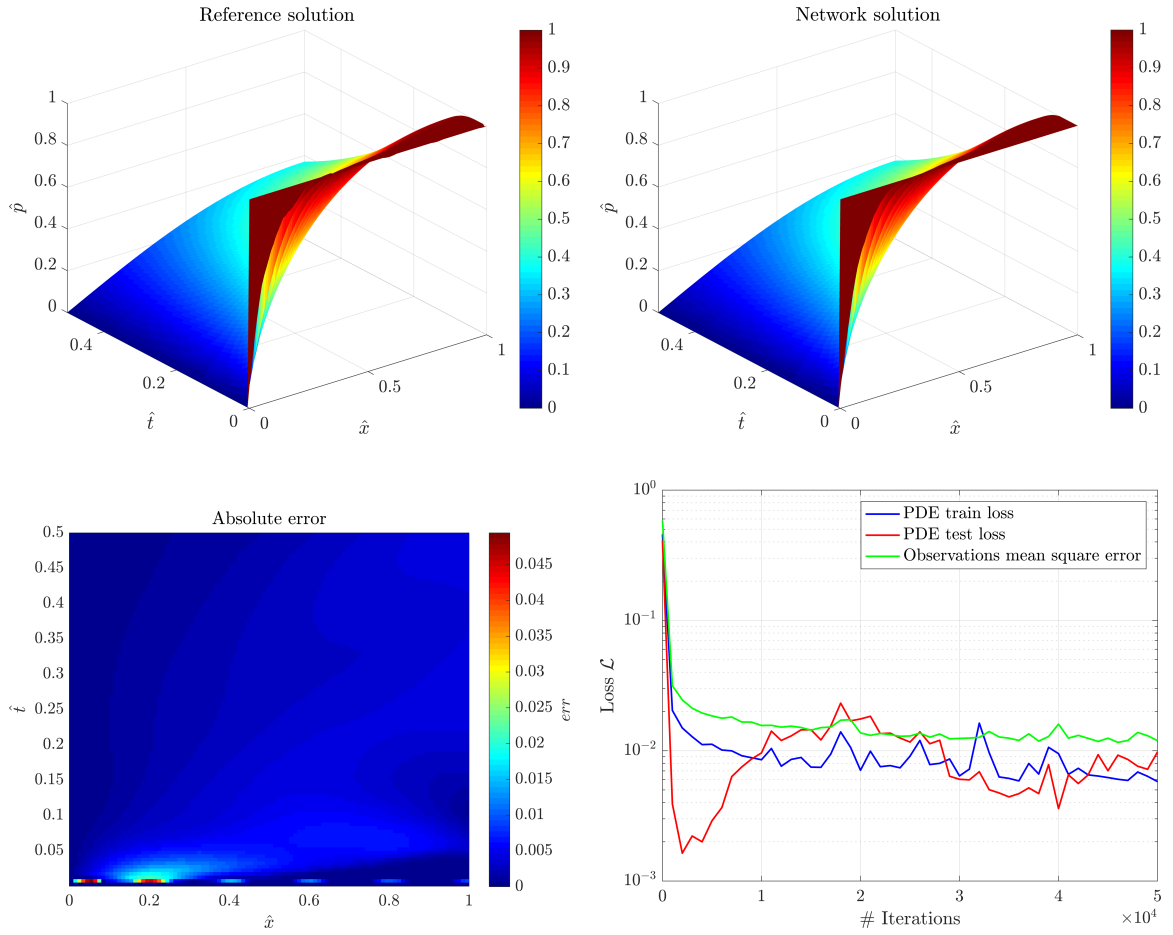

Figure 2: PINN training results for the one dimensional consolidation problem. For the fourth figure, we need to mention the PDE test loss on Γ_{test} is only due to mismatch with the governing equation, *i.e.*, $\mathcal{L}_f$. The green line is also evaluated from the same test set Γ_{test}. For the PDE train loss, we recall `pde, [bc1,bc2,ic]` from the source code, thus we take an average to get train loss which is consistent with a weight of $1/4$.

The training results are shown in the Fig. 3, where the two numerical solutions are consistent with each other. Note in Fig. 3, we have enforced all the output values are larger than 0. The largest absolute error happens near the Dirichlet boundary because Eq. (15) is not regarded as a hard constrain in the construction of the loss $\mathcal{L}$. A feasible way to handle this problem would be to introduce a smooth distance function $\hat{d}(\boldsymbol{x})$ that gives the distance for $\boldsymbol{x} \in \Omega$ to $\partial\Omega$ [3], and then we multiply $\hat{d}$ with the original network output to get our new improved output, which will automatically satisfy Eq. (15).

4.3. Inverse problem for simple structural dynamic system

The last example comes from the dynamic response of the single degree of freedom system [10] and we want to use this example to illustrate the capability of PINN in the inverse modeling or data discovery process. In structural dynamics, the free vibration equation with viscous damping for single mass point is obtained through D'Alembert principle [10]

$$m\frac{\mathrm{d}^2u}{\mathrm{d}t^2} + c\frac{\mathrm{d}u}{\mathrm{d}t} + ku = 0 \qquad t > 0 \tag{16}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 September 2020 doi:10.20944/preprints202006.0258.v2

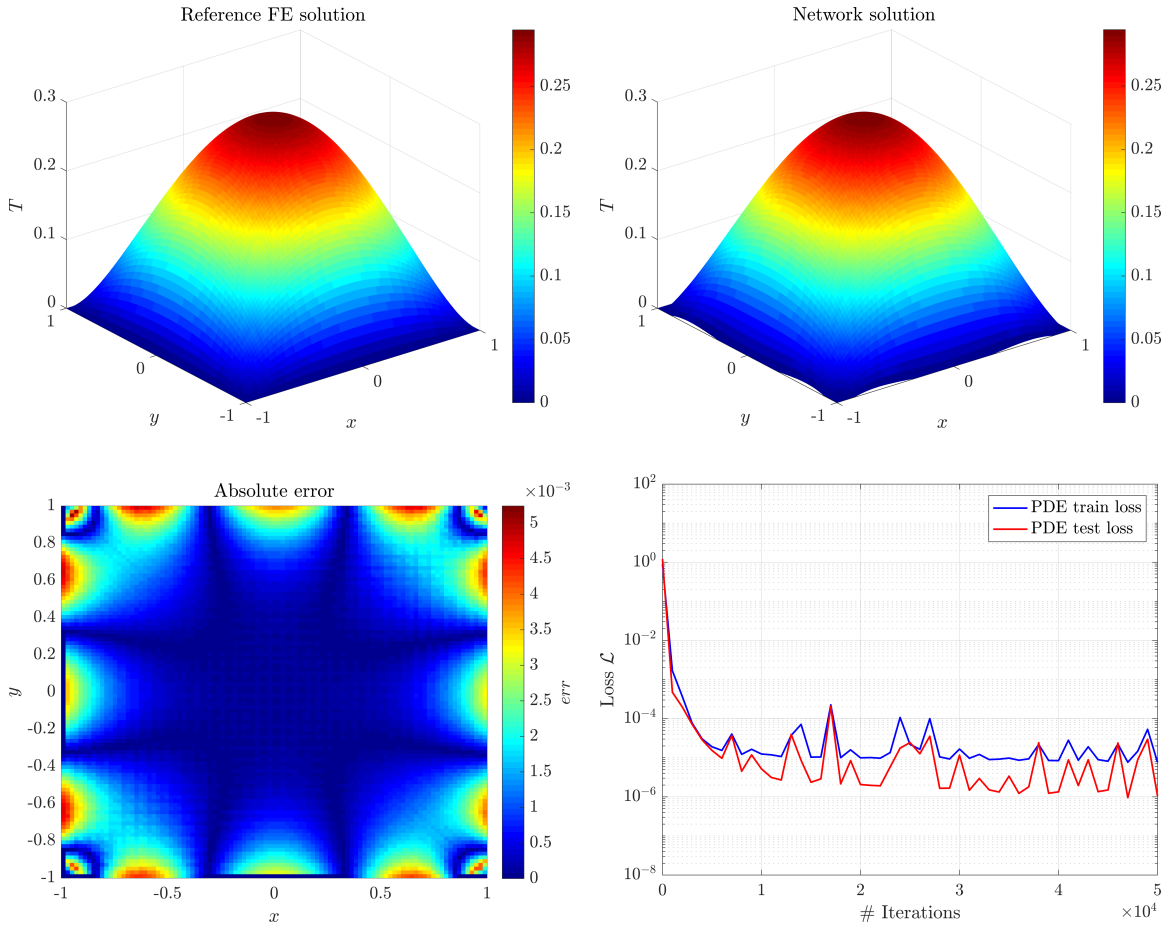

Figure 3: PINN training results for the steady state heat conduction problem. For the fourth figure, we need to mention the PDE test loss on Γ_{test} is only due to mismatch with the governing equation, *i.e.*, $\mathcal{L}_f$. For the PDE train loss, we recall `pde, bc` from the source code, thus we take an average to get train loss which is consistent with a weight of $1/2$.

with $u(0) = u_0$ and $\dot{u}(0) = v_0$. In above equation, $m > 0$ is the mass, $c > 0$ is the damping coefficient, $k > 0$ is the stiffness, u is the displacement, u_0 is the initial displacement and v_0 is the initial velocity. The analytical solution to this ODE is given as

$$u(t) = \mathrm{e}^{-\xi\omega_n t}\left[u_0 \cos(\omega_D t) + \frac{v_0 + \xi\omega_n u_0}{\omega_D}\sin(\omega_D t)\right], \tag{17}$$

where $\omega_n = \sqrt{k/m}$ is known as the natural frequency, $\xi = c/(2m\omega_n)$ is known as the damping ratio, $\omega_D = \omega_n\sqrt{1-\xi^2}$. In this problem, we will assume $\xi < 1$ which is known as the underdamped system.

Now suppose $m = u_0 = v_0 = 1$, c and k are the two parameters to be identified from the observations at certain times. The observations are produced with the underlying true parameters $c = 0.4$ and $k = 4$ using Eq. (17). We choose 50 equispaced points to form Γ_d, 100 random points to form Γ_f, 50 random points to form Γ_b and 10000 equispaced points to form Γ_{test}. The loss weights are assigned as $w_f = w_b = w_d = 1/4$. For the architecture of the neural network, it will contain 3 hidden layers with 32 neurons each, and we will not apply any transform for the network output. For the optimization procedure, we will only run Adam

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 September 2020 doi:10.20944/preprints202006.0258.v2

for 100000 epochs with learning rate 0.0005. For the network initialization, we will use the Glorot uniform initializer.

The evolution trajectories of c and k are presented in Fig. 4, with final values of $\bar{k} = 3.9933197$ and $\bar{c} = 0.40125215$, which agree with their true values.

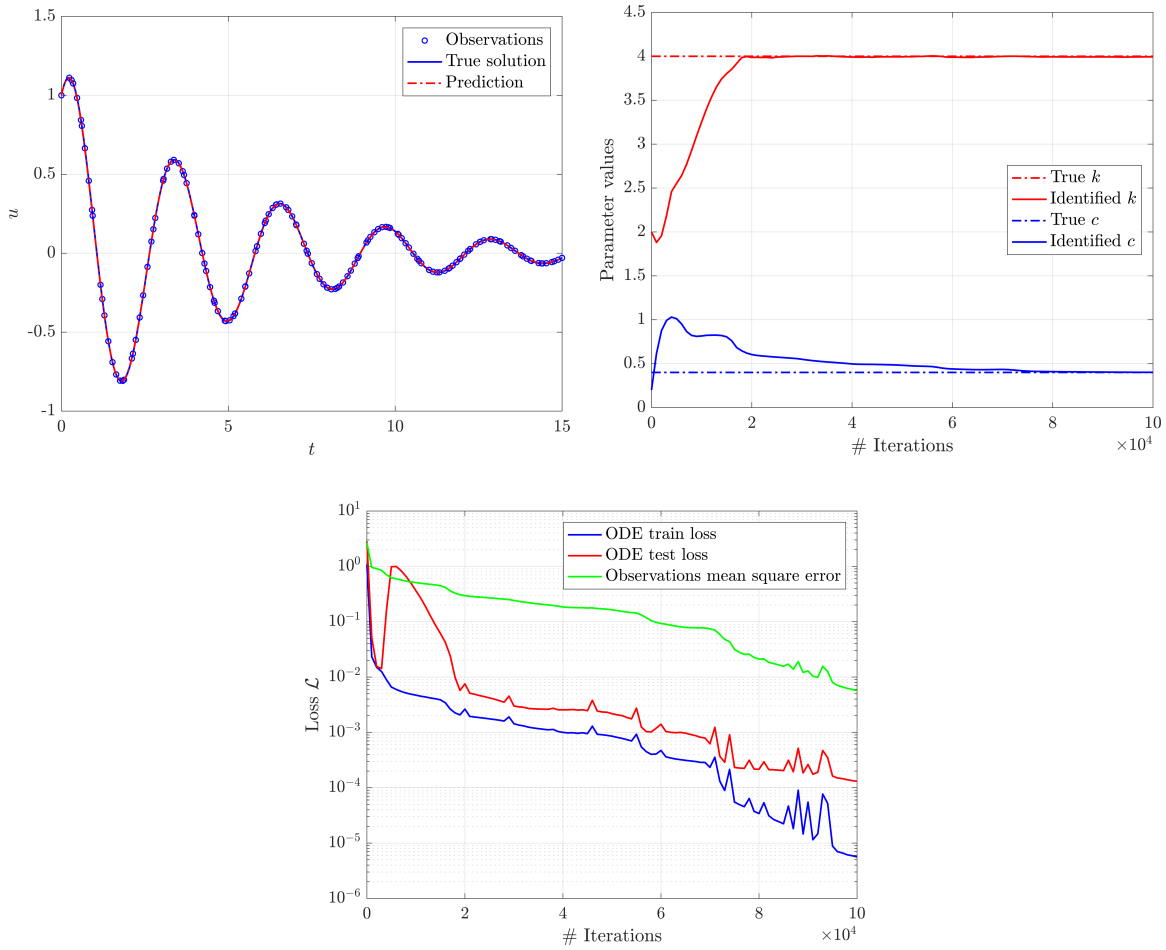

Figure 4: PINN training results for the parameter identification problem, the identified values converge to the true values during the training process. For the first figure, the observations contain 150 points (Γ_f and Γ_b), the other two overlapping lines come from the Γ_{test}. For the third figure, the red line and green line come from Γ_{test}, and the blue line now contains Γ_d, recall `pde, [bc1,bc2,observe_y]` from the source code, thus we take an average to get train loss which is consistent with a weight of $1/4$. ODE test loss on Γ_{test} is only due to mismatch with the governing equation, *i.e.*, $\mathcal{L}_f$.

5. Conclusions

In this project, we have adopted PINN, a new class of universal function approximators that is capable of encoding any underlying conservation laws which can be described by differential equations. Compared to the traditional numerical methods, PINN employs automatic differentiation to handle differential operators, and thus are mesh-free. We use PINN as a data-driven algorithm for inferring solutions and parameter identifications of differential equations. A series of promising results for a diverse collection of problems in

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 September 2020 doi:10.20944/preprints202006.0258.v2

mechanics are presented and discussed, which opens the path for endowing deep learning with the powerful capacity of mathematical physics to model the world around us. We hope this project could benefit future practitioners across a wide range of scientific domains who want to incorporate deep learning methods in modeling physics-related problems.

For the future work, it is worth to try a group of coupled differential equations such as Eqs. (18)(19) in poromechanics [40] to see the PINN's performance, and develop some training points selection strategy similar to the adaptive re-meshing in the mesh-based methods to achieve better accuracy.

$$\frac{1}{M}\frac{\partial p}{\partial t} - \frac{k}{\mu_f}\nabla^2 p + \alpha\frac{\partial \nabla\cdot\boldsymbol{u}}{\partial t} = 0\,, \tag{18}$$

$$G\nabla^2\boldsymbol{u} + \frac{G}{1-2\nu}\nabla\left(\nabla\cdot\boldsymbol{u}\right) - \alpha\nabla p + \rho_b\boldsymbol{g} = \boldsymbol{0}\,. \tag{19}$$

Acknowledgement

The authors deeply appreciate the deep learning library `https://github.com/lululxvi/deepxde` which facilitates the code implementation through its built-in modules. The suggestions from Dr. Lu are also well-acknowledged.

References

[1] Y. Bar-Sinai, S. Hoyer, J. Hickey, and M. P. Brenner, *Learning data-driven discretizations for partial differential equations*, Proceedings of the National Academy of Sciences, 116 (2019).

[2] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, *Automatic differentiation in machine learning: a survey*, The Journal of Machine Learning Research, 18 (2017).

[3] J. Berg and K. Nyström, *A unified deep artificial neural network approach to partial differential equations in complex geometries*, Neurocomputing, 317 (2018).

[4] S. Boyd and L. Vandenberghe, *Convex optimization*, Cambridge university press, 2004.

[5] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, *A limited memory algorithm for bound constrained optimization*, SIAM Journal on scientific computing, 16 (1995), pp. 1190–1208.

[6] N. Castelletto, J. A. White, and H. A. Tchelepi, *Accuracy and convergence properties of the fixed-stress iterative solution of two-way coupled poromechanics*, International Journal for Numerical and Analytical Methods in Geomechanics, 39 (2015).

[7] X. Chen, J. Duan, and G. E. Karniadakis, *Learning and meta-learning of stochastic advection-diffusion-reaction systems from sparse measurements*, arXiv preprint arXiv:1910.09098, (2019).

[8] A. H. D. Cheng, *Poroelasticity*, Springer, 2016.

[9] M. Chiaramonte and M. Kiener, *Solving differential equations using neural networks*, CS229 Project, (2013).

[10] A. K. Chopra, *Dynamics of structures: theory and applications to earthquake engineering*, Prentice Hall, 1995.

[11] E. Detournay and A. H. D. Cheng, *Fundamentals of poroelasticity*, in Analysis and design methods, Elsevier, 1993, pp. 113–171.

[12] M. W. M. G. Dissanayake and N. Phan-Thien, *Neural-network-based approximations for solving partial differential equations*, Communications in Numerical Methods in Engineering, 10 (1994), pp. 195–201.

[13] T. Dockhorn, *A discussion on solving partial differential equations using neural networks*, arXiv preprint arXiv:1904.07200, (2019).

[14] L. C. Evans, *Partial differential equations*, American Mathematical Society, 2010.

[15] J. Ghaboussi, J. H. Garrett Jr, and X. Wu, *Knowledge-based modeling of material behavior with neural networks*, Journal of engineering mechanics, 117 (1991).

[16] E. Haghighat, M. Raissi, A. Moure, H. Gomez, and R. Juanes, *A deep learning framework for solution and discovery in solid mechanics: linear elasticity*, arXiv preprint arXiv:2003.02751, (2020).

[17] J. Han, A. Jentzen, and E. Weinan, *Solving high-dimensional partial differential equations using deep learning*, Proceedings of the National Academy of Sciences, 115 (2018).

[18] Q. He, D. Barajas-Solano, G. Tartakovsky, and A. M. Tartakovsky, *Physics-informed neural networks for multi-physics data assimilation with application to subsurface transport*, Advances in Water Resources, (2020), p. 103610.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 September 2020 doi:10.20944/preprints202006.0258.v2

[19] T. Kadeethum, T. M. Jørgensen, and H. M. Nick, *Physics-informed neural networks for solving nonlinear diffusivity and biot's equations*, PloS one, 15 (2020), p. e0232683.
[20] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, arXiv preprint arXiv:1412.6980, (2014).
[21] G. Kissas, Y. Yang, E. Hwuang, W. R. Witschey, J. A. Detre, and P. Perdikaris, *Machine learning in cardiovascular flows modeling: Predicting arterial blood pressure from non-invasive 4d flow mri data using physics-informed neural networks*, Computer Methods in Applied Mechanics and Engineering, 358 (2020).
[22] I. E. Lagaris, A. Likas, and D. I. Fotiadis, *Artificial neural networks for solving ordinary and partial differential equations*, IEEE transactions on neural networks, 9 (1998).
[23] M. Lefik, D. P. Boso, and B. A. Schrefler, *Artificial neural networks in numerical modelling of composites*, Computer Methods in Applied Mechanics and Engineering, 198 (2009).
[24] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis, *DeepXDE: A deep learning library for solving differential equations*, arXiv preprint arXiv:1907.04502, (2019).
[25] X. Meng, Z. Li, D. Zhang, and G. E. Karniadakis, *Ppinn: Parareal physics-informed neural network for time-dependent pdes*, arXiv preprint arXiv:1909.10145, (2019).
[26] M. Mozaffar, R. Bostanabad, W. Chen, K. Ehmann, J. Cao, and M. A. Bessa, *Deep learning predicts path-dependent plasticity*, Proceedings of the National Academy of Sciences, 116 (2019).
[27] M. Raissi, *Deep hidden physics models: Deep learning of nonlinear partial differential equations*, The Journal of Machine Learning Research, 19 (2018), pp. 1–24.
[28] M. Raissi and G. E. Karniadakis, *Hidden physics models: Machine learning of nonlinear partial differential equations*, Journal of Computational Physics, 357 (2018), pp. 125–141.
[29] M. Raissi, P. Perdikaris, and G. E. Karniadakis, *Machine learning of linear differential equations using gaussian processes*, Journal of Computational Physics, 348 (2017), pp. 683–693.
[30] M. Raissi, P. Perdikaris, and G. E. Karniadakis, *Physics informed deep learning (part i): data-driven solutions of nonlinear partial differential equations*, arXiv preprint arXiv:1711.10561, (2017).
[31] M. Raissi, P. Perdikaris, and G. E. Karniadakis, *Physics informed deep learning (part ii): data-driven discovery of nonlinear partial differential equations*, arXiv preprint arXiv:1711.10566, (2017).
[32] M. Raissi, P. Perdikaris, and G. E. Karniadakis, *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations*, Journal of Computational Physics, 378 (2019), pp. 686–707.
[33] J. Sirignano and K. Spiliopoulos, *Dgm: A deep learning algorithm for solving partial differential equations*, Journal of Computational Physics, 375 (2018), pp. 1339–1364.
[34] Y. Sun, W. Zeng, Y. Zhao, X. Zhang, Y. Shu, and Y. Zhou, *Modeling constitutive relationship of ti40 alloy using artificial neural network*, Materials & Design, 32 (2011).
[35] A. Tartakovsky, C. O. Marrero, P. Perdikaris, G. Tartakovsky, and D. Barajas-Solano, *Physics-informed deep neural networks for learning parameters and constitutive relationships in subsurface flow problems*, Water Resources Research, 56 (2020).
[36] J. Tompson, K. Schlachter, P. Sprechmann, and K. Perlin, *Accelerating eulerian fluid simulation with convolutional networks*, in Proceedings of the 34th International Conference on Machine Learning, 2017.
[37] H. F. Wang, *Theory of linear poroelasticity with applications to geomechanics and hydrogeology*, Princeton University Press, 2000.
[38] E. Weinan, J. Han, and A. Jentzen, *Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations*, Communications in Mathematics and Statistics, 5 (2017), pp. 349–380.
[39] E. Weinan and B. Yu, *The deep ritz method: a deep learning-based numerical algorithm for solving variational problems*, Communications in Mathematics and Statistics, 6 (2018), pp. 1–12.
[40] Q. Zhang, *Hydromechanical modeling of solid deformation and fluid flow in the transversely isotropic fissured rocks*, Computers and Geotechnics, 128 (2020).