Noname manuscript No.
(will be inserted by the editor)

Physics informed deep learning for coupled flow and poromechanics. Mandel's problem

Saumik Dana · Teeratorn Kadeethum

Received: date / Accepted: date

Abstract We evaluate the performance of the physics informed deep learning paradigm for solving the Biot system modeling coupled flow and poromechanics using Mandel's problem analytical solution. The solution presents a unique set of challenges to the deep learning paradigm, such as the disparity in expected orders of magnitude of the output variables as well as the non-monotonicity and steep gradient in one of the output variables in a certain spatio-temporal domain. We tackle those challenges in this work and comment on the effect of activation function and minimization algorithm on the deep learning framework.

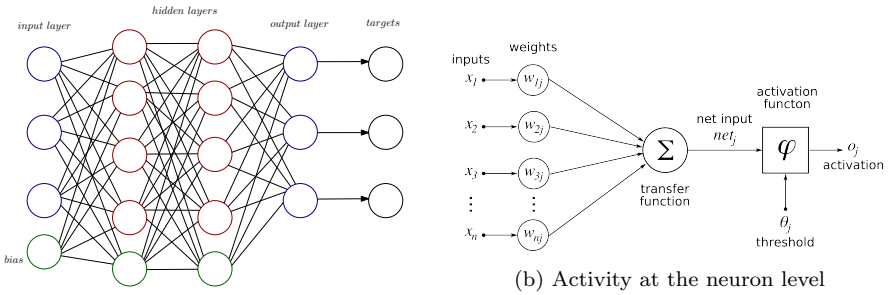
Keywords Physics informed deep learning · Biot system · Mandel's problem · TensorFlow · SciPy · Activation function · Batch gradient descent

1 Introduction

The world of business has been leveraging machine learning in several diverse application areas like analyzing sales data towards streamlining the data, real-time mobile personalization to promote user experience, fraud detection with the help of pattern detection, product recommendations based on customer feedback, dynamic pricing based on user demand and natural language processing to better the ability of machines to interact with humans. The advent of open-source packages like TensorFlow [1], PyTorch [2], Theano [3] along with the availability of Python packages like SciPy [4], numpy [5], scikit-learn [6] has spawned interest in exploring the deep learning paradigm in engineering sciences as well [7–27]. As shown in Figure 1a, a neural network is composed of several connected layers of artificial neurons and biases where the data is fed into the input layer and flows through some hidden layers. As shown in Figure 1b, the output of each neuron is computed by multiplying the outputs from the previous layer with the corresponding weights

S. Dana
University of Southern California, Los Angeles, CA 90007
E-mail: sdana@usc.edu

T. Kadeethum
Cornell University, Ithaca, NY 14850



(a) Basic neural net construction

(b) Activity at the neuron level

Fig. 1: Aspects of neural net architecture

followed by an alteration by an activation function [28,29]. In the training phase, the weights of the neural network are initialized [30,31], followed by continual updates on the weights using an algorithm such that the objective function is minimized. The objective function is the cumulative error due to training of the data set on the spatio-temporal boundary of the domain and the physics informed error corresponding to the residue of the partial differential equations. To formalize the framework mathematically, let $\mathcal{N}(\mathcal{X};\mathbf{W},\mathbf{b})$ be a L-layer neural network with input vector $\mathcal{X}$, output vector $\mathbf{y}$ and network parameters $\mathbf{W}$ and $\mathbf{b}$ called weights and biases respectively. Each layers creates data for the next layer through the nested transformations: $\mathbf{o}^l = \varphi^l(\mathbf{W}^l\mathbf{o}^{l-1} + \mathbf{b}^l)$, $l = 1, \cdots, L$ where $\mathbf{o}^0 \equiv \mathbf{x}$ and $\mathbf{o}^L \equiv \mathbf{y}$ and φ^l are activation functions and the mapping is represented as $\mathbf{y} \mapsto \mathcal{N}(\mathcal{X};\mathbf{W},\mathbf{b})$.

In this work, we test the neural net construction framework's metrics in the coupled flow and poromechanics space for the prevalent Mandel's problem [32,33] which presents its unique set of challenges to the deep learning paradigm. This paper is structured as follows: the Biot system is presented in this section, the physics informed machine learning paradigm for Mandel's problem is presented in section 2, results are presented in section 3, and conclusions and outlook are presented in section 4.

1.1 Coupled flow and poromechanics

The coupled flow and poromechanics phenomena is typically resolved numerically using a staggered solution algorithm [34–40] in which the system of equations is solved sequentially and iteratively. Let Ω be the domain under consideration. The Biot system consists of the flow model and the poromechanics model. Let the boundary $\partial\Omega = \Gamma_D^f \cup \Gamma_N^f \equiv \Gamma_D^p \cup \Gamma_N^p$ where Γ_D^f, Γ_N^f and Γ_D^f, Γ_N^f are the Dirichlet and Neumann boundaries for the flow and poromechanics models respectively. The

system of equations is

$$\begin{aligned}
\text{Mass conservation} &: \tfrac{\partial}{\partial t}(\tfrac{1}{M}p + \alpha\nabla\cdot\mathbf{u}) + \nabla\cdot\mathbf{z} = q \\
\text{Darcy law} &: \mathbf{z} = -\tfrac{\mathbf{K}}{\mu}(\nabla p - \rho_0\mathbf{g}) = -\boldsymbol{\kappa}(\nabla p - \rho_0\mathbf{g}) \\
\text{Linear momentum balance} &: \nabla\cdot\boldsymbol{\sigma} + \mathbf{f} = \mathbf{0} \\
\text{Constitutive law} &: \boldsymbol{\sigma} = \lambda(\nabla\cdot\mathbf{u})\mathbf{I} + 2G\boldsymbol{\epsilon} - \alpha p\mathbf{I} \\
\text{Small strain kinematics} &: \boldsymbol{\epsilon}(\mathbf{u}) = \tfrac{1}{2}(\nabla\mathbf{u} + (\nabla\mathbf{u})^T) \\
\text{Boundary conditions} &: p = g \text{ on } \Gamma_D^f, \mathbf{z}\cdot\mathbf{n} = q_D \text{ on } \Gamma_N^f, \\
&\quad \mathbf{u}\cdot\mathbf{n}_1 = u_D \text{ on } \Gamma_D^p, \boldsymbol{\sigma}^T\mathbf{n}_2 = \mathbf{t} \text{ on } \Gamma_N^p \\
\text{Initial conditions} &: p(\mathbf{x},0) = p_0(\mathbf{x}),\ \rho(\mathbf{x},0) = \rho_0(\mathbf{x}), \\
&\quad \phi(\mathbf{x},0) = \phi_0(\mathbf{x}), \mathbf{u}(\mathbf{x},0) = \mathbf{u}_0(\mathbf{x})
\end{aligned}$$

where $p : \Omega \times (0,T] \to \mathbb{R}$ is the fluid pressure, $\mathbf{z} : \Omega \times (0,T] \to \mathbb{R}^3$ is the fluid flux, M is the Biot modulus, q is the source or sink term, $\mathbf{K}$ is the uniformly symmetric positive definite absolute permeability tensor, μ is the fluid viscosity, ρ_0 is a reference density, ϕ is the porosity, $\boldsymbol{\kappa} = \frac{\mathbf{K}}{\mu}$ is a measure of the hydraulic conductivity of the pore fluid, c is the fluid compressibility, $T > 0$ is the time interval, $\mathbf{u} : \Omega \times [0,T] \to \mathbb{R}^3$ is the solid displacement, $\mathbf{f}$ is the body force per unit volume, $\boldsymbol{\epsilon}$ is the strain tensor, $\boldsymbol{\sigma}$ is the Cauchy stress tensor, G is shear modulus and λ is lame parameter.

1.2 Mandel's problem and its importance in deep learning

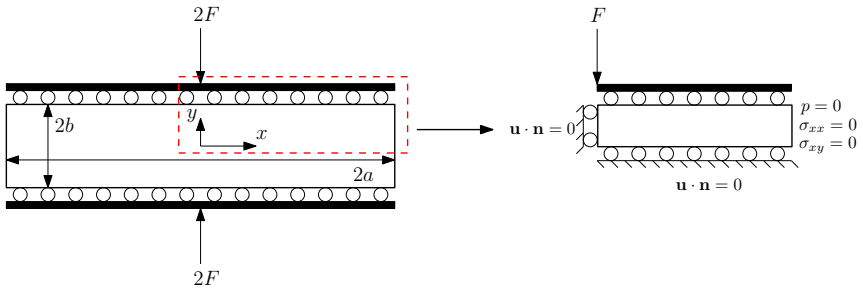
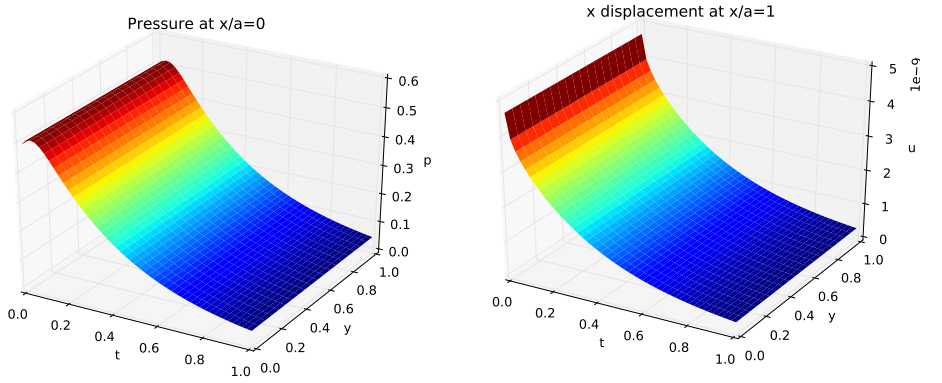


Fig. 2: Circles indicate rollers and solid black boxes indicate rigid frictionless plates

As shown in Figure 2, a rectangular isotropic specimen is sandwiched between rigid, frictionless plates and the lateral sides are traction-free. At $t = 0^+$, a force intensity of $2F\ N/m$ is applied to the rigid plates. Plane strain condition is applicable i.e. $\epsilon_{zz} = 0$. The boundary conditions are

$$\int_{-a}^{a} \sigma_{yy}|_{y=\pm b}dx = -2F, p|_{x=\pm a} = 0, v|_{y=\pm b} = 0, \sigma_{xx}|_{x=\pm a} = \sigma_{xy}|_{x=\pm a} = 0 \qquad \forall t > 0$$

Given the biaxial symmetry of the problem, only a quarter of the domain needs to be modeled. With U representing the analytical solution for the y displacement



(a) Non-monotonic pressure response at center of specimen

(b) The x-displacement at the lateral free edge is roughly 8 orders of magnitude less than the pressure

Fig. 3: Mandel's problem

at $y = b$, the boundary conditions are recast as

$$u|_{x=0} = v|_{y=0} = p|_{x=a} = \sigma_{xx}|_{x=a} = \sigma_{xy}|_{x=a} = 0, v|_{y=b} = U \qquad \forall t > 0$$

We use the following material parameters: $a = 1\,m$, $b = 1\,m$, $T = 1\,s$, $\lambda = 8.48 \times 10^7\,Pa$, $\alpha = 1$, $k = 100\,D$, $\mu = 10\,cp$, $F = 1\,N/m$ to generate the analytical solution which is plotted in Figure 3. The important aspects of the solution that present a challenge to the machine learning paradigm are the disparity in expected order of magnitude in the target variables and the non-monotonicity in one of the target variables in a certain spatio-temporal subdomain.

2 Physics informed deep learning

The deep learning model used for training is a fully-connected neural network where the number of hidden layers N_h and the number of neurons N_n per hidden layer is adjusted depending on the problem under consideration [41], [42]. With $\mathcal{X} \equiv (x, y, t)$ representing the spatio-temporal input vector, we deploy a separate neural net for each primary variable as follows

$$\hat{\phi} \mapsto \mathcal{N}_\phi(\mathcal{X}, \mathbf{W}, \mathbf{b}), \;\; \phi = u, v, p, \sigma_{xx}, \sigma_{xy}, \sigma_{yy}$$

We rewrite the governing differential equations for the Mandel's problem as

$$\begin{gathered}
\frac{\partial}{\partial t}\left(\frac{1}{M}p + \alpha\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)\right) - \kappa\nabla^2 p = 0 \\
\sigma_{xx} = \lambda\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) + 2G\frac{\partial u}{\partial x} - \alpha p \\
\sigma_{xy} = G\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \\
\sigma_{yy} = \lambda\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) + 2G\frac{\partial v}{\partial y} - \alpha p \\
\frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{xy}}{\partial y} = 0 \\
\frac{\partial \sigma_{yx}}{\partial y} + \frac{\partial \sigma_{yy}}{\partial y} = 0
\end{gathered}$$

The corresponding residuals are

$$\left|\begin{array}{c} \mathit{\Pi}_p(\mathcal{X}) := \frac{\partial}{\partial t}\left(\frac{1}{M}\hat{p} + \alpha\left(\frac{\partial \hat{u}}{\partial x} + \frac{\partial \hat{v}}{\partial y}\right)\right) - \kappa\left(\frac{\partial^2 \hat{p}}{\partial x^2} + \frac{\partial^2 \hat{p}}{\partial y^2}\right) \\ \mathit{\Pi}_{\sigma_{xx}}(\mathcal{X}) := \hat{\sigma}_{xx} - \left(\lambda\left(\frac{\partial \hat{u}}{\partial x} + \frac{\partial \hat{v}}{\partial y}\right) + 2G\frac{\partial \hat{u}}{\partial x} - \alpha\hat{p}\right) \\ \mathit{\Pi}_{\sigma_{xy}}(\mathcal{X}) := \hat{\sigma}_{xy} - G\left(\frac{\partial \hat{u}}{\partial y} + \frac{\partial \hat{v}}{\partial x}\right) \\ \mathit{\Pi}_{\sigma_{yy}}(\mathcal{X}) := \hat{\sigma}_{yy} - \left(\lambda\left(\frac{\partial \hat{u}}{\partial x} + \frac{\partial \hat{v}}{\partial y}\right) + 2G\frac{\partial \hat{v}}{\partial y} - \alpha\hat{p}\right) \\ \mathit{\Pi}_u(\mathcal{X}) := \lambda\left(\frac{\partial^2 \hat{u}}{\partial x^2} + \frac{\partial^2 \hat{v}}{\partial x \partial y}\right) + 2G\frac{\partial^2 \hat{u}}{\partial x^2} - \alpha\frac{\partial \hat{p}}{\partial x} + G\left(\frac{\partial^2 \hat{u}}{\partial y^2} + \frac{\partial^2 \hat{v}}{\partial x \partial y}\right) \\ \mathit{\Pi}_v(\mathcal{X}) := \lambda\left(\frac{\partial^2 \hat{u}}{\partial x \partial y} + \frac{\partial^2 \hat{v}}{\partial y^2}\right) + 2G\frac{\partial^2 \hat{v}}{\partial y^2} - \alpha\frac{\partial \hat{p}}{\partial y} + G\left(\frac{\partial^2 \hat{v}}{\partial x^2} + \frac{\partial^2 \hat{u}}{\partial x \partial y}\right) \end{array}\right.$$

2.1 Loss function for Mandel's problem

Let N_0 be number of initial points sampled, N_b be number of boundary points sampled and N_f be the number of collocation points sampled in the interior of the domain using the Latin hypercube method [43]. With the notation $||(\cdot)|| := |(\cdot)|^2$, the loss function $\mathcal{L}$ to be minimised is

$$\left|\begin{array}{c} \mathcal{L} = \mathcal{L}_b + \sum_{\phi \in \{u,v,p,\sigma_{xx},\sigma_{xy},\sigma_{yy}\}} (\mathcal{L}_{0_\phi} + \mathcal{L}_\phi) \\ \mathcal{L}_{0_\phi} = \frac{1}{N_0}\sum_{i=1}^{N_0} ||\hat{\phi}(\mathcal{X}|^i_{t=0}) - \phi(\mathcal{X}|^i_{t=0})|| \\ \mathcal{L}_b = \frac{1}{N_b}\sum_{i=1}^{N_b} \big(||\hat{u}(\mathcal{X}|^i_{x=0})|| + ||\hat{v}(\mathcal{X}|^i_{y=0})|| + ||\hat{p}(\mathcal{X}|^i_{x=a})|| + ||\hat{v}(\mathcal{X}|^i_{y=b}) - U(\mathcal{X}|^i_{y=b})|| \\ + ||\hat{\sigma}_{xx}(\mathcal{X}|^i_{x=a})|| + ||\hat{\sigma}_{xy}(\mathcal{X}|^i_{x=a})||\big) \\ \mathcal{L}_\phi = \frac{1}{N_f}\sum_{i=1}^{N_f} (||\mathit{\Pi}_\phi(\mathcal{X}^i)|| + ||\hat{\phi}(\mathcal{X}^i) - \phi(\mathcal{X}^i)||) \end{array}\right.$$

where $\mathcal{L}_0$ and $\mathcal{L}_b$ correspond to loss due to training on initial and boundary conditions respectively, whereas $\mathcal{L}_p, \mathcal{L}_u, \mathcal{L}_v, \mathcal{L}_{\sigma_{xx}}, \mathcal{L}_{\sigma_{xy}}$ and $\mathcal{L}_{\sigma_{yy}}$ correspond to loss due to training on the variables $p, u, v, \sigma_{xx}, \sigma_{xy}$ and σ_{yy} respectively.

2.2 Training process

We test three different activation functions i.e. sigmoid, ReLU [44], and hyperbolic tangent. Weights are initialized in accordance with Xavier scheme [30]. The residuals involve computation of derivatives which are obtained using automatic differentiation [45]. The loss function is optimized with two competing schemes: Limited memory Broyden-Fletcher-Goldfarb-Shanno, L-BFGS [46] and Batch gradient descent, BGD, [47]. There are many BGD algorithmic variants and we choose the Adam [48] variant due to its popularity in the machine learning community [49].

3 Numerical results

We use TensorFlow [1] and run simulations on a dual core AMD Ryzen 3 3200U processor with Radeon Vega 3 Graphics card. We generate the exact solution points on a rectangular mesh ($[0,1] \times [0,1]$) with 20 equidistant intervals in both

x and y direction. Using 49 equidistant intervals in the temporal domain $[0, 1]$, in total, we have $N_{total} = 20 \times 20 \times 50 = 20000$ examples. We use $N_0 = 200$ for initial conditions, $N_b = 500$ data points for boundary conditions on each of the four boundaries, $N_f = 15000$ data points in the interior of the domain, $N_h = 4$ and $N_n = 10$. The stopping criterion for both schemes is $\frac{\|\mathcal{L}^{k+1} - \mathcal{L}^k\|}{\|\mathcal{L}^{k+1}\|} < 1 \times 10^{-6}$

3.1 Capturing the non-monotonic pressure response

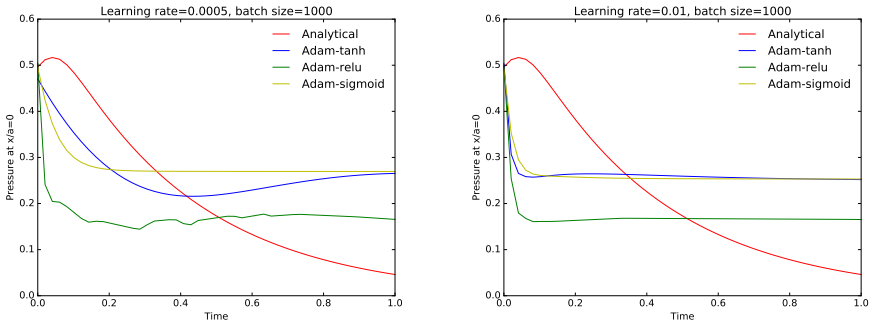


Fig. 4: Comparison of pressure response using different learning rates for Adam Batch Gradient Method

As shown in Figure 4, we attempt to capture the non-monotonic pressure response using the different activation functions for the Adam BGD with a batch size of 1000 and learning rate of 0.0005. While none of the combinations does a perfect job, we observe that a learning rate of 0.0005 is more optimal from the standpoint of capturing this critical physical phenomena

3.2 Relative error in pressure

As shown in Figure 5, we track the relative error in pressure as the training epochs proceed. We observe that the ReLU activation function with a batch size of 1000 and a learning rate of 0.0005 is optimal from the standpoint of the error metrics. This is consistent with our observation from the standpoint of capturing the non-monotonicity in the previous module.

3.3 Decay in loss function

We plot the decay in loss function versus training epochs in Figure 6. We again observe that the batch size of 1000 with a learning rate of 0.0005 is optimal from the standpoint of avoiding the fluctuations in the loss function as the training epochs proceed.

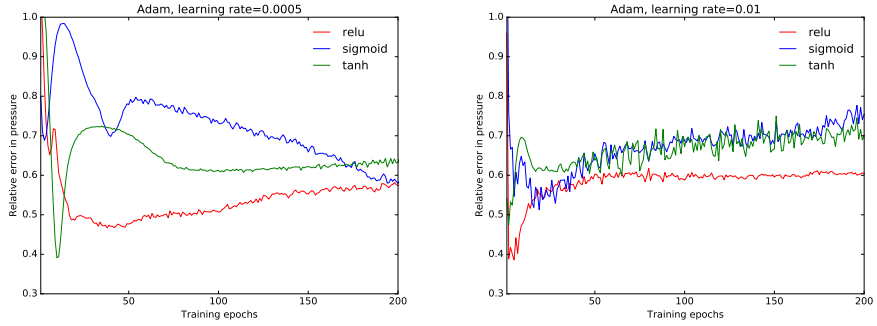


Fig. 5: Relative error in pressure is $\frac{\|p-\hat{p}\|}{\|p\|}$

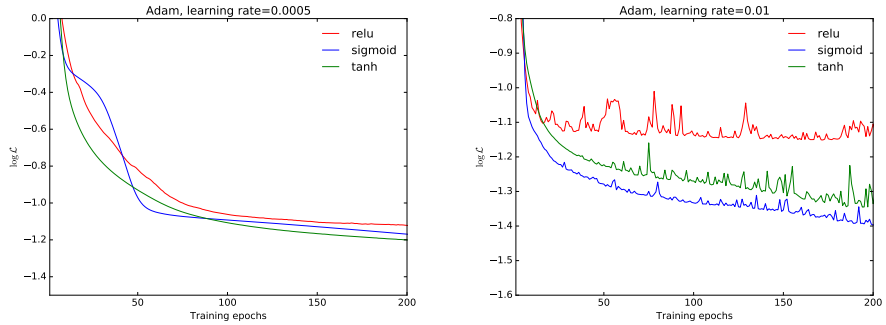


Fig. 6: Loss function decay

3.4 Introducing the truncated sine and cosine activation functions

We now introduce the truncated sine and cosine activation functions, and proceed to observe the effects of this choice on the overall behavior of the deep learning framework. The argument for choosing these functions is that the non-monotonicity would be better captured is the activation function itself is non-monotonic in behavior. The truncated sine function is such that

$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ sin(x) & \text{for } x > 0 \end{cases}$$

and the truncated cosine function is such that

$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ cos(x) & \text{for } x > 0 \end{cases}$$

The functions are plotted in Figure 7. We compare the output of the deep learning with the truncated sine and cosine activation functions as shown in Figure 8. It is evident that the truncated sine activation function with BFGS optimizer holds the most promise in terms of being able to capture the non-monotonic behavior.

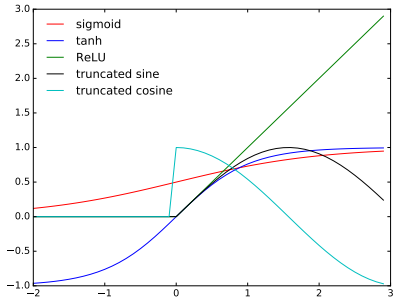


Fig. 7: Activation functions employed

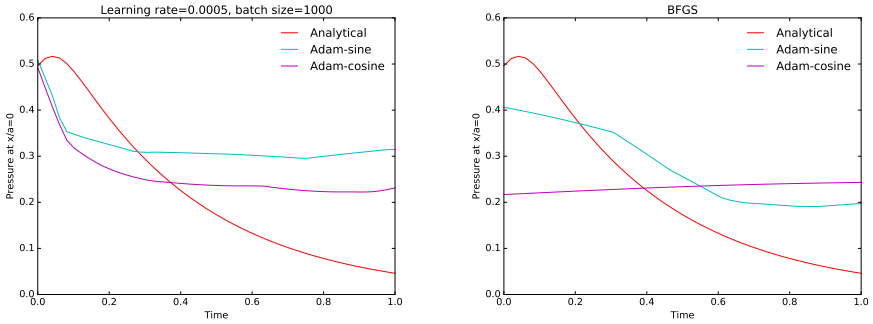


Fig. 8: Comparing of the deep learning framework with truncated sine and cosine activation functions

4 Conclusions

It is evident that a truncated sine activation function holds a lot of promise for the problem under consideration. It is also clear that the loss function definition needs to be looked at in more detail to carefully capture the non-monotonic pressure response. Aspects like regularization to avoid overfitting would be critical in this respect, since we would ideally like the deep learning framework to not be too dependent on the hyperparameters like number of layers and number of neurons per hidden layer as well as the number of data points being factored into consideration for training the deep learning model.

References

1. Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283, 2016.
2. Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch:

An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037, 2019.
3. James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a cpu and gpu math expression compiler. In *Proceedings of the Python for scientific computing conference (SciPy)*, volume 4, pages 1–7. Austin, TX, 2010.
4. Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.
5. Stéfan van der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in science & engineering*, 13(2):22–30, 2011.
6. Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.
7. Jian-Xun Wang, Jin-Long Wu, and Heng Xiao. Physics-informed machine learning approach for reconstructing reynolds stress modeling discrepancies based on dns data. *Physical Review Fluids*, 2(3):034603, 2017.
8. J Nathan Kutz. Deep learning in fluid dynamics. *Journal of Fluid Mechanics*, 814:1–4, 2017.
9. Romit Maulik and Omer San. A neural network approach for the blind deconvolution of turbulent flows. *Journal of Fluid Mechanics*, 831:151–181, 2017.
10. Arvind T. Mohan and Datta V. Gaitonde. A deep learning based approach to reduced order modeling for turbulent flow control using lstm neural networks, 2018.
11. Gavin D. Portwood, Peetak P. Mitra, Mateus Dias Ribeiro, Tan Minh Nguyen, Balasubramanya T. Nadiga, Juan A. Saenz, Michael Chertkov, Animesh Garg, Anima Anandkumar, Andreas Dengel, Richard Baraniuk, and David P. Schmidt. Turbulence forecasting via neural ode, 2019.
12. Soheil Esmaeilzadeh, Chiyu Max Jiang, Kamyar Azizzadenesheli, Karthik Kashinath, Mustafa Mustafa, Hamdi A Tchelepi, Philip Marcus, Mr Prabhat, and Anima Anandkumar. A deep learning based physics informed continuous spatio temporal super-resolution framework. *Bulletin of the American Physical Society*, 2020.
13. Ali Al-Aradi, Adolfo Correia, Danilo Naiff, Gabriel Jardim, and Yuri Saporito. Solving nonlinear and high-dimensional partial differential equations via deep learning. *arXiv preprint arXiv:1811.08782*, 2018.
14. Dengpeng Huang, Jan Niklas Fuhg, Christian Weißenfels, and Peter Wriggers. A machine learning based plasticity model using proper orthogonal decomposition. *arXiv preprint arXiv:2001.03438*, 2020.
15. Nikolaos N. Vlassis, Ran Ma, and WaiChing Sun. Geometric deep learning for computational mechanics part i: anisotropic hyperelasticity. *Computer Methods in Applied Mechanics and Engineering*, 371:113299, Nov 2020.
16. Maziar Raissi. Deep hidden physics models: Deep learning of nonlinear partial differential equations. *The Journal of Machine Learning Research*, 19(1):932–955, 2018.
17. Maziar Raissi, Paris Perdikaris, and George 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*, 378:686–707, 2019.
18. Ehsan Haghighat, Maziar Raissi, Adrian Moure, Hector Gomez, and Ruben Juanes. A deep learning framework for solution and discovery in solid mechanics. *arXiv preprint arXiv:2003.02751*, 2020.
19. Saumik Dana and Mary F Wheeler. A machine learning accelerated fe ^{2} homogenization algorithm for elastic solids. *arXiv preprint arXiv:2003.11372*, 2020.
20. Ehsan Haghighat and Ruben Juanes. Sciann: A keras/tensorflow wrapper for scientific computations and physics-informed deep learning using artificial neural networks, 2020.
21. Ravi G Patel, Indu Manickam, Nathaniel A Trask, Mitchell A Wood, Myoungkyu Lee, Ignacio Tomas, and Eric C Cyr. Thermodynamically consistent physics-informed neural networks for hyperbolic systems. *arXiv preprint arXiv:2012.05343*, 2020.
22. Teeratorn Kadeethum, Thomas M Jørgensen, and Hamidreza M Nick. Physics-informed neural networks for solving nonlinear diffusivity and biot's equations. *PloS one*, 15(5):e0232683, 2020.

23. T. Kadeethum, T. Jørgensen, and H. Nick. 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, 2020. American Rock Mechanics Association.
24. Yared W Bekele. Physics-informed deep learning for flow and deformation in poroelastic media. *arXiv preprint arXiv:2010.15426*, 2020.
25. Olga Fuks and Hamdi A. Tchelepi. Limitations of physics informed machine learning for nonlinear two-phase transport in porous media. *Journal of Machine Learning for Modeling and Computing*, 1(1):19–37, 2020.
26. Cedric G. Fraces, Adrien Papaioannou, and Hamdi Tchelepi. Physics informed deep learning for transport in porous media. buckley leverett problem, 2020.
27. Renee Swischuk, Laura Mainini, Benjamin Peherstorfer, and Karen Wilcox. Projection-based model reduction: Formulations for physics-based machine learning. *Computers & Fluids*, 179:704 – 717, 2019.
28. Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.
29. Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. Activation functions: Comparison of trends in practice and research for deep learning, 2018.
30. Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.
31. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, 2015.
32. J. Mandel. Consolidation des sols (étude mathématique)*. *Géotechnique*, 3(7):287–299, 1953.
33. Y. Abousleiman, A. H. D. Cheng, L. Cui, E. Detournay, and J. C. Roegiers. Mandel's problem revisited. *Géotechnique*, 46(2):187–195, 1996.
34. Saumik Dana, Benjamin Ganis, and Mary F. Wheeler. A multiscale fixed stress split iterative scheme for coupled flow and poromechanics in deep subsurface reservoirs. *Journal of Computational Physics*, 352:1–22, 2018.
35. Saumik Dana and Mary F Wheeler. Design of convergence criterion for fixed stress split iterative scheme for small strain anisotropic poroelastoplasticity coupled with single phase flow. *arXiv preprint arXiv:1912.06476*, 2019.
36. Saumik Dana. System of equations and staggered solution algorithm for immiscible two-phase flow coupled with linear poromechanics. *arXiv preprint arXiv:1912.04703*, 2019.
37. Saumik Dana, Joel Ita, and Mary F Wheeler. The correspondence between voigt and reuss bounds and the decoupling constraint in a two-grid staggered algorithm for consolidation in heterogeneous porous media. *Multiscale Modeling & Simulation*, 18(1):221–239, 2020.
38. S. Dana and M. F. Wheeler. Convergence analysis of fixed stress split iterative scheme for anisotropic poroelasticity with tensor biot parameter. *Computational Geosciences*, 22(5):1219–1230, 2018.
39. S. Dana and M. F. Wheeler. Convergence analysis of two-grid fixed stress split iterative scheme for coupled flow and deformation in heterogeneous poroelastic media. *Computer Methods in Applied Mechanics and Engineering*, 341:788–806, 2018.
40. S. Dana. *Addressing challenges in modeling of coupled flow and poromechanics in deep subsurface reservoirs*. PhD thesis, The University of Texas at Austin, 2018.
41. Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.
42. Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.
43. A Lee. pydoe: The experimental design package for python. *Python package version 0.3*, 8, 2015.
44. Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.
45. Andreas Griewank et al. On automatic differentiation. *Mathematical Programming: recent developments and applications*, 6(6):83–107, 1989.
46. Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.
47. Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.
48. Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.
49. Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.