COMPARISON OF TEMPERED AND TRUNCATED FRACTIONAL MODELS

HAYLEY A. OLSON*, MARTA D'ELIA†, MIKIL FOSS‡, MAMIKON GULIAN§, AND PETRONELA RADU¶

Abstract. Tempered fractional operators are able to model effects that classical partial differential equations cannot capture, such as the super- and sub-diffusive effects that are present in hydrology and geophysics models. However, tempered fractional operators are computationally intensive due to the infinite range of interaction for the integral operator. We analyze a truncated variation of the fractional operators, which are less computationally intensive, in an effort to use them in place of the more complex tempered variation. In particular, we train parameters of the truncated operator using neural networks in order to optimize the difference of the actions of the two operators.

1. Introduction. Fractional models are nonlocal models that can be utilized in place of standard partial differential equations (PDEs) when trying to model anomalous effects that standard PDEs fail to describe. For example, fractional models have found applications in subsurface diffusion and transport, turbulence, and machine-learning algorithms. Here, we are considering the tempered variation of the fractional Laplacian introduced in [5] which has applications in; e.g. hydrology and geophysics [1, 6].

These nonlocal operators, such as the tempered fractional Laplacian, are integral operators that act on a so-called "interaction horizon" that determine the radius of interaction between points in the domain. The tempered fractional Laplacian has an infinite interaction horizon. This is useful for modelling long range forces and it reduces the regularity requirements on the solution. However, the infinite interaction horizon causes the operator to be computationally complex. This work explores whether a similar operator can be generated that mimics the tempered fractional Laplacian while being less expensive computationally.

In particular, we investigate a truncated version of the fractional Laplacian, restricting the interaction horizon to a ball of radius $0 < \delta < \infty$. This truncation allows for a reduction in the computational cost of the numerical evaluation of the operator. The authors have studied these two operators previously in [3] and determined that the nonlocal fractional energy norms of the tempered fractional Laplacian and truncated fractional Laplacian, in its simplest form, are equivalent. In this work, we introduce a modified form of the truncated fractional Laplacian and we parametrize it with the goal of identifying the parameters that minimize the difference of the actions of the operators. Similar parameter identification problems for fractional models can be found in e.g. [2, 8]. In this work the parameters are trained using deep neural networks (DNNs). Machine learning has been used to tackle many aspects of nonlocal models, we mention [4] as an example of the use of DNNs for the approximation of the solution of a nonlocal equation.

This report is organized as follows. Section 2 includes relevant definitions and previous results that will be referenced throughout. The formulation of the problem as a loss function to optimize and characterization of the learned parameters is defined

*University of Nebraska-Lincoln, hayley.olson@huskers.unl.edu
†Sandia National Laboratories, mdelia@sandia.gov
‡University of Nebraska-Lincoln, mikil.foss@unl.edu
§Sandia National Laboratories, mgulian@sandia.gov
¶University of Nebraska-Lincoln, pradu@unl.edu

in Section 3. Section 4 has information about the discretization of the problem for the numerical analysis. Computational results of learning the parameters can be found in Section 5 with conclusions in Section 6.

2. Notation and Previous Work. Let $\Omega \in \mathbb{R}^n$ be an open bounded domain. Define the corresponding *interaction domain* as

$$\begin{aligned}\Omega_I &= \{\mathbf{y} \in \mathbb{R}^n \backslash \Omega \text{ such that } \mathbf{x} \text{ interacts with } \mathbf{y} \text{ for some } \mathbf{x} \in \Omega\} \\ &= \{\mathbf{y} \in \mathbb{R}^n \backslash \Omega : |\mathbf{x}-\mathbf{y}| \leq \delta \text{ for some } \mathbf{x} \in \Omega\},\end{aligned}$$

where $\delta > 0$ is the so-called interaction radius or horizon. For the tempered fractional operator, the operator has an infinite radius of interaction. Hence, $\delta = \infty$ and thus $\Omega_I = \mathbb{R}^n \setminus \Omega$.

We will be considering two fractional operators of the general form

$$\mathcal{L}u(\mathbf{x}) = \int_{\Omega \cup \Omega_I} \frac{u(\mathbf{y}) - u(\mathbf{x})}{|\mathbf{x}-\mathbf{y}|^{n+2s}} \gamma(\mathbf{x}, \mathbf{y}) d\mathbf{y},$$

where n is the dimension and $0 < s < 1$ is the fractional order. In particular, we will consider the tempered fractional Laplacian, introduced in [5], which has the kernel

$$\gamma_{\text{tem}}(\mathbf{x}, \mathbf{y}; \lambda) = e^{-\lambda|\mathbf{x}-\mathbf{y}|} \tag{2.1}$$

alongside a truncated version of the fractional Laplacian which, in the simplest form, has the kernel

$$\gamma_{\text{tr}}(\mathbf{x}, \mathbf{y}; \sigma, \delta) = \sigma \mathcal{X}(|\mathbf{x}-\mathbf{y}| < \delta), \tag{2.2}$$

where σ and δ are positive real numbers that represent a scaling constant and the horizon of truncation, respectively. The function $\mathcal{X}(\cdot)$ represents the indicator function; thus, the support of the truncated kernel is limited to a Euclidean ball of radius δ centered at $\mathbf{x}$.

With the purpose of improving the descriptive power of the truncated operator, we propose a modified version of γ_{tr} (that we denote by the same symbol) where both the scaling parameter and the horizon depend on the space variable. Keeping in mind that we will use optimization algorithms to learn these parameters, we substitute the indicator function with a smooth approximation, so as to facilitate the use of available machine-learning tools (this is discussed in more detail in Section 4). Thus, we define the new truncated kernel as follows:

$$\gamma_{\text{tr}}(\mathbf{x}, \mathbf{y}; \sigma, \delta) = \sigma(\mathbf{x}) \eta(|\mathbf{x}-\mathbf{y}|, \delta(\mathbf{x})), \tag{2.3}$$

where σ and δ are now space-dependent parameters. Here, for the sigmoid function $S(x) = \frac{1}{1+e^{-x}}$ and a fixed sharpness parameter $\alpha > 0$, we define η as

$$\eta(|\mathbf{x}-\mathbf{y}|, \delta(\mathbf{x})) = 1 - S(\alpha(|\mathbf{x}-\mathbf{y}| - \delta(\mathbf{x}))). \tag{2.4}$$

The fractional Laplacians with the tempered and truncated kernels will be referred to as $\mathcal{L}_{\text{tem}}$ and $\mathcal{L}_{\text{tr}}$, respectively.

Note that the use of the sigmoid function, supported in $\mathbb{R}$, might seem in contrast with the goal of restricting the domain of integration in the definition of the integral operator. However, such function decays to very small values which are negligible at the numerical level, i.e. values that are below machine precision. Thus, when

evaluating the sigmoid-truncated operator, one is allowed to restrict the integration domain to the set corresponding to function values above machine precision.

The tempered fractional Laplacian and the truncated fractional Laplacian in the simple form of equation (2.2) have been compared previously in [3], where their energy norms are shown to be equivalent. We summarize that result below. For a kernel γ, the *nonlocal fractional energy norm* is defined as

$$E(u;\mu) = \iint_{(\Omega\cup\Omega_I)^2} \frac{(u(\mathbf{x})-u(\mathbf{y}))^2}{|\mathbf{x}-\mathbf{y}|^{n+2s}}\gamma(\mathbf{x},\mathbf{y},\mu)d\mathbf{y}d\mathbf{x} \tag{2.5}$$

where μ is a parameter that determines the kernel. Then for γ_{tem} and γ_{tr} as defined in (2.1) and (2.2), we have the following result [3].

THEOREM 2.1. *There exist positive constants* $\underline{A}$ *and* $\overline{A}$ *such that, given* $\lambda > 0$,

$$\underline{A}E_{\text{tr}}(u;\delta) \le E_{\text{tem}}(u;\lambda) \le \overline{A}E_{\text{tr}}(u;\delta), \quad \forall\, u \in H^s_\Omega(\mathbb{R}^n),\ \delta < \infty. \tag{2.6}$$

This result, although valid only for the symmetric kernel in (2.2), provides the foundation for our proposed kernel representation and sets the groundwork for its mathematical analysis, which is the subject of our future work.

3. Formulation of the learning problem as the minimization of a loss function. We describe our procedure in a one-dimensional setting and we refer to the representation in equation (2.3). The goal is to learn functions $\sigma(x)$ and $\delta(x)$ such that they minimize the difference of the actions of the tempered and truncated fractional Laplace operators. We consider the operators acting on a set of training functions $\{u_i\}_{i=1}^N$ and minimize the following loss function, which is a percent error of the L^2 norm of the difference of the operators,

$$\text{Loss}(\delta,\sigma) = \frac{1}{N}\sum_{u_i} \frac{\|\mathcal{L}_{\text{tem}}u_i(x) - \mathcal{L}_{\text{tr}}u_i(x)\|_{L^2(\Omega)}}{\|\mathcal{L}_{\text{tem}}u_i(x)\|_{L^2(\Omega)}} \quad \text{for } x \in [-A, A]. \tag{3.1}$$

The training functions are constructed as linear combinations of basis functions, chosen with increasing levels of complexity and different regularity properties. In a one-dimensional setting, they are defined as follows. Let $\{x_i\}_{i=1}^N$ be a uniform partition of $[-a, a]$ and Δx the distance between the points. Note that $a \neq A$; indeed, in our numerical tests, $A < a$. The following functions are utilized, for some covering or scale parameter c:

1. Linear hat functions ($C^0(\mathbb{R})$):

$$u_i(x) = \begin{cases} \frac{1}{c}(x - x_i) + 1, & x_i - c < x \le x_i \\ \frac{-1}{c}(x - x_i) + 1, & x_i < x < x_i + c \\ 0, & \text{else.} \end{cases} \tag{3.2}$$

2. Exponential bump functions ($C^\infty(\mathbb{R})$):

$$u_i(x) = \begin{cases} \exp\left(\frac{c^2}{(c)^2-(x-x_i)^2}\right), & x_i - c < x < x_i + c \\ 0, & \text{else.} \end{cases} \tag{3.3}$$

3.1. Characterization of σ and δ. The parameters σ and δ are learned as the outputs of a deep neural networks (DNNs), i.e. $\delta(x) = DNN_1(x)$ and $\sigma(x) = DNN_2(x)$. The networks have the following structure. In addition, in some tests the parameter δ is trained as a constant.

The DNN for $\sigma(x)$ is a fully connected DNN with m nodes in each layer, referred to as the width of the network, and n-many hidden layers, referred to the depth of the network. All hidden layers have the same activation function, which is specified for each result below. Finally, the output layer is associated with a linear activation function (i.e., an affine map, which is common practice when using NNs for regression purposes). The DNN for $\delta(x)$ has an almost identical structure. However, the output layer is an affine map, composed with a custom ReLU activation to force the DNN to have a positive lower bound. This is to reflect the fact that $\delta(x)$ cannot take negative values. In place of a standard ReLU function that maps to $f(x) = \max\{0, x\}$, the custom ReLU function maps to $f(x) = \max\{\epsilon, x\}$ Here, $\epsilon = 0.05$ throughout.

4. Discretization of Operators, and Optimization. Several methods of discretization were employed in order to approximate the actions of the operators and minimize their difference. Here, we outline some of the main discretizations and approximations employed in our numerical tests. In all of the following experiments, we fix the fractional order $s = 0.25$.

For the computation of both the tempered and truncated fractional operators, the integrals are approximated as follows. Let $I = [-b, b]$ and $\{y_k\}_{k=1}^S$ a uniform partition of I such that $y_k \neq 0$ for any k and let Δy be the distance between the points. Then for a point x in the domain, we approximate the value of the operator acting on a function u at a the point x using a Riemann sum over the interval $[x-b, x+b]$. That is,

$$\mathcal{L}_\ell u(x) \approx \mathcal{A}_\ell u(x) := \sum_{k=1}^{S} \frac{u(x+y_k) - u(x)}{|y_k|^{n+2s}} \gamma_\ell(x, x+y_k)\Delta y,$$

where ℓ can refer to either the tempered or truncated operator. In the following results, $I = [-10, 10]$ and $S = 1000$, and thus $\Delta y = 0.02$.

Likewise, the L^2 norm of the operators is approximated using the ℓ^2 norm on a set of discrete points in the domain $\Omega = [-A, A]$. Specifically, let $\{x_j\}_{j=1}^R$ be a uniform partition of Ω.

Additionally, index the training functions as $\{u_i\}_{i=1}^N$. Let

$$f_{\gamma_\ell}(u_i(x_j))(x_j + y_k) = \frac{(u(x_j + y_k) - u(x_j))}{|y_k|} \gamma_\ell(x_j, x_j + y_k), \tag{4.1}$$

Then approximate the loss function (3.1) as

$$\text{Loss} \approx \frac{1}{N} \sum_{u_i} \left[\frac{\sum_{x_j} |\mathcal{A}_{\text{tem}} u_i(x_j) - \mathcal{A}_{\text{tr}} u_i(x_j)|^2}{\sum_{x_j} |\mathcal{A}_{\text{tem}} u_i(x_j)|^2} \right]^{1/2} \tag{4.2}$$

We recall that we are using the smoothed version of the truncated kernel (2.2), as discussed in Section 2. The reason of this approximation resides in the fact that δ cannot be a boolean variable during training, which is how a standard piecewise indicator function is defined. The indicator is then approximated using the sigmoid function as in equation (2.4), with $\alpha = 10$.

Width × Depth	**Training Loss**	**Test Error**
8×2	32.857 %	30.826 %
16×4	30.790 %	28.952 %
32×8	30.031 %	30.511 %
128×16	29.589 %	29.581 %

TABLE 5.1
Training and testing loss for different neural network sizes.

5. Computational Results. We report the results of the regression algorithm. In all our tests, the hidden layers have a hyperbolic tangent (tanh) activation function and we use the Adam optimizer.

For the first test, we train $\delta(x)$ and $\sigma(x)$ as NNs with 8 nodes in each layer and 2 hidden layers. The training and testing functions both utilize the exponential bump basis function as defined in (3.3). There are 32 training and 32 testing functions formed as linear combinations of 10 exponential bump basis functions spread evenly across the interval $[-8, 8]$.

Results are reported in Figure 5.1. The upper left and upper center plots contain the DNN prediction for the $\delta(x)$ and $\sigma(x)$ parameters, respectively, on the last iteration of the optimization. The lower left plot displays the training loss, which is the percent error defined in (4.2) for the set of training functions. The lower center plot displays the testing loss, which is computed in the same manner as the training loss but on a set of testing functions. The testing functions are distinct from the training functions (which are used to learn the parameters), but formed using the same basis functions as the training functions. This graph illustrates how well the learned parameters can generalize to functions outside the training set. The upper right plot is a random test function from the set of testing functions. In the lower right plot we report the values of the tempered and truncated operators acting on the random test function from the upper right plot, as well as the difference of the actions of the operators. The final value of the training loss indicates a percent error of 30%. While this result is not entirely satisfactory, we believe that by using a more stable optimization algorithm, such as L-BFGS, after training with Adam, lower loss values could be achieved. The value of the testing loss is of the same order of the training loss; this indicates that the learnt operator is as accurate when evaluated on (new) functions belonging to the same family, i.e. the set of bump functions.

We performed the same test with varying choices of depth and width of the neural networks used to train δ and σ. Result of these simulations are reported in Table 5.1. We note that the training and testing loss are not sensitive to the sizes of the NNs.

Due to the near-constant behavior of the $\delta(x)$ when trained as DNNs, we perform tests where this parameter is a positive real number, and learn this values as well as the neural network for $\sigma(x)$ using the same algorithm as described above. An example of these results is reported in Figure 5.2. Here, the plots represent the same quantities as in Figure 5.1.

With the purpose of testing the generalization properties of our learning procedure, we also perform cross-validation tests, i.e. we train the parameters using a set of basis functions and test the learnt operator on a set of functions generated with a different basis. In this test we kept all the parameters the same as in the first test with the exception of the shape of the basis functions used for the testing functions, which are generated by using the linear hat functions defined in (3.2). The corresponding

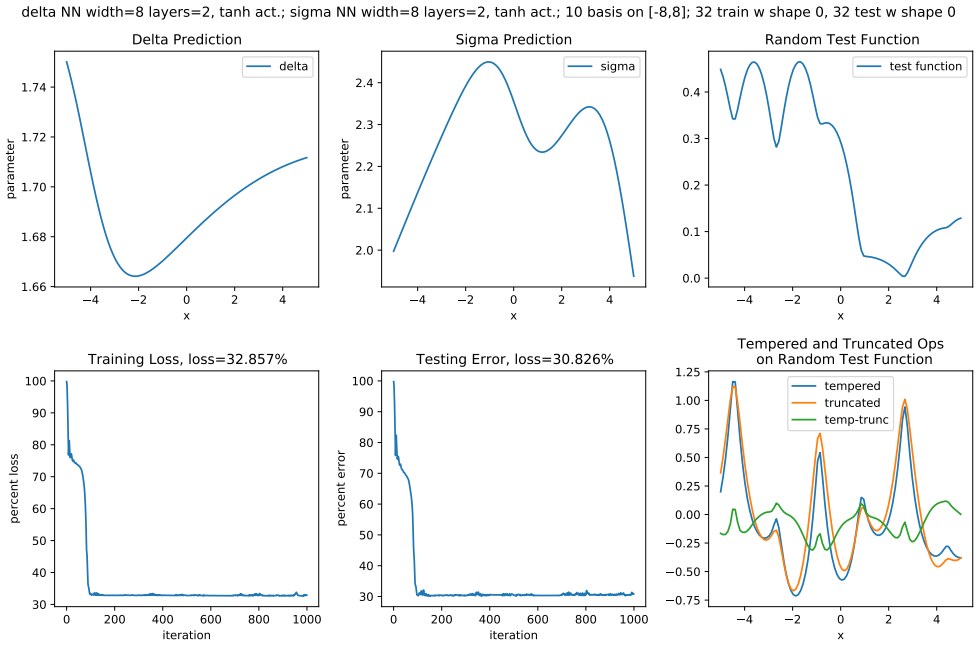

FIG. 5.1. *Result of training and testing NNs with width 8 and depth 2. Both training and testing functions are generate using bump basis functions.*

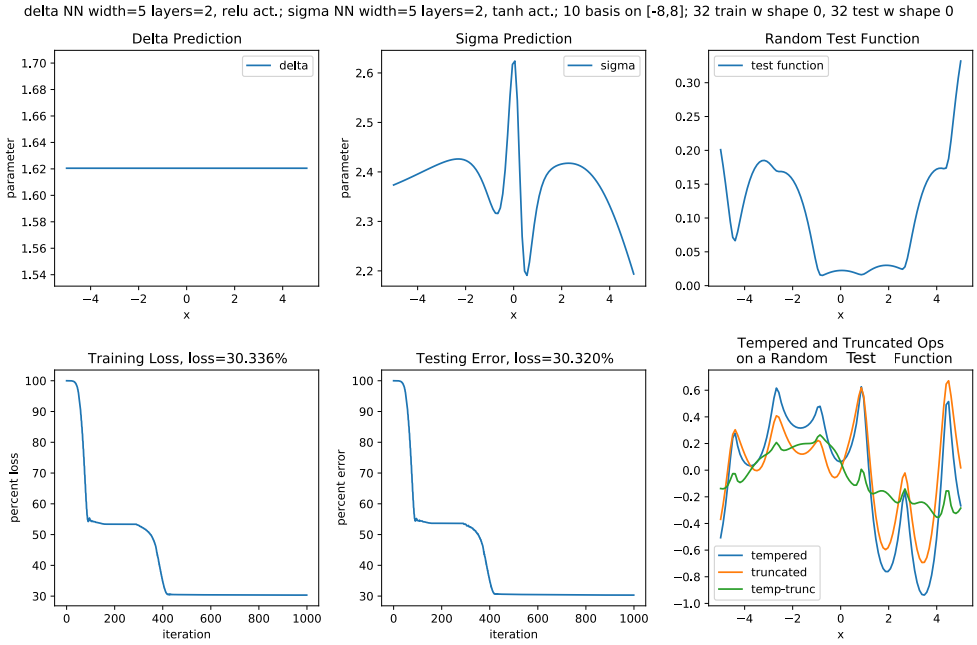

FIG. 5.2. *Results of training with the parameter δ trained as a constant value, and σ trained as neural network.*

results, for a new run of the algorithm, that results in slightly different $\delta(x)$ and $\sigma(x)$ than in Figure 5.1, are reported in Figure 5.3. We observe a testing percent error of 40%; the higher value is justified by the fact that the test functions are substantially different from the training functions, as they are generated using different basis functions.

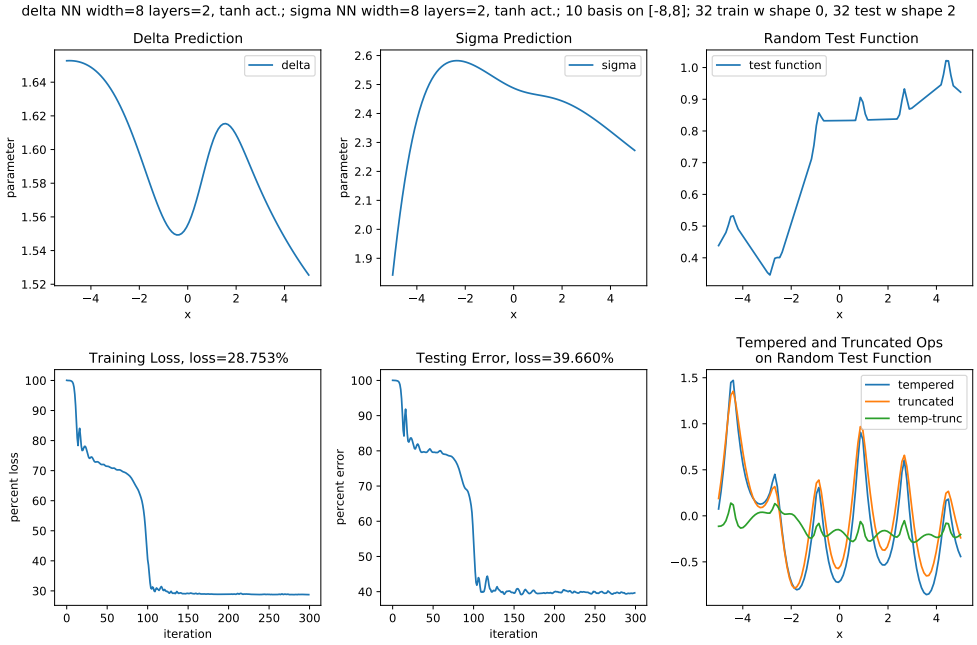

FIG. 5.3. *Result of training and cross-validation tests for NNs with width 8 and depth 2. The training functions are generated using bump basis functions whereas the testing functions are generated using hat functions.*

6. Conclusions. In this work we explored the approximation of the tempered fractional Laplacian by means of a truncated fractional Laplacian operator with the purpose of improving the efficiency of numerical computations. We propose a modified truncated fractional Laplacian operator parametrized by a scaling parameter and a horizon and we learn, via deep learning, the optimal values of these parameters such that the action of the truncated operator is as close as possible to the one of the tempered fractional Laplacian. Our results indicate that, for the studied basis set and training/test functions, the percent error between the truncated and the tempered operator is of the order of 30%, and that this is quite independent of the neural network architecture, or even whether δ is a constant value. The same accuracy is achieved when testing the optimal operator on new functions belonging to the same family of functions used for training. However, when tested on functions of a different family, the percent error increases to 40%.

While a deeper architecture does not improve the accuracy of the learnt operator, we believe that more sophisticated optimization algorithms may yield accuracy improvements. Thus, part of our planned follow-up work includes the use of improved optimization algorithms such as L-BFGS after training with the Adam optimizer to improve final test error. If this succeeds in improving training and test error, we also plan to run additional cross-validation tests to study how the qualitative differences

between the training and test sets effects generalization. Furthermore, we plan to utilize a symmetric form of the truncated operator by symmetrizing the truncated kernel as done in, e.g., [7], to reflect the symmetric nature of the tempered fractional kernel. Furthermore, we plan to study the differences in solutions to exterior value problems using the tempered Laplacian and the learned truncated operator, which may be significantly different that the observed test errors due to regularity.

7. Acknowledgments. This work was supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research under the Collaboratory on Mathematics and Physics-Informed Learning Machines for Multiscale and Multiphysics Problems (PhILMs) project. M.D. and M.G. are supported by Sandia National Laboratories (SNL), SNL is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration contract number DE-NA0003525. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

REFERENCES

[1] D. A. Benson, S. W. Wheatcraft, and M. M. Meerschaert, *Application of a fractional advection-dispersion equation*, Water Resources Research, 36 (2000), pp. 1403–1412.

[2] O. Burkovska, C. Glusa, and M. D'Elia, *An optimization-based approach to parameter learning for fractional type nonlocal models*, Computers & Mathematics with Applications, (2021).

[3] H. Olson, M. D'Elia, and M. Gulian, *The tempered fractional Laplacian as a special case of the nonlocal Laplace operator*, Computer Science Research Institute Summer Proceedings 2020. Technical Report SAND2020-12580R, (2020), pp. 111–126.

[4] G. Pang, M. D'Elia, M. Parks, and G. E. Karniadakis, *nPINNs: nonlocal Physics-Informed Neural Networks for a parametrized nonlocal universal Laplacian operator. Algorithms and Applications*, Journal of Computational Physics, 422 (2020), p. 109760.

[5] F. Sabzikar, M. M. Meerschaert, and J. Chen, *Tempered fractional calculus*, J. Comput. Phys., 293 (2015), p. 14–28.

[6] R. Schumer, D. A. Benson, M. M. Meerschaert, and B. Baeumer, *Multiscaling fractional advection-dispersion equations and their solutions*, Water Resources Research, 39 (2003), pp. 1022–1032.

[7] Y. Tao, X. Tian, and Q. Du, *Nonlocal models with heterogeneous localization and their application to seamless local-nonlocal coupling*, Multiscale Modeling & Simulation, 17 (2019), pp. 1052–1075.

[8] H. You, Y. Yu, N. Trask, M. Gulian, and M. D'Elia, *Data-driven learning of robust nonlocal physics from high-fidelity synthetic data*, Computer Methods in Applied Mechanics and Engineering, (2021).