Physics-Informed Deep Neural Networks for Learning Parameters and Constitutive Relationships in Subsurface Flow Problems

A. M. Tartakovsky[1], C. Ortiz Marrero[1], Paris Perdikaris[2], G.D. Tartakovsky[1], and D. Barajas-Solano[1]

[1]Pacific Northwest National Laboratory
[2]University of Pennsylvania

Key Points:

- Physics constraints improve the accuracy of machine learning methods, especially when learning from sparse data.
- Physics constraints allow learning constitutive relationships without direct observations of the quantities of interest.
- For considered examples, the proposed physics-informed neural networks provide a more accurate parameter estimation than the maximum a posteriori probability method.

Corresponding author: A.M. Tartakovsky, alexandre.tartakovsky@pnnl.gov

This article has been accepted for publication and undergone full peer review but has not been through the copyediting, typesetting, pagination and proofreading process, which may lead to differences between this version and the Version of Record. Please cite this article as doi: 10.1029/2019WR026731

Abstract

We present a physics-informed deep neural network (DNN) method for estimating hydraulic conductivity in saturated and unsaturated flows governed by Darcy's law. For saturated flow, we approximate hydraulic conductivity and head with two DNNs and use Darcy's law in addition to measurements of hydraulic conductivity and head to train these DNNs. For unsaturated flow, we approximate unsaturated conductivity function and capillary pressure with DNNs and train these DNNs using measurements of capillary pressure and the Richards equation. Because it is difficult to measure unsaturated conductivity in the field, we assume that no measurements of unsaturated conductivity are available. The proposed approach enforces the partial differential equation (PDE) (Darcy or Richards equation) constraints by minimizing the PDE residual at select points in the simulation domain. We demonstrate that physics constraints increase the accuracy of DNN approximations of sparsely observed functions and allow for training DNNs when no direct measurements of the functions of interest are available. For the saturated conductivity estimation problem, we show that the physics-informed DNN method is more accurate than the state-of-the-art maximum a posteriori probability method. For the unsaturated flow in homogeneous porous media, we find that the proposed method can accurately estimate the pressure-conductivity relationship based on the capillary pressure measurements only, even in the presence of measurement noise.

1 Introduction

In the last 20 years, data-driven discovery, including machine learning (ML), has emerged as the forth paradigm of science (in addition to experimental science, model-based science, and computational science) and has become a popular tool in hydrology. For example, deep neural networks (DNNs) have been used for flood and wind forecasting (Liu et al., 2017; Dalto et al., 2015), predicting fracture evolution in brittle materials (Schwarzer et al., 2019), modeling groundwater levels (Daliakopoulos et al., 2005), and uncertainty quantification in subsurface flow models (Zhu et al., 2019; Y. Yang & Perdikaris, 2019; Mo et al., 2019; Zhu & Zabaras, 2018; L. Yang et al., 2019).

Traditional ML methods, including DNNs, are trained only using data. In the following, we will refer to such methods as data-driven ML methods. To better understand what applications can benefit the most from data-driven ML methods, it is convenient to think about problems in terms of how much data is available versus how much physics is known (i.e., how few assumptions one should make to model the problem in question). The corresponding "physics-data" space is shown in Figure 1. For big-data problems, ML methods are able to discover patterns, classify different geological units, and predict the system behavior under a wide range of conditions, essentially acting as a surrogate model. Problems with fully known physics can be effectively modeled by differential and/or algebraic equations describing conservation laws. Predictive modeling of problems with partially known physics and sparse data still presents a significant challenge.

One guiding principle in placing the problem at question in the physics-data space can be Einstein's quote (Einstein, 1921):

"As far as the laws of mathematics refer to reality, they are not certain, and as far as they are certain, they do not refer to reality."

Physical models of many complex natural systems are at best only partially known. This is because conservation laws do not provide a complete system of differential equations. Accurate theoretical models for closing the system of conservation equations are available for homogeneous systems exhibiting time and length scale separation. Examples of accurate closures include Newtonian stress for homogeneous (Newtonian) fluids, Fick's law for mass flux in diffusion processes, and Darcy's law for fluid flux in porous media.

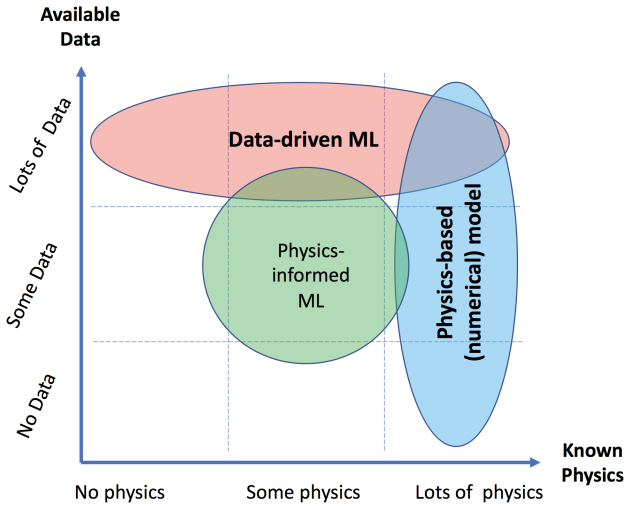


Figure 1: Data-reach versus physics-informed problems.

For more complex systems, including non-homogeneous turbulence, non-Newtonian fluid flow, multiphase flow in porous media, and granular materials, accurate theoretical closures are not available. Phenomenological constitutive relationships are used instead, which are usually only accurate for a narrow range of conditions. Even when sufficiently accurate closed-form partial differential equation (PDE) models are available, (space-dependent) parameters are typically unknown.

Because of high spatial heterogeneity of natural systems, including subsurface environments, there are usually not enough measurements to learn the spatial distribution of properties, e.g., hydraulic conductivity, from data only. Also, it is not always possible to measure relevant variables in the field to directly learn constitutive relationships. For example, it is not possible to obtain in situ measurements of unsaturated hydraulic conductivity to directly learn its relationship to the capillary pressure.

In this work, we focus on applications that have partially known physics and some (sparse) data. Specifically, we consider saturated flow in heterogeneous porous media with unknown conductivity and unsaturated flow in homogeneous porous media with an unknown relationship between capillary pressure and unsaturated conductivity. We propose a physics-informed ML method, which uses conservation laws in addition to data to train DNNs representing states (hydraulic head or capillary pressure), space-dependent coefficients (hydraulic conductivity), and constitutive relationships (the capillary pressure –relative conductivity relationship).

Traditional computational approaches for parameter estimation such as Bayesian inference and maximum a posteriori probability (MAP) estimation (Stuart, 2010; Carrera et al., 2005) cast this inherently ill-posed problem as an optimization or a statistical inference task. Solving these tasks requires repeated evaluation of expensive forward solvers until the parameters that minimize a given error metric are found or the space of parameter configurations given the available data is fully explored. This results in significant and often intractable computational cost. Furthermore, minimization via gradient-based methods requires computing gradients from said expensive forward models, which either requires additional computational cost or careful formulation of the adjoint problems. Additionally, forward modeling requires the knowledge of initial and boundary conditions, which usually are not fully known and therefore also must be estimated

from data together with unknown parameters and constitutive relations, significantly complicating parameter estimation.

Although significant progress has been made over the last two decades involving high-order schemes for PDEs, automatic differentiation (AD), PDE-constrained optimization, and optimization under uncertainty, parameter estimation in large-scale problems remains a significant challenge (Lieberman et al., 2010). Furthermore, existing approaches for learning unknown physics from partially known models and data are few and not fully mature. Unknown forms of equations at a given scale can either be found by upscaling (coarsening) known equations governing the same process at a smaller scale (e.g., Whitaker (2013)), or learning them from data. In this work, we are interested in the latter.

In the most general case, the dynamics of a system can be described as

$$\frac{\partial u(t)}{\partial t} = F(u(t)), \tag{1}$$

where F is a function or differential operator. A recent review of methods for learning F can be found in Rudy et al. (2018). These methods include the non-linear auto-regressive moving average with exogenous inputs (NARMAX) model (Chen & Billings, 1989), equation-free methods (Kevrekidis et al., 2003), Laplacian spectral analysis (Giannakis & Majda, 2012), and neural networks (Gonzalez-Garcia et al., 1998). Recently, data-driven approximations of Koopman operators, including dynamic mode decomposition (Williams et al., 2015), diffusion maps (Williams et al., 2015), delay coordinates (Giannakis, 2017; Brunton et al., 2017; Askham & Kutz, 2018), and neural networks (Yeung et al., 2017; Wehmeyer & Noé, 2018; Mardt et al., 2018; Lusch et al., 2018; Raissi, 2018; Zhu et al., 2019; Y. Yang & Perdikaris, 2019), have gained significant attention. This work is concerned with problems where conservation laws and other physical knowledge provide significantly more information about the structure of the operator $F(u)$. Specifically, we are interested in a steady state of the problem

$$\frac{\partial u(\mathbf{x}, t)}{\partial t} = -\nabla \cdot (K(\mathbf{x}, u)\nabla u(\mathbf{x}, t)), \tag{2}$$

where $K(\mathbf{x}, u)$ is an unknown constitutive relationship, which is a function of space and the PDE state. The boundary conditions may or may not be known. Among other physical phenomena, this equation describes flow in porous media (Bear, 2013). Two special cases of this problem are $K(\mathbf{x}, u) = K(\mathbf{x})$, where Eq (2) becomes a linear diffusion equation with heterogeneous diffusion coefficient $K(\mathbf{x})$; and $K(\mathbf{x}, u) = K(u)$, where Eq (2) becomes a nonlinear diffusion equation with a state-dependent coefficient $K(u)$. We are interested in learning $K(\mathbf{x}, u) = K(\mathbf{x})$ when measurements of u and K are available and $K(\mathbf{x}, u) = K(u)$ when only u measurements are available.

To achieve this objective, we propose a physics-informed DNN method, where PDEs and data are used to train DNN representations of the PDE states, unknown parameters, and constitutive relations. With application to Eq (2), this approach consists of defining two DNNs, one for $K(\mathbf{x}, u)$ and another for $u(\mathbf{x})$, together with auxiliary DNNs obtained by substituting K and u into Eq (2) and using AD to evaluate the right-hand side expression and boundary conditions. These networks are trained simultaneously using available data. Our work extends the physics-informed DNN (PINN) method proposed in Raissi et al. (2017a, 2017b) for finding unknown constants and solutions to PDEs given states observations. It is important to note that the PINN method was originally developed for time-dependent PDEs, and the extension of this method proposed herein also can be applied to time-dependent problems. In the remainder of this paper, we refer to our method as the PINN method.

The work is organized as follows: in Section 2, we introduce the PDE model and the PINN approach for parameter estimation and learning unknown physics. In Section 3, we provide a detailed study of the accuracy and performance of the proposed PINN

method for a linear parameter estimation problem with $K(\mathbf{x}, u) = K(\mathbf{x})$. In Section 4, we demonstrate the method's accuracy for learning the non-linear constitutive relationship $K(\mathbf{x}, u) = K(u)$. Discussion and conclusions are presented in Section 5.

2 PINN Approach

2.1 Feed-forward DNN approximation of unknown functions

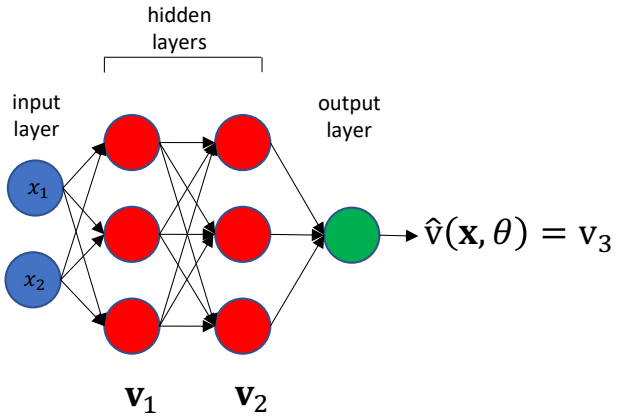


Figure 2: An example of a feed-forward DNN $\hat{v}(\mathbf{x}; \theta)$ (with parameters θ and $L = 2$ hidden layers) of a scalar function $v(\mathbf{x})$, where $\mathbf{x} = [x_1, x_2]^T$ is the two-dimensional position vector.

The starting point of our extended PINN method for parameter estimation and learning unknown physics is to approximate all unknown functions, including system states, space-dependent coefficients, and constitutive relationships, with fully connected DNNs. An example of a feed-forward DNN with two hidden layers for approximating a scalar function $v(\mathbf{x})$ is shown in Figure 2. The first layer is called the input layer, and the last layer is the output layer. The rest are hidden layers. The functional form of the DNN approximation $\hat{v}(\mathbf{x}; \theta)$ (with L hidden layers and parameters θ to be defined later) of a scalar function $v(\mathbf{x})$ is given as:

$$v(\mathbf{x}) \approx \hat{v}(\mathbf{x}; \theta) = v_{L+1}(\mathbf{v}_L(...(\mathbf{v}_1(\mathbf{x})))), \tag{3}$$

where $\hat{(\cdot)}$ denotes a DNN approximation, and

$$\begin{aligned} \mathbf{v}_1(\mathbf{x}) &= \sigma(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) \\ \mathbf{v}_2(\mathbf{v}_1) &= \sigma(\mathbf{W}_2\mathbf{v}_1 + \mathbf{b}_2) \\ &... \\ \mathbf{v}_L(\mathbf{v}_{L-1}) &= \sigma(\mathbf{W}_L\mathbf{v}_{L-1} + \mathbf{b}_L) \\ v_{L+1}(\mathbf{v}_L) &= \mathbf{W}_{L+1}\mathbf{v}_L + \mathbf{b}_{L+1}. \end{aligned} \tag{4}$$

Here, σ is the activation function, $\mathbf{x} \in \mathbb{R}^d$ is the d-dimensional coordinate vector and the input of the $\hat{v}(\mathbf{x}; \theta)$ DNN, d is the number of spatial dimensions, v_{L+1} is the scalar output of the DNN, $\mathbf{W}_i$ and $\mathbf{b}_i$ are the matrix of weights and vector of biases of the i^{th} layer, respectively. Together, these form the DNN parameter vector θ:

$$\theta = \{\mathbf{W}_1, \mathbf{W}_2, ..., \mathbf{W}_{L+1}, \mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_{L+1}\}. \tag{5}$$

The number of units in the input layer is equal to the dimensionality of the $\mathbf{x}$ vector. For the $d = 2$ and 3 spatial dimensions, the number of units in the input layer will be 2 and

3, respectively. The number of units in the output layer is equal to the dimensionality of v. For the scalar v, the output layer has one unit. The number of layers and the number of units in each layer depend on the smoothness of $v(\mathbf{x})$. Some of the commonly used activation functions include logistic sigmoid, hyperbolic tangent, ReLu, and leaky ReLu. Because we use DNNs to approximate the states of second-order PDEs, we require σ to be at least twice differentiable. Here, we adopt the hyperbolic tangent activation function $\sigma(x) = \tanh(x)$, which is infinitely differentiable. If a sufficient number N_v of measurements v^* of $v(\mathbf{x})$ at locations $\mathbf{x}_i^*$ $(i = 1, ..., N_v)$ is available, θ can be estimated by minimizing the loss function:

$$\theta = \arg\min_{\theta} \sum_{i=1}^{N_v} (\hat{v}(\mathbf{x}_i^*; \theta) - u^*(\mathbf{x}_i^*))^2. \tag{6}$$

Several methods are available for solving this optimization problem, including stochastic gradient descent (SGD), limited memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS), and conjugate gradient (CG) (Le et al., 2011). The relative performance of these methods depends on the number of DNN parameters, data size, and other factors. All these methods require computing gradients of $\hat{v}(\mathbf{x}, \theta)$ with respect to θ. Because the forms of σ and its derivatives are given in a closed form, the gradient of $\hat{v}(\mathbf{x}, \theta)$ with respect to θ can be computed exactly and fast. Most libraries for training DNNs, including the TensorFlow library used in this work (Ramsundar & Zadeh, 2018), employ AD (Baydin et al., 2015) to evaluate derivatives. Enforcing physics constraints in the form of PDE equations requires computing the derivatives of $\hat{v}(\mathbf{x}, \theta)$ with respect to the vector $\mathbf{x}$ components. In the PINN method, we use AD to evaluate the DNN's spatial derivatives, which allows physics PDE constraints to be enforced without numerically discretizing and solving the PDEs, as described in Section 2.2.

2.2 Enforcing physics constraints in DNN training

We consider a system (e.g., subsurface flow) that is characterized by a state $u(\mathbf{x})$ (e.g., hydraulic head or capillary pressure) and space- and/or state-dependent property (e.g., hydraulic conductivity) $K(u, \mathbf{x})$. We assume that N_K measurements of K and N_u measurements of u are available. We are interested in cases where N_K and N_u are not large enough to learn $K(u, \mathbf{x})$ and $u(\mathbf{x})$ from data only.

Furthermore, we assume that K and u satisfy the steady-state PDE

$$\mathcal{L}[u, K(\mathbf{x}, u)] = 0, \quad \mathbf{x} \in \Omega, \tag{7}$$

subject to the Dirichlet and Neumann boundary conditions

$$u(\mathbf{x}) = g(\mathbf{x}), \qquad \mathbf{x} \in \partial_D \Omega, \tag{8}$$

$$\mathbf{n} \cdot K(\mathbf{x}, u) \nabla u(\mathbf{x}) = q(\mathbf{x}), \qquad \mathbf{x} \in \partial_N \Omega. \tag{9}$$

Here, $\mathcal{L}$ is the known differential operator; $\Omega \subset \mathbb{R}^d$, $d \in [1, 3]$ is the simulation domain with the boundary $\partial\Omega$; $\partial_D\Omega$ and $\partial_N\Omega$ are the "Dirichlet" and "Neumann" parts of the boundary satisfying $\partial\Omega = \partial_D\Omega \cup \partial_N\Omega$ and $\partial_D\Omega \cap \partial_N\Omega \equiv \varnothing$; and $\mathbf{n}$ is the outward unit vector normal to $\partial\Omega$. We denote the state of the model by $u : \Omega \to \mathcal{U} \subseteq \mathbb{R}$ and the unknown constitutive relation by $K : \Omega \times \mathcal{U} \to \mathcal{K} \subseteq \mathbb{R}$.

In addition to K and u measurements, we assume that N_D measurements of g, and N_N measurements of q are collected at the locations $\{\mathbf{x}_i^K\}_{i=1}^{N_K}$, $\{\mathbf{x}_i^u\}_{i=1}^{N_u}$, $\{\mathbf{x}_i^D\}_{i=1}^{N_D}$, and $\{\mathbf{x}_i^N\}_{i=1}^{N_N}$, respectively. The observations are denoted by $K_i^* \equiv K(\mathbf{x}_i^K, u(\mathbf{x}_i^K))$ $(i = 1, \ldots, N_K)$, $u^* \equiv u(\mathbf{x}_i^u)$ $(i = 1, \ldots, N_u)$, $g_i^* \equiv g(\mathbf{x}_i^D)$ $(i = 1, \ldots, N_D)$, and $q_i^* \equiv q(\mathbf{x}_i^N)$ $(i = 1, \ldots, N_N)$.

To learn $K(\mathbf{x}, u)$, we approximate $u(\mathbf{x})$ and $K(\mathbf{x}, u)$ with DNNs:

$$u(\mathbf{x}) \approx \hat{u}(\mathbf{x}; \theta) \quad \text{and} \quad K(\mathbf{x}) \approx \hat{K}(\mathbf{x}, u; \gamma), \tag{10}$$

where θ and γ are the DNN parameters. Then, we use these two DNNs to approximate the residual of Eq (7):

$$f(\mathbf{x}) = \mathcal{L}[u, K(\mathbf{x}, u)] \approx \mathcal{L}\left[\hat{u}(\mathbf{x};\theta), \hat{K}\left(\mathbf{x}, \hat{u}\left(\mathbf{x};\theta\right);\gamma\right)\right] = \hat{f}(\mathbf{x};\theta,\gamma), \tag{11}$$

and the normal flux in the Neumann boundary condition (9):

$$f_N(\mathbf{x}) = \mathbf{n}\cdot K(\mathbf{x},u)\nabla u(\mathbf{x}) \approx \mathbf{n}\cdot \hat{K}\left(\mathbf{x}, \hat{u}(\mathbf{x};\theta);\gamma\right)\nabla\hat{u}(\mathbf{x};\theta) = \hat{f}_N(\mathbf{x};\theta,\gamma). \tag{12}$$

Here, spatial derivatives of $\hat{u}(\mathbf{x};\theta)$ and $\hat{K}\left(\mathbf{x}, \hat{u}\left(\mathbf{x};\theta\right);\gamma\right)$ are taken using AD. Next, we define the following loss function to train these four networks simultaneously:

$$\begin{aligned} L(\theta,\gamma) = &\frac{1}{N_K}\sum_{i=1}^{N_K}\left[\hat{K}(\mathbf{x}_i^K, \hat{u}(\mathbf{x}_i^K;\theta);\gamma) - K_i^*\right]^2 + \frac{1}{N_u}\sum_{i=1}^{N_u}[\hat{u}(\mathbf{x}_i^u;\theta) - u_i^*]^2 \\ &+ \frac{1}{N_D}\sum_{i=1}^{N_D}\left[\hat{u}(\mathbf{x}_i^D;\theta) - g_i^*\right]^2 + \frac{1}{N_N}\sum_{i=1}^{N_N}\left[\hat{f}_N(\mathbf{x}_i^N;\gamma,\theta) - q_i^*\right]^2 \\ &+ \frac{1}{N_c}\sum_{i=1}^{N_c}\hat{f}(\mathbf{x}_i^c;\gamma,\theta)^2. \end{aligned} \tag{13}$$

The first and second terms in L force the K and u DNNs to match the K and u measurements. The third and fourth terms enforce Dirichlet and Neumann boundary conditions. Finally, the fifth term enforces the PDE at N_c "collocation" points $\{\mathbf{x}_i^c\}_{i=1}^{N_c}$ that can be chosen uniformly or non-uniformly over Ω, depending on the problem.

The DNNs are trained, i.e., θ and γ are found, by minimizing the loss function (13), that is,

$$(\theta,\gamma) = \arg\min_{\theta,\gamma} L(\theta,\gamma). \tag{14}$$

The minimization is carried out using the L-BFGS-B method (Byrd et al., 1995), which is an extension of the L-BFGS method and is implemented in TensorFlow together with Xavier's normal initialization scheme (Glorot & Bengio, 2010). We use a quasi-Newton optimizer such as L-BFGS-B instead of SGD (a more common optimizer for DNNs) because of its superior rate of convergence, lower gradient vanishing, and lower computational cost for problems with a relatively small number of observed data (Raissi et al., 2019; Berg & Nyström, 2018; Le et al., 2011).

Note that the proposed method does not make any assumptions about the measurement noise. Also, our method can be easily extended to time-dependent PDEs by defining DNNs in Eq (10) as functions of both $\mathbf{x}$ and t (Raissi et al., 2017a, 2017b).

In the following two sections, we apply the PINN method to learn parameters and constitutive relationships in PDE models of the form (7)–(9).

3 Parameter estimation in a linear diffusion equation

In this section, we consider a linear diffusion equation with unknown diffusion coefficient $K(\mathbf{x})$,

$$\nabla\cdot(K(\mathbf{x})\nabla u(\mathbf{x})) = 0, \quad \mathbf{x}\equiv(x_1,x_2)^T \in (0,1)\times(0,1) \tag{15}$$

subject to the Dirichlet boundary conditions

$$u(\mathbf{x}) = 1, \quad x_2 = 0 \quad \text{and} \quad u(\mathbf{x}) = 0, \quad x_2 = 1 \tag{16}$$

and the Neumann boundary conditions

$$\frac{\partial u(\mathbf{x})}{\partial x_1} = 0 \quad x_1 = \{0,1\}. \tag{17}$$

Among other problems, this equation describes saturated flow in heterogeneous porous media with hydraulic conductivity $K(\mathbf{x})$ (Bear, 2013). We assume that N_K measurements of $K(\mathbf{x})$ and N_u measurements of $u(\mathbf{x})$ are available: $K_i^* \equiv K(\mathbf{x}_i^K)$ $(i = 1, \ldots, N_K)$ and $u_i^* \equiv u(\mathbf{x}_i^u)$ $(i = 1, \ldots, N_u)$. We approximate $K(\mathbf{x})$ and $u(\mathbf{x})$ with the DNNs $K(\mathbf{x}) \approx \hat{K}(\mathbf{x}; \gamma)$ and $u(\mathbf{x}) = \hat{u}(\mathbf{x}; \theta)$. These DNNs are used to approximate the residual of Eq (15):

$$\hat{f}(\mathbf{x}) = \nabla \cdot (K(\mathbf{x}) \nabla u(\mathbf{x})) \approx \hat{f}(\mathbf{x}; \theta, \gamma) = \nabla \cdot [\hat{K}(\mathbf{x}; \gamma) \nabla \hat{u}(\mathbf{x}; \theta)], \tag{18}$$

and the derivative of $u(\mathbf{x})$ with respect to x_2 that we need to enforce the boundary condition (BC) (17):

$$f_N(\mathbf{x}) = \frac{\partial u(\mathbf{x})}{\partial x_2} = \hat{f}_N(\mathbf{x}; \theta) = \frac{\partial \hat{u}(\mathbf{x}; \theta)}{\partial x_2}. \tag{19}$$

For this problem, the loss function takes the form:

$$\begin{aligned} L(\theta, \gamma) = & \frac{1}{N_K} \sum_{i=1}^{N_K} \left[\hat{K}(\mathbf{x}_i^K; \gamma) - K_i^* \right]^2 + \frac{1}{N_u} \sum_{i=1}^{N_u} [\hat{u}(\mathbf{x}_i^u; \theta) - u_i^*]^2 \\ & + \frac{1}{N_D} \sum_{i=1}^{N_D} \left[\hat{u}(\mathbf{x}_i^D; \theta) - g_i^* \right]^2 + \frac{1}{N_N} \sum_{i=1}^{N_N} \hat{f}_N(\mathbf{x}_i^N; \gamma, \theta)^2 \\ & + \frac{1}{N_c} \sum_{i=1}^{N_c} \hat{f}(\mathbf{x}_i^c; \gamma, \theta)^2. \end{aligned} \tag{20}$$

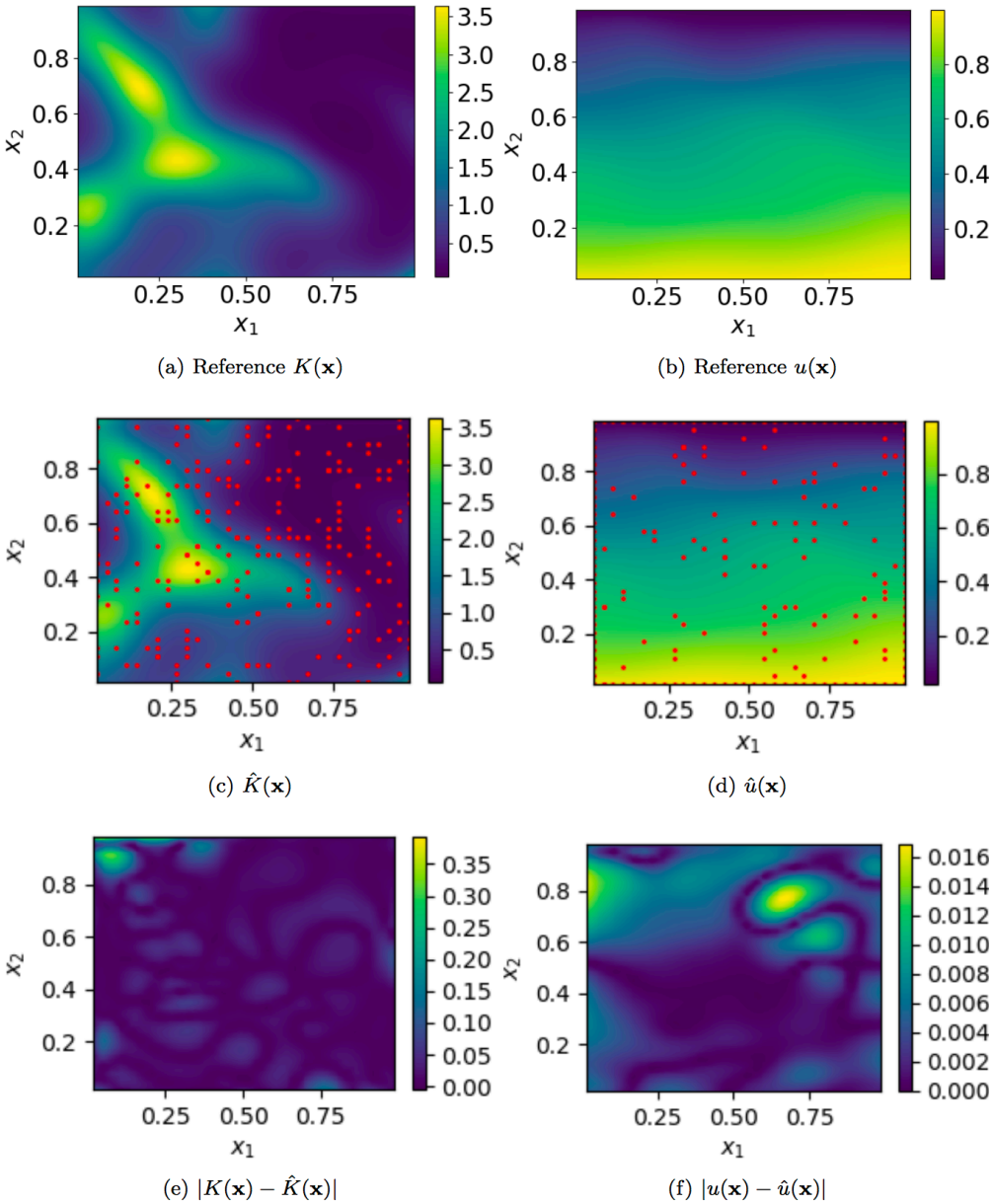
The DNNs are trained by minimizing the loss function (20) as described in Section 2. Throughout this work, we use feed-forward networks with three hidden layers and 50 units per layer. This DNN size is determined empirically to yield accurate results. Smaller DNNs might not be able to accurately approximate the unknown functions. Excessively large DNNs are difficult to train with the (relatively small) amount of data at hand. The optimal choice of DNN size is an active area of research (Claesen & De Moor, 2015) and is outside the scope of this work.

The L-BFGS-B method, used for solving the optimization problem (20), is an iterative method. The DNNs' training stops after changes in the loss function are smaller than the machine precision (i.e., $\frac{L^k - L^{k+1}}{\max(|L^k|, |L^{k+1}|, 1)} \leq ftol$) for 10 consecutive iterations, where $ftol = 2.220446049250313 \times 10^{-16}$. The iteration number for achieving convergence varies with the DNNs' initialization, data size, and the distribution of measurement locations. In this study, we observed that the required number of iterations for most simulations is in the range between 10000 and 50000. Therefore, we set a hard stop criterion of 50000 iterations, i.e., we stop the DNN training after 50000 iterations even if the full convergence is not achieved.

To demonstrate the proposed approach, we generate a reference $\ln K(\mathbf{x})$ field on a 32×32 regular mesh as a realization of the Gaussian process with zero mean and covariance function $C(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp(-\|\mathbf{x} - \mathbf{x}'\|^2 / 2\lambda^2)$, with $\sigma = 1$ and $\lambda = 0.15$. The reference u is generated by solving Eqs (15)–(17) using the finite volume (FV) method with the two-point flux approximation and the cell-centered 32×32 mesh. Figures 3(a) and (b) present the reference K and u fields, respectively. We randomly select the N_K and N_u FV cell centroids as measurement locations for K and u, respectively. These measurement locations are shown in Figure 3. To evaluate the loss function and train the DNNs, we use $N_c = 1024$ uniformly distributed collocation points.

We quantify the error between the estimated and reference K and u fields in terms of the relative L_2 errors, defined as

$$\varepsilon_u = \frac{\int_\Omega [u(\mathbf{x}) - \hat{u}(\mathbf{x})]^2 \, \mathrm{d}\mathbf{x}}{\int_\Omega u^2(\mathbf{x}) \, \mathrm{d}\mathbf{x}}, \quad \varepsilon_K = \frac{\int_\Omega \left[K(\mathbf{x}) - \hat{K}(\mathbf{x}) \right]^2 \mathrm{d}\mathbf{x}}{\int_\Omega K^2(\mathbf{x}) \, \mathrm{d}\mathbf{x}}.$$



(a) Reference $K(\mathbf{x})$ (b) Reference $u(\mathbf{x})$

(c) $\hat{K}(\mathbf{x})$ (d) $\hat{u}(\mathbf{x})$

(e) $|K(\mathbf{x}) - \hat{K}(\mathbf{x})|$ (f) $|u(\mathbf{x}) - \hat{u}(\mathbf{x})|$

Figure 3: Reference (a) K and (b) u fields. Estimated (c) $\hat{K}(\mathbf{x})$ and (d) $\hat{u}(\mathbf{x})$ fields. The red dots indicate the observation locations. Absolute point errors in (e) $\hat{K}(\mathbf{x})$, $|K(\mathbf{x}) - \hat{K}(\mathbf{x})|$, and (f) $\hat{u}(\mathbf{x})$, $|u(\mathbf{x}) - \hat{u}(\mathbf{x})|$. $N_K = 250$, $N_u = 100$, and $N_c = 1024$.

Figure 3 shows the estimated $\hat{K}$ and $\hat{u}$ with $N_K = 250$, $N_u = 100$, and $N_c = 1024$. The relative L_2 errors are $\varepsilon_u \approx 0.5\%$ and $\varepsilon_K \approx 1.7$ %. Figure 3 also depicts the point-wise absolute error in estimates of K and u. The point errors in K are concentrated in the upper left corner where no K measurements are available, with a maximum point error of approximately 30%. The point errors in u are much smaller (maximum error is approximately 1%) and more uniformly distributed throughout the domain.

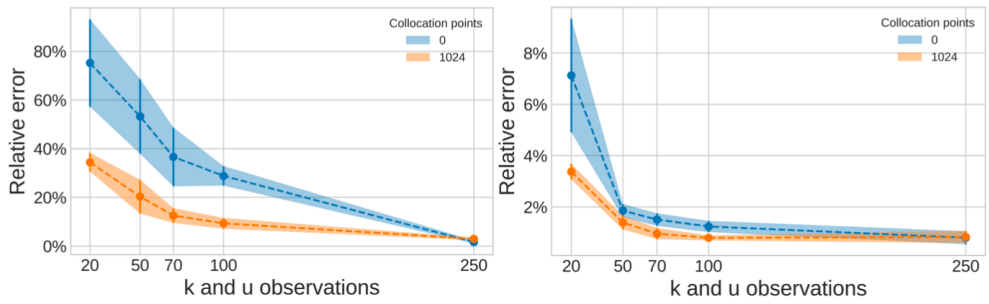


Figure 4: Mean and standard deviation (the width of the shaded area is equal to two standard deviations) of ε_u and ε_K as a function of $N = N_K = N_u$ obtained from $N_s = 10$ different network initializations. The u and K measurement locations are fixed, and $N_c = 0$ and $N_c = 1024$ collocation points are used.

Next, we study the effect of DNN initialization and the number of K and u measurements on the errors in the K and u fields, estimated with DNNs trained only on data ($N_c = 0$) and data and physics enforced at $N_c = 1024$ collocation points. For this purpose, we draw multiple initializations of the DNNs employing Xavier's scheme (see Section 2), and, for each initialization, we train the DNNs for the different numbers of $N = N_K = N_u$ measurements. We quantify the effect of initialization in terms of the mean and standard deviation of the relative L_2 errors of the estimated fields obtained for each initialization, i.e.,

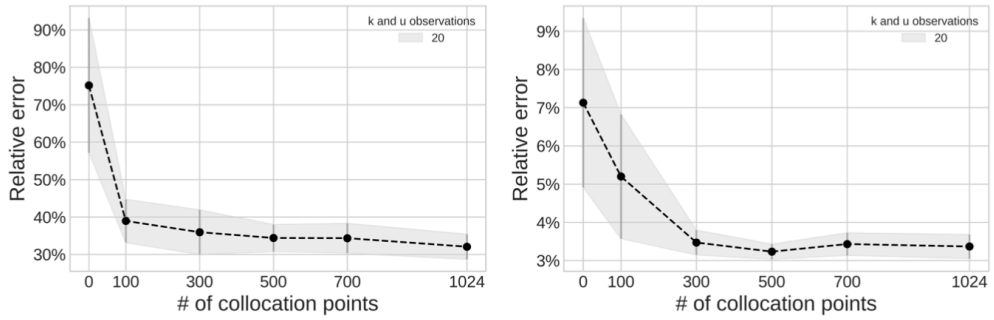
$$\overline{\varepsilon}_{(\cdot)} = \frac{1}{N_s}\sum_{i=1}^{N_s}\varepsilon_{(\cdot),i}, \quad \sigma_{\varepsilon_{(\cdot)}} = \sqrt{\frac{1}{N_s}\sum_{i=1}^{N_s}\left(\varepsilon_{(\cdot),i} - \overline{\varepsilon}_{(\cdot)}\right)^2},$$

where $\varepsilon_{(\cdot),i}$ is the relative L_2 error for either K or u for the i^{th} initialization, and N_s is the number of network initializations. Figure 4 shows the mean and standard deviation (the width of the shaded area is equal to two standard deviations) of ε_u and ε_K as a function of $N = N_K = N_u$ obtained from $N_s = 10$ different network initializations. The u and K measurement locations are the same for different realizations, and $N_c = 1024$ collocation points are used. As expected, $\overline{\varepsilon}_u$, $\overline{\varepsilon}_K$, σ_{ε_u}, and σ_{ε_K} decrease with increasing N for both the data-driven DNN ($N_c = 0$) and PINN ($N_c = 1024$) methods. For a small number of measurements (in this case, $N < 300$), the error in estimated K with data-driven DNNs is more than 100% larger than the error in K computed with the PINN method. On the other hand, for a large number of measurements ($N = 250$), the data-driven DNN method is slightly more accurate than the PINN method. A similar trend is observed for the errors in the estimated u.

There are three potential reasons for the data-driven DNN method being more accurate than the PINN method given a large number of measurements: (1) the number of measurements is sufficient to accurately train DNNs with data only; (2) adding physics constraints complicates the loss function landscape and makes it more difficult to find optimal weights; and (3) the physics model is not exact. In our case, there is a discrepancy between the Darcy's-law-based PDE model and data because the data is generated

using a weak form (finite volume) solution of the governing equations, while the strong form of the equation is enforced in the loss function. Of course, in real-life applications, physics models are always an approximation of the data. The results in Figure 4 confirm that enforcing physics is important for problems with a small number of measurements and is less beneficial for problems with a lot of data. For the considered K field, seven K measurements per correlation length is sufficient for the data-driven DNNs to be more accurate than the physics-based DNNs. It is safe to assume that for this number of measurements, any regression method would produce an accurate result. On the other hand, the PINN method can produce a useful prediction of K (30% error) with less than one measurement per correlation length ($N = 20$), while the data-driven DNNs fail in this case.

Figure 4 indicates that the mean error and uncertainty in $\hat{u}$ are much smaller than those in $\hat{K}$, i.e., $\overline{\varepsilon}_K \gg \overline{\varepsilon}_u$, and $\sigma_{\varepsilon_K} \gg \sigma_{\varepsilon_u}$. For both $\hat{u}$ and $\hat{K}$, the standard deviation associated with the initialization is approximately 10 times smaller than the mean value (i.e., the coefficient of variation is ≈ 0.1), which indicates that the initialization of the DNNs has a small effect on DNN predictions.



(a) $\overline{\varepsilon}_K$ and σ_{ε_K} (b) $\overline{\varepsilon}_u$ and σ_{ε_u}

Figure 5: Mean and standard deviation of the estimated (a) K and (b) u relative errors as a function of the number of collocation points N_c. For a given N_c, the DNNs are trained 10 times for different configurations of collocation points to compute the mean and variance of the relative errors. The u and K measurement locations are fixed, and $N = N_K = N_u = 20$. Shaded area width is equal to two standard deviations. For $N_c = 0$, the error bars represent the uncertainty in weight initialization.

To study the effect of the number and location of collocation points, we compute the relative errors ε_u and ε_K as a function of the number and location of the collocation points. Figures 5(a) and (b) show the mean and standard deviation of the relative errors versus N_c for $N = N_u = N_K = 20$. For a given N_c, the DNNs are trained $N_s = 10$ times for different locations of the collocation points using Latin hypercube sampling (McKay et al., 1979) and the mean and variance of the relative errors are computed. The error in $\hat{K}$ is about eight times larger than in $\hat{u}$. The mean and standard deviation of ε_u and ε_K decrease with increasing N_c until they reach asymptotic values at approximately $N_c = 300$, which is approximately 33% of the number of grid points used to generate the ground truth K field and simulate the ground truth solution for u. For $N_c = 0$ (which corresponds to the data-driven DNNs trained without any physics), the mean error and the standard deviation are computed from 10 simulations corresponding to different initializations of the DNNs. The data-driven DNN has a mean error in the estimated K of more than 70% (relative to less than 40% in the PINN method with $N_c = 100$) and standard deviation of $\approx 20\%$ (as compared to 5% in the physics-based DNN).

On the other hand, the mean error in the estimated u increases less than 2% and the standard deviation of the error increases less than 1% as the number of collocation points decreases from 100 to 0. This demonstrates that for estimating $K(x)$ using 20 K and 20 u measurements, physics constraints are essential. For small N_c, the location of collocation points has a notable effect on the error in K and u, which is evident from the relatively large coefficients of variation $\sigma_{\varepsilon_K}/\bar{\varepsilon}_K$ and $\sigma_{\varepsilon_u}/\bar{\varepsilon}_u$. The large coefficient of variation indicates that the number of collocation points should be increased.

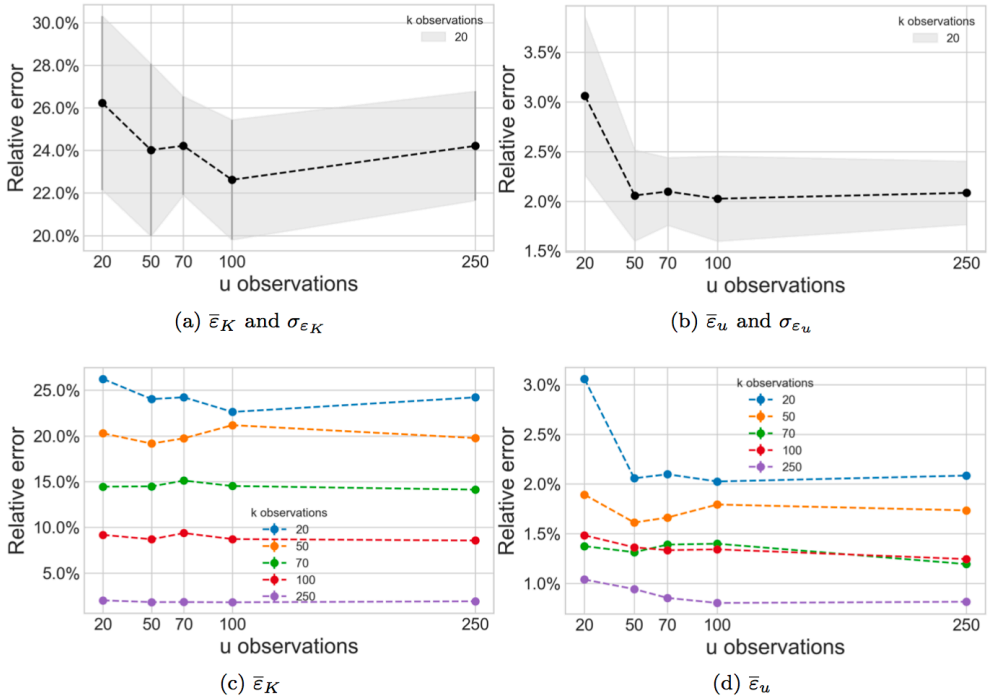
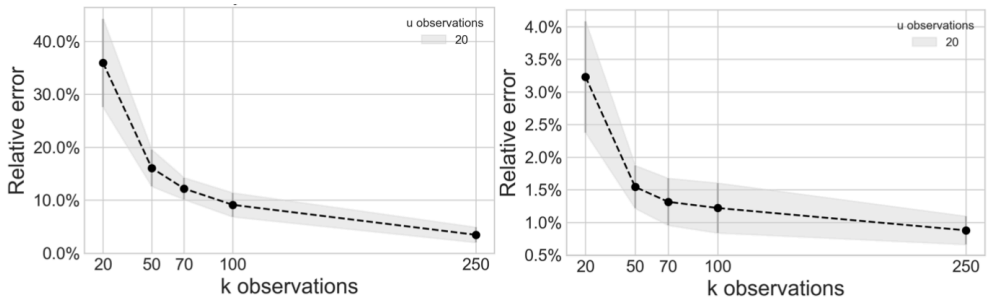


Figure 6: (a) and (b): Mean relative L_2 error of the predicted K and u as a function of the number of u observations. The number of K observations is 20. Shaded area width is equal to two standard deviations computed for 11 different configurations of u observations. (c) and (d): Mean relative L_2 error of the predicted K and u as a function of the number of u and K observations.

Figures 6 and 7 show how the number of K observations versus the number of u observations affects ε_u and ε_K. Here, the number of collocation points is $N_c = 1024$. Figures 6(a) and (b) depict the effect of N_u on the mean and standard deviation of ε_u and ε_K for $N_K = 20$. For each N_u value, we estimate $\hat{K}$ and $\hat{u}$ with $N_s = 10$ different distributions of the u measurement locations generated with the Latin hypercube sampler. Then, we compute $\bar{\varepsilon}_u$, $\bar{\varepsilon}_K$, σ_{ε_u}, and σ_{ε_K}. We see that N_u does not significantly affect ε_K with $\bar{\varepsilon}_K \approx 24\%$ and $\sigma_{\varepsilon_K} \approx 6\%$. For $\hat{u}$, $\bar{\varepsilon}_u$ deceases from more than 3% for $N_u = 20$ to $\approx 2\%$ for $N_u > 100$. The standard deviation σ_{ε_u} is 1.5% for $N_u = 20$ and less than 1% for $N_u > 50$.

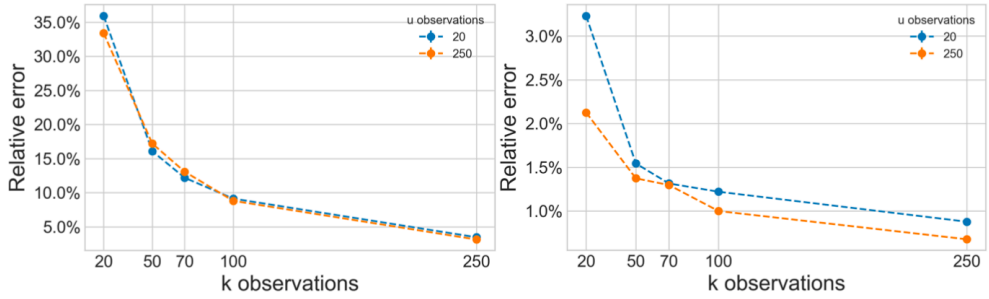
Figures 6 (c) and (d) show $\bar{\varepsilon}_K$ and $\bar{\varepsilon}_u$ as functions of N_u for different N_K. For all considered N_K, $\bar{\varepsilon}_K$ is practically independent of N_u and decreases with increasing N_K. On the other hand, $\bar{\varepsilon}_u$ decreases with increasing N_u and/or N_K.

Figures 7(a) and (b) reveal that all $\bar{\varepsilon}_u$, $\bar{\varepsilon}_K$, σ_{ε_u}, and σ_{ε_K} decrease with increasing N_K for fixed N_u. Alternatively, figures 7(c) and (d) demonstrate that N_u has a relatively



(a) $\bar{\varepsilon}_K$ and σ_{ε_K}

(b) $\bar{\varepsilon}_u$ and σ_{ε_u}



(c) $\bar{\varepsilon}_K$

(d) $\bar{\varepsilon}_u$

Figure 7: (a) and (b): Mean relative L_2 error of the predicted K and u as a function of the number of K observations. The number of u observations is 20. Shaded area width is equal to two standard deviations computed for 11 different configurations of K observations. (c) and (d): Mean relative L_2 error of the predicted K and u as a function of the number of u and K observations.

minor effect on $\bar{\varepsilon}_u$ and $\bar{\varepsilon}_K$. In all considered cases, $\bar{\varepsilon}_u$ is almost an order of magnitude smaller than $\bar{\varepsilon}_K$.

The main conclusion to be drawn from figures 6 and 7 is that K measurements are more important than u measurements for reducing error in $\hat{K}$ and $\hat{u}$. This may be partially explained by the fact that $u(\mathbf{x})$ is much smoother than $K(\mathbf{x})$, and a relatively small number of u measurements is needed to describe the u field. Beyond this number (approximately 50 for this example), additional u measurements do not have a significant effect on $\bar{\varepsilon}_u$ and $\bar{\varepsilon}_K$.

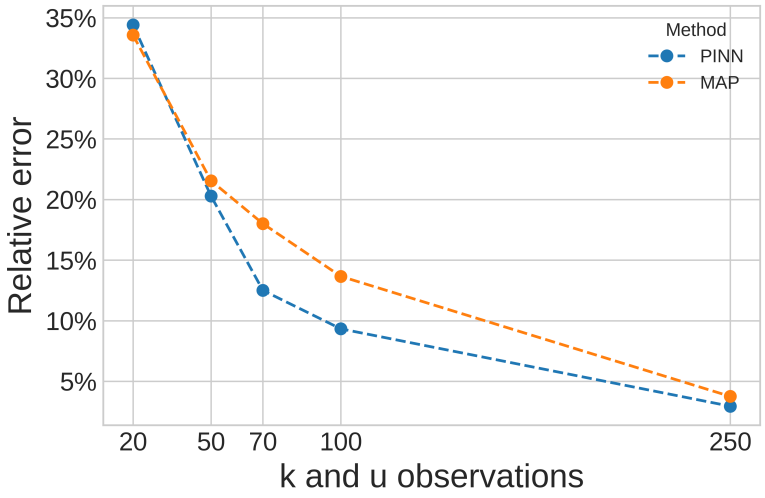


Figure 8: Comparison between L_2 error in the MAP estimate of K and mean L_2 error $\bar{\varepsilon}_K$ in the DNN K estimate.

Finally, we compare our approach against MAP estimation. In MAP, we penalize the norm of the gradient of K, which promotes smoother estimates of K (Barajas-Solano et al., 2014). For this regularizer and the regular FV discretization with $M = 1024$ cells, the MAP estimate $\hat{\mathbf{k}} \in \mathbb{R}^M$ is defined as the vector of cell-centered values of K computed as the solution to the minimization problem

$$\begin{aligned} \hat{\mathbf{k}} = \arg\min_{\mathbf{k}} \quad & \|\mathbf{u}^* - \mathbf{H}_u \mathbf{u}\|_2^2 + \|\ln \mathbf{k}^* - \mathbf{H}_K \ln \mathbf{k}\|_2^2 + \gamma \|\mathbf{L} \ln \mathbf{k}\|_2^2 \\ \text{subject to} \quad & \mathbf{l}(\mathbf{u}, \mathbf{k}) = \mathbf{0}, \end{aligned} \tag{21}$$

where $\mathbf{k}^* \equiv (K_1^*, \ldots, K_{N_K}^*)^\top$ and $\mathbf{u}^* \equiv (u_1^*, \ldots, u_{N_u}^*)^\top$ are the vectors of K and u observations, respectively; $\mathbf{H}_K \in \mathbb{R}^{N_K \times M}$ and $\mathbf{H}_u \in \mathbb{R}^{N_u \times M}$ are observation operators; $\mathbf{u}$ is the discretized state; $\mathbf{l}(\mathbf{u}, \mathbf{k}) = \mathbf{0}$ is the discretized problem (15)–(17); $\mathbf{L}$ is the discrete gradient operator; and $\gamma > 0$ is a regularization coefficient. For the considered problems, we find that the smallest L_2 error is obtained with $\gamma = 10^{-6}$. The minimization problem (21) is solved via the Levenberg–Marquardt (LM) algorithm (Barajas-Solano et al., 2014).

Figures 8 and 9 compare the DNN prediction and the MAP estimate. Figure 8 presents ε_K as a function of $N = N_K = N_u$ for $\hat{K}$ found from the two methods. The PINN method produces $\hat{K}$ with smaller errors for all considered N. Figure 9 depicts $\hat{K}$ obtained from the two methods for $N = 50$. The $\hat{K}$ field estimated from the PINN method is significantly smoother (and closer to the reference K shown in Figure 3)(a) than that estimated from MAP, even though the ε_K errors for the two methods are relatively similar: 19% for NN versus 22% for MAP. The tent-like character of the MAP prediction

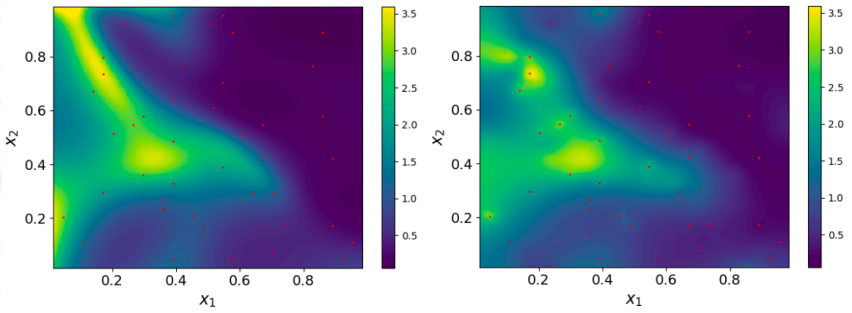


Figure 9: Comparison between the DNN prediction of K (left) and the MAP prediction of K (right) using 50 observations of both u and K.

in Figure 9 stems from the discrepancy with respect to observations being penalized more than the smoothness of the estimate (result of a relatively small γ). We find that larger γ results in a smoother field, but also in a larger prediction error.

Both the MAP and PINN methods involve an objective-function minimization. In addition, MAP estimation via gradient-based optimization algorithms (such as the LM algorithm) requires computing the gradient of the predicted observations, $\mathbf{H}_u\mathbf{u}$ and $\mathbf{H}_K \ln \mathbf{k}$, with respect to $\mathbf{k}$. For an FV discretization, this is done via the discrete adjoint method. The total cost for each iteration of the LM algorithm is one forward solution of the PDE problem to evaluate the objective function and one adjoint solution to compute the gradient. Therefore, MAP requires careful discretization of the PDE problem and formulation and solution of the corresponding adjoint problem. In contrast, in the PINN method, both spatial derivatives and the gradients with respect to DNN parameters are computed via AD, and the methodology does not require solving the PDE problem or formulating an adjoint problem. Finally, significant gains can be achieved in the PINN method performance by employing graphics processing unit (GPU) accelerators for training DNNs (L. Yang et al., 2019). GPUs are efficient for DNN training because they have many more resources and faster memory bandwidth, and DNN computations (mostly, matrices multiplication) are very fast on the GPUs.

We find that the PINN method time needed to obtain a solution varies with the number and location of measurements and collocation points. For example, when using 12 CPUs and an RTX2080ti GPU, a problem with 50 observations of both u and K fields and 1024 collocation points takes approximately 9 minutes to converge after 14213 iterations.

The extension of the PINN method to three-dimensional problems is trivial; it only requires modifying the DNNs to be functions of the three-dimensional position vector $\mathbf{x}$ instead of the two-dimensional position vector $\mathbf{x}$, i.e., increasing the input layer size of these DNNs from two units to three units. This change would increase the size of the DNNs (the number of unknown coefficients in DNNs), the number of collocation points, and, as a result, the cost of the PINN method.

4 Nonlinear diffusion equation

In this section, we consider a nonlinear diffusion equation with unknown state-dependent diffusion coefficient $K(u)$,

$$\nabla \cdot [K(u)\nabla u(\mathbf{x})] = 0, \quad (x_1, x_2) \in (0, L_1) \times (0, L_2) \tag{22}$$

subject to the boundary conditions

$$u(\mathbf{x}) = u_0, \qquad x_1 = L_1, \tag{23}$$

$$-K(u)\frac{\partial u(\mathbf{x})}{\partial x_1} = q, \qquad x_1 = 0 \tag{24}$$

$$\frac{\partial u(\mathbf{x})}{\partial x_2} = 0, \qquad x_2 = \{0, L_2\}. \tag{25}$$

This is a two-dimensional Richards equation describing a horizontal unsaturated flow in a homogeneous porous medium, where $u(\mathbf{x})$ is the water pressure and $K(u)$ is the pressure-dependent unsaturated conductivity of the porous medium (Bear, 2013). In practice, $K(u)$ is difficult to measure directly. Therefore, in this work we assume that no measurements of $K(u)$ are available and that only N_u measurements of u are given.

We approximate the unknown $u(\mathbf{x})$ and $K(u)$ functions with two DNNs,

$$u(\mathbf{x}) \approx \hat{u}(\mathbf{x};\theta), \quad K(u) \approx \hat{K}(u;\gamma). \tag{26}$$

These two DNNs are used to approximate the residual of the PDE (22),

$$f(\mathbf{x}) = \nabla \cdot \left[K\left(u(\mathbf{x})\right)\nabla u(\mathbf{x})\right] \approx \hat{f}(\mathbf{x};\theta,\gamma) = \nabla \cdot \left[\hat{K}\left(\hat{u}(\mathbf{x};\theta);\gamma\right)\nabla \hat{u}(\mathbf{x};\theta)\right], \tag{27}$$

and the x_1 and x_2 components of flux $\mathbf{f}_N(\mathbf{x};\phi,\gamma) = -K\left(u(\mathbf{x})\right)\nabla u(\mathbf{x})$ in the boundary conditions (24) and (25), so that

$$\mathbf{f}_N(\mathbf{x};\phi,\gamma) = -K\left(u(\mathbf{x})\right)\nabla u(\mathbf{x}) \approx \hat{\mathbf{f}}_N(\mathbf{x};\theta,\gamma) = -\hat{K}\left(\hat{u}(\mathbf{x};\theta);\gamma\right)\nabla \hat{u}(\mathbf{x};\theta). \tag{28}$$

Note that the DNN $\hat{\mathbf{f}}_N$ is a vector, i.e., $\hat{\mathbf{f}}_N = [f_N^{(x_1)}, f_N^{(x_2)}]^T = [-\hat{K}\partial\hat{u}/\partial x_1, -\hat{K}\partial\hat{u}/\partial x_2]^T$. Then, the loss function becomes

$$\begin{aligned}\mathcal{L}(\theta,\gamma) = &\frac{1}{N_u}\sum_{i=1}^{N_u}\left[\hat{u}(\mathbf{x}_i^u;\theta) - u_i^*\right]^2 \\ &+ \frac{1}{N_c}\sum_{i=1}^{N_c} f(\mathbf{x}_i;\theta,\gamma)^2 + \frac{1}{N_D}\sum_{i=1}^{N_D}\left[\hat{u}(\mathbf{x}_i^D;\theta) - u_0\right]^2 \\ &+ \frac{1}{N_N^{(x_1)}}\sum_{i=1}^{N_N}\left[f_N^{(x_1)}(\mathbf{x}_i^{N,x_1};\phi,\gamma) - q\right]^2 + \frac{1}{N_N^{(x_2)}}\sum_{i=1}^{N_N}\left[f_N^{(x_2)}(\mathbf{x}_i^{N,x_2};\phi,\gamma)\right]^2,\end{aligned}$$

where $\mathbf{x}_i^{N,x_1}$ $(i = 1, ..., N_N^{(x_1)})$ are the collocation points on the Neumann boundary $(x_1 = 0, x_2)$ and $\mathbf{x}_i^{N,x_2}$ $(i = 1, ..., N_N^{(x_2)})$ are the collocation points on the Neumann boundaries $(x_1, x_2 = 0)$ and $(x_1, x_2 = L_2)$.

This model is tested with data generated using the Subsurface Transport Over Multiple Phases (STOMP) code (White et al., 1995) with the van Genuchten model (Van Genuchten, 1980) for the $K(u)$ function

$$K(s(u)) = K_s s^{\frac{1}{2}}\left[1 - \left(1 - s^{\frac{1}{m}}\right)^m\right]^2, \tag{29}$$

$$s(u) = \left\{1 + \left[\alpha(u_g - u)\right]^{\frac{1}{1-m}}\right\}^{-m}. \tag{30}$$

Here, K_s is the saturated hydraulic conductivity, $u_g = \frac{P_g}{\rho g}$, P_g is the air pressure, ρ is the density, g is gravity, and α and m are the van Genuchten parameters. The following parameter values are used in the STOMP simulation: $u_0 = -10$ m, $\alpha = 0.1$, $m = 0.469$, $q = 8.25 \times 10^{-5}$ m/s, $u_g = 0$, and $K_s = 8.25 \times 10^{-4}$ m/s.

Figure 10 shows the reference $u(\mathbf{x})$ field generated with STOMP and the assumed locations of u measurements. Figure 11 presents the estimated $\hat{K}(u)$ function and the

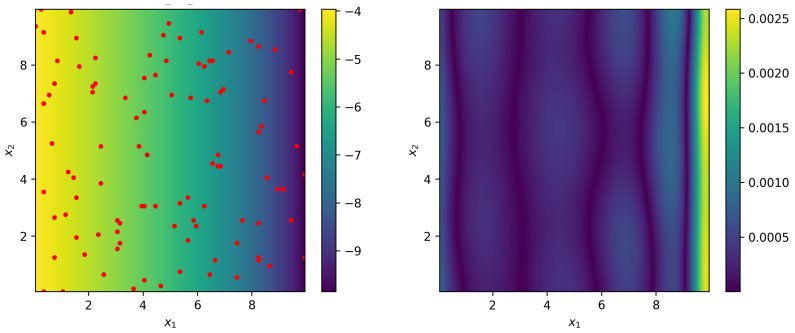


Figure 10: (Left) Referenced $u(\mathbf{x})$ field generated using STOMP with the van Genuchten model for $K(u)$ and the locations of u observations. (Right) Absolute error in the $u(\mathbf{x})$ field estimated with the PINN method.

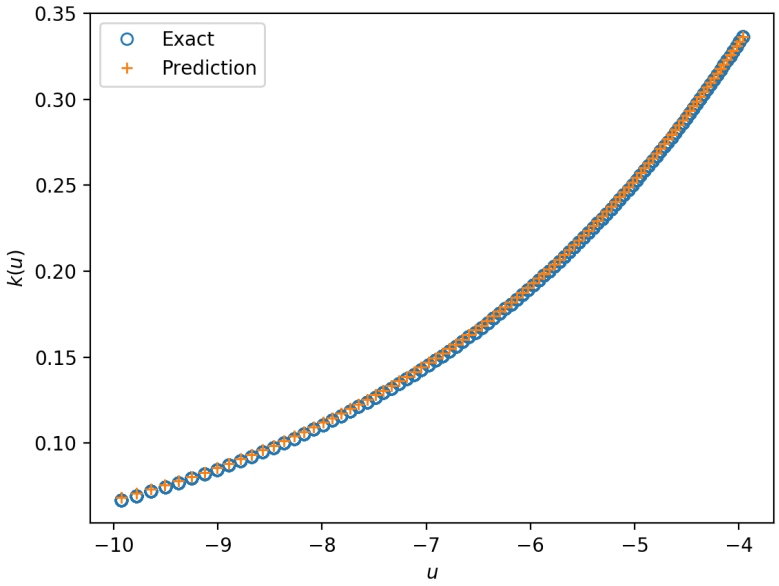


Figure 11: Comparison of the estimated $\hat{K}(u)$ and the reference $K(u)$ given by the van Genuchten model.

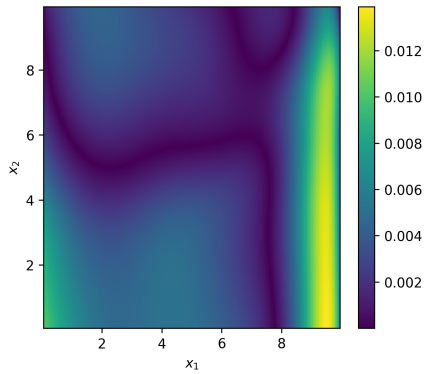


Figure 12: Absolute error in $\hat{u}(\mathbf{x})$ in the non-linear diffusion (unsaturated flow) problem in the presence of measurement noise.

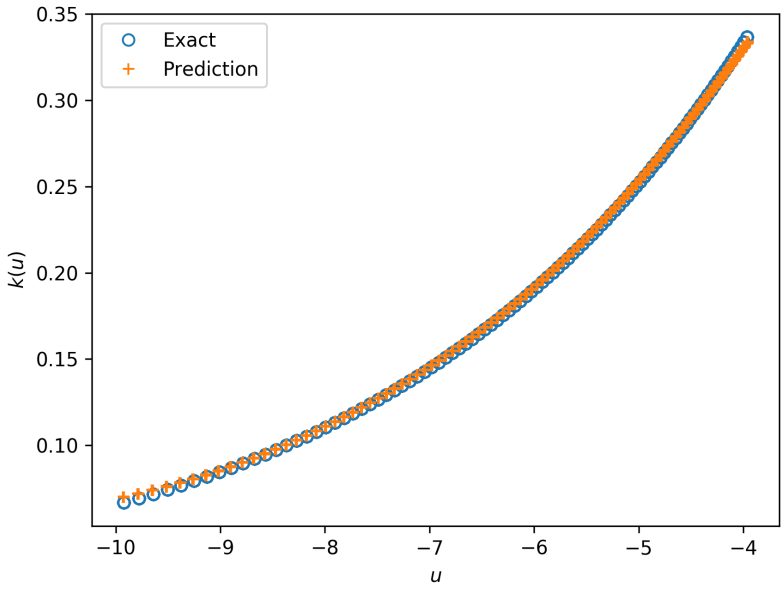


Figure 13: Comparison of the reference $K(u)$ given by the van Genuchten model and estimated $\hat{K}(u)$ in the non-linear diffusion (unsaturated flow) problem in the presence of measurement noise.

reference $K(u)$ function given by equations (29) and (30). It is evident that the PINN method provides an accurate estimate of unknown $K(u)$ without any direct measurements of K as a function of u.

Finally, we examine the robustness of the PINN method in the presence of observation noise. We use the exact same setup as before except we add 1% random noise to the values of observed u. Figures 12 and 13 show the difference between predicted and referenced $u(x)$ and $K(u)$, respectevely. The added noise increases maximum error in the reconstructed u from 0.002 to 0.012, but the accuracy of the reconstructed $K(u)$ practically does not change. Note that the relative L_2 prediction errors for the "noisy" case are quite small, including 7.4×10^{-4} for u and 6.4×10^{-3} for K. For comparison, the relative L_2 prediction errors for the noiseless case are 5.8×10^{-5} for u and 5.9×10^{-3} for K.

5 Conclusions

In this work, we have presented the PINN method for estimating parameters and unknown physics (constitutive relationships) in PDE models. The proposed method uses both PDEs and measurements to train DNNs to approximate unknown parameters, constitutive relationships, and states (the PDE solution). Physical knowledge increases the accuracy of DNN training with small data sets and affords the ability to train DNNs when no direct measurements of the functions of interest are available.

We have tested this method for estimating an unknown space-dependent diffusion coefficient in a linear diffusion equation and an unknown constitutive relationship in a non-linear diffusion equation. For the parameter estimation problem, we assume that partial measurements of the coefficient and state are available and have demonstrated that the proposed method is more accurate than the state-of-the-art MAP parameter estimation method. For the non-linear diffusion PDE model with an unknown constitutive relationship (state-dependent diffusion coefficient), the proposed method has been

shown to accurately estimate the non-linear diffusion coefficient without any measurements of the diffusion coefficient and with measurements of the state only. We have also demonstrated that adding physics constraints to DNN training could increase the accuracy of DNN parameter estimation by as much as 50%.

Parameter estimation is an ill-posed problem, and standard parameter estimation methods, including MAP, rely on regularization. In this work, we have trained DNNs for unknown parameters without regularizing the estimated parameter field or unknown function. In the absence of regularization, we have found that the estimates of the parameter and state depend on the DNN Xavier initialization scheme. For the considered problem, the uncertainty (standard deviation) and mean error decreased with an increasing number of parameter measurements. The coefficient of variation of the relative error (the ratio of the relative error standard deviation to the mean value) was found to be approximately 0.1. In future research, we will investigate the effect of regularization in the PINN method on parameter estimation accuracy.

Acknowledgments

This research was partially supported by the U.S. Department of Energy (DOE) Advanced Scientific Computing (ASCR) program and the Pacific Northwest National Laboratory (PNNL) Deep Learning for Scientific Discovery Agile Investment program. PNNL is operated by Battelle for the DOE under Contract DE-AC05-76RL01830. The data and codes used in this paper are available at https://doi.org/10.5281/zenodo.3547944.

References

Askham, T., & Kutz, J. N. (2018). Variable projection methods for an optimized dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems*, *17*(1), 380–416.

Barajas-Solano, D. A., Wohlberg, B. E., Vesselinov, V. V., & Tartakovsky, D. M. (2014). Linear functional minimization for inverse modeling. *Water Resour. Res.*, *51*, 4516-4531. doi: 10.1002/2014WR016179

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., & Siskind, J. M. (2015). Automatic differentiation in machine learning: a survey. Retrieved from `http://arxiv.org/abs/1502.05767` doi: 10.1016/j.advwatres.2018.01.009

Bear, J. (2013). *Dynamics of fluids in porous media*. Courier Corporation.

Berg, J., & Nyström, K. (2018). A unified deep artificial neural network approach to partial differential equations in complex geometries. *Neurocomputing*, *317*, 28–41. Retrieved from `https://doi.org/10.1016/j.neucom.2018.06.056` doi: 10.1016/j.neucom.2018.06.056

Brunton, S. L., Brunton, B. W., Proctor, J. L., Kaiser, E., & Kutz, J. N. (2017). Chaos as an intermittently forced linear system. *Nature communications*, *8*(1), 19.

Byrd, R. H., Lu, P., Nocedal, J., & Zhu, C. (1995). A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, *16*(5), 1190–1208.

Carrera, J., Alcolea, A., Medina, A., Hidalgo, J., & Slooten, L. J. (2005, Mar 01). Inverse problem in hydrogeology. *Hydrogeology Journal*, *13*(1), 206–222. Retrieved from `https://doi.org/10.1007/s10040-004-0404-7` doi: 10.1007/s10040-004-0404-7

Chen, S., & Billings, S. A. (1989). Representations of non-linear systems: the narmax model. *International Journal of Control*, *49*(3), 1013–1032.

Claesen, M., & De Moor, B. (2015). Hyperparameter search in machine learning. *arXiv preprint arXiv:1502.02127*.

Daliakopoulos, I. N., Coulibaly, P., & Tsanis, I. K. (2005). Groundwater level forecasting using artificial neural networks. *Journal of hydrology*, *309*(1-4), 229–

240.

Dalto, M., Matuško, J., & Vašak, M. (2015). Deep neural networks for ultra-short-term wind forecasting. In *2015 ieee international conference on industrial technology (icit)* (pp. 1657–1663).

Einstein, A. (1921). Geometrie und erfahrung. In *Geometrie und erfahrung: Erweiterte fassung des festvortrages gehalten an der preussischen akademie der wissenschaften zu berlin am 27. januar 1921* (pp. 2–20). Berlin, Heidelberg: Springer Berlin Heidelberg. Retrieved from `https://doi.org/10.1007/978-3-642-49903-6_1` doi: 10.1007/978-3-642-49903-6_1

Giannakis, D. (2017). Data-driven spectral decomposition and forecasting of ergodic dynamical systems. *Applied and Computational Harmonic Analysis*.

Giannakis, D., & Majda, A. J. (2012). Nonlinear laplacian spectral analysis for time series with intermittency and low-frequency variability. *Proceedings of the National Academy of Sciences*, *109*(7), 2222–2227.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (pp. 249–256).

Gonzalez-Garcia, R., Rico-Martinez, R., & Kevrekidis, I. (1998). Identification of distributed parameter systems: A neural net based approach. *Computers & chemical engineering*, *22*, S965–S968.

Kevrekidis, I. G., Gear, C. W., Hyman, J. M., Kevrekidid, P. G., Runborg, O., Theodoropoulos, C., et al. (2003). Equation-free, coarse-grained multiscale computation: Enabling mocroscopic simulators to perform system-level analysis. *Communications in Mathematical Sciences*, *1*(4), 715–762.

Le, Q. V., Ngiam, J., Coates, A., Lahiri, A., Prochnow, B., & Ng, A. Y. (2011). On optimization methods for deep learning. *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, 265–272.

Lieberman, C., Willcox, K., & Ghattas, O. (2010). Parameter and state model reduction for large-scale statistical inverse problems. *SIAM Journal on Scientific Computing*, *32*(5), 2523–2542.

Liu, F., Xu, F., & Yang, S. (2017). A flood forecasting model based on deep learning algorithm via integrating stacked autoencoders with bp neural network. In *2017 ieee third international conference on multimedia big data (bigmm)* (pp. 58–61).

Lusch, B., Kutz, J. N., & Brunton, S. L. (2018). Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications*, *9*(1), 4950. Retrieved from `https://doi.org/10.1038/s41467-018-07210-0` doi: 10.1038/s41467-018-07210-0

Mardt, A., Pasquali, L., Wu, H., & Noé, F. (2018). Vampnets for deep learning of molecular kinetics. *Nature communications*, *9*(1), 5.

McKay, M. D., Beckman, R. J., & Conover, W. J. (1979). Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, *21*(2), 239–245.

Mo, S., Zhu, Y., Zabaras, N., Shi, X., & Wu, J. (2019). Deep convolutional encoder-decoder networks for uncertainty quantification of dynamic multiphase flow in heterogeneous media. *Water Resources Research*, *55*(1), 703–728.

Raissi, M. (2018). Deep hidden physics models: Deep learning of nonlinear partial differential equations. *arXiv preprint arXiv:1801.06637*.

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2017a). Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*.

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2017b). Physics informed deep learning (part ii): Data-driven discovery of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10566*.

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (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. Retrieved from https://doi.org/10.1016/j.jcp.2018.10.045 doi: 10.1016/j.jcp.2018.10.045

Ramsundar, B., & Zadeh, R. B. (2018). *Tensorflow for deep learning: from linear regression to reinforcement learning.* " O'Reilly Media, Inc.".

Rudy, S., Alla, A., Brunton, S. L., & Kutz, J. N. (2018). Data-driven identification of parametric partial differential equations. *arXiv preprint arXiv:1806.00732*.

Schwarzer, M., Rogan, B., Ruan, Y., Song, Z., Lee, D. Y., Percus, A. G., ... others (2019). Learning to fail: Predicting fracture evolution in brittle material models using recurrent graph convolutional neural networks. *Computational Materials Science*, *162*, 322–332.

Stuart, A. M. (2010). Inverse problems: a bayesian perspective. *Acta Numerica*, *19*, 451–559.

Van Genuchten, M. T. (1980). A closed-form equation for predicting the hydraulic conductivity of unsaturated soils 1. *Soil science society of America journal*, *44*(5), 892–898.

Wehmeyer, C., & Noé, F. (2018). Time-lagged autoencoders: Deep learning of slow collective variables for molecular kinetics. *The Journal of Chemical Physics*, *148*(24), 241703.

Whitaker, S. (2013). *The method of volume averaging* (Vol. 13). Springer Science & Business Media.

White, M., Oostrom, M., & Lenhard, R. (1995). Modeling fluid flow and transport in variably saturated porous media with the stomp simulator. 1. nonvolatile three-phase model description. *Advances in Water Resources*, *18*(6), 353–364.

Williams, M. O., Kevrekidis, I. G., & Rowley, C. W. (2015). A data–driven approximation of the koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, *25*(6), 1307–1346.

Yang, L., Treichler, S., Kurth, T., Fischer, K., Barajas-Solano, D., Romero, J., ... others (2019). Highly-ccalable, physics-informed gans for learning solutions of stochastic PDEs. In *2019 ieee/acm third workshop on deep learning on supercomputers (DLS)* (pp. 1–11).

Yang, Y., & Perdikaris, P. (2019). Adversarial uncertainty quantification in physics-informed neural networks. *Journal of Computational Physics*, *394*, 136–152.

Yeung, E., Kundu, S., & Hodas, N. (2017). Learning deep neural network representations for koopman operators of nonlinear dynamical systems. *arXiv preprint arXiv:1708.06850*.

Zhu, Y., & Zabaras, N. (2018). Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification. *Journal of Computational Physics*, *366*, 415–447.

Zhu, Y., Zabaras, N., Koutsourelakis, P.-S., & 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.

Figure 1.

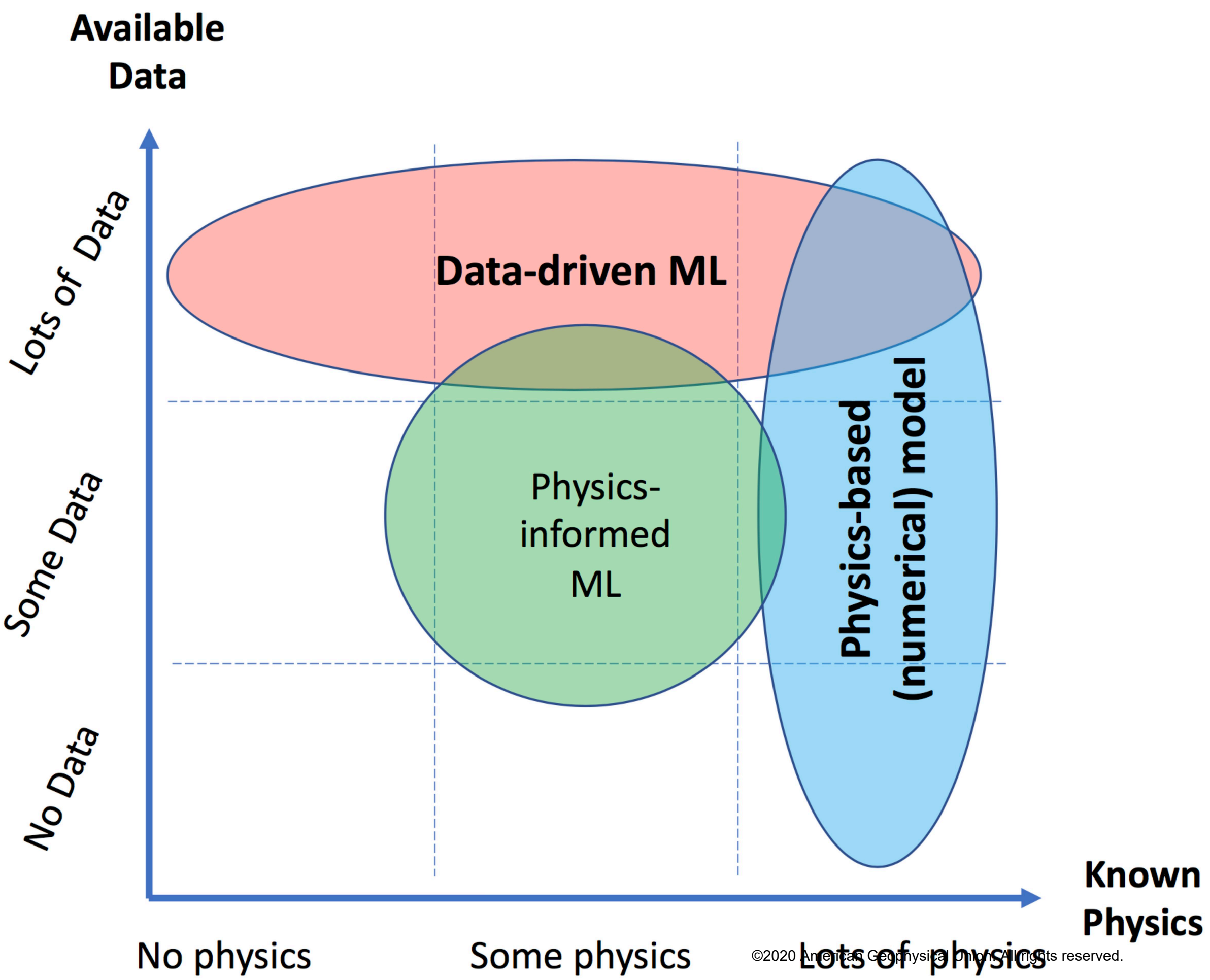
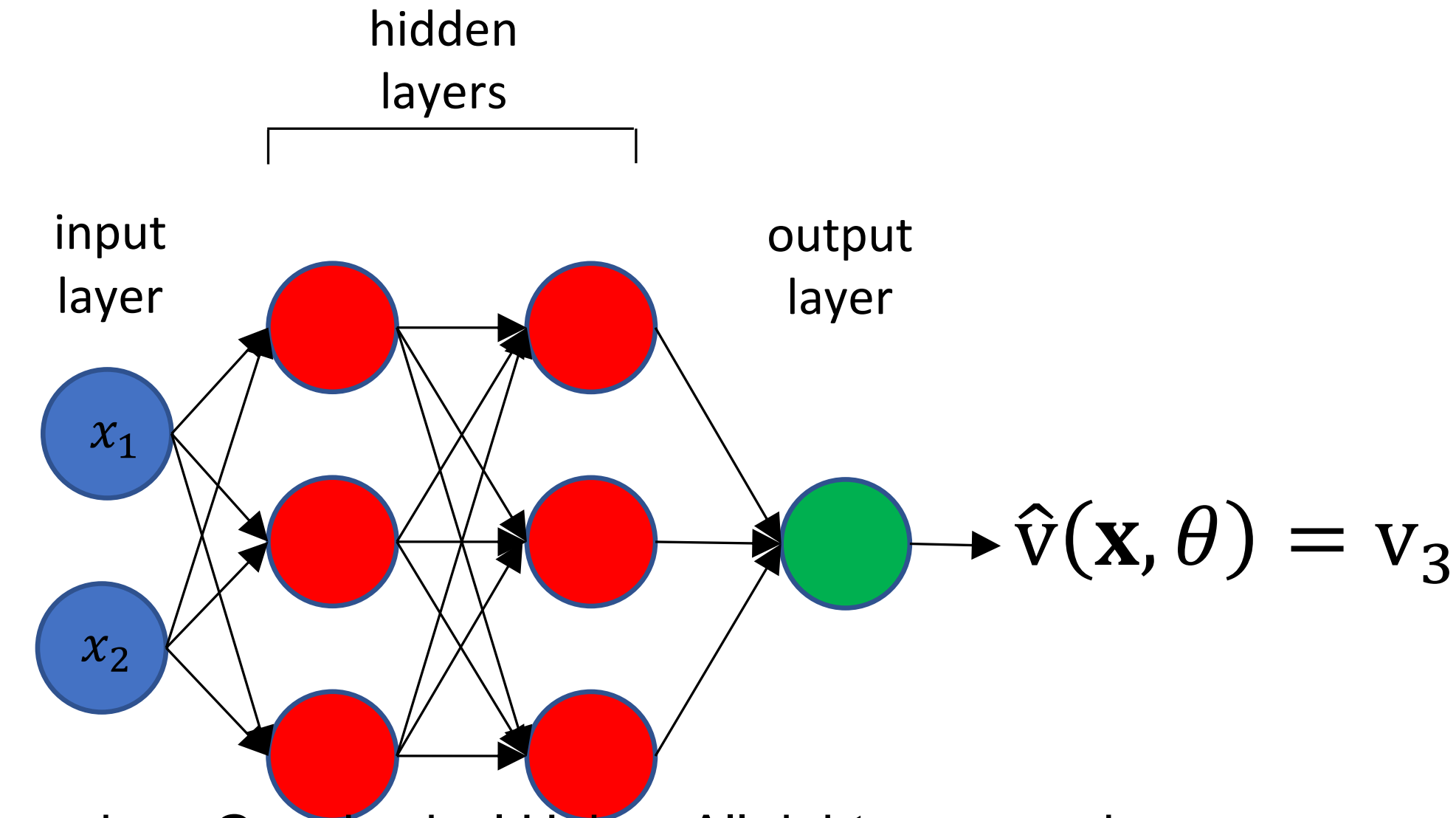
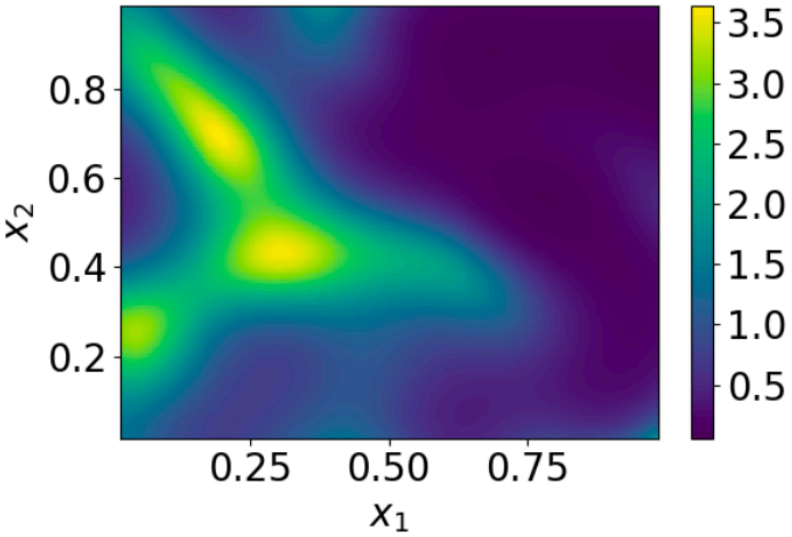


Figure 2.

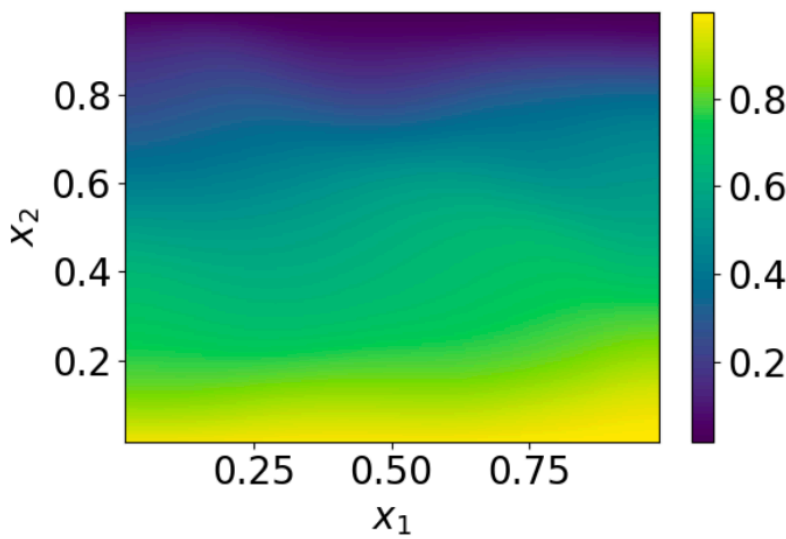


$\mathbf{v}_1$ $\mathbf{v}_2$

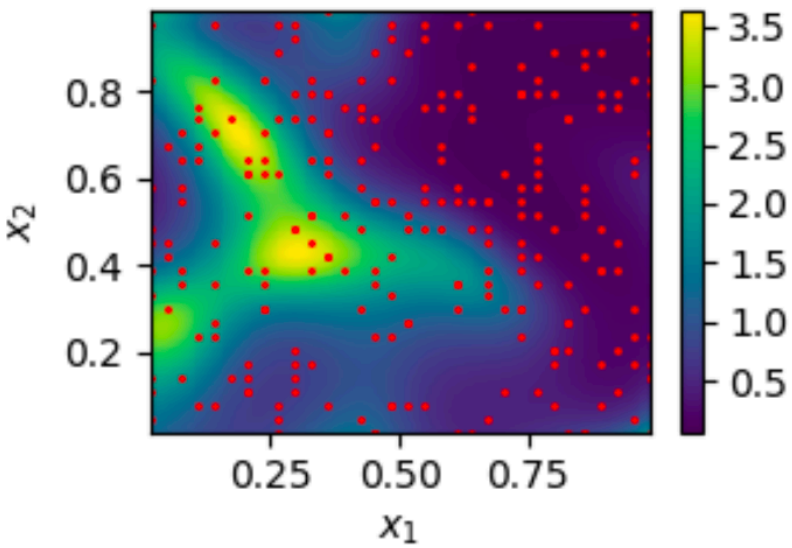
Figure 3.



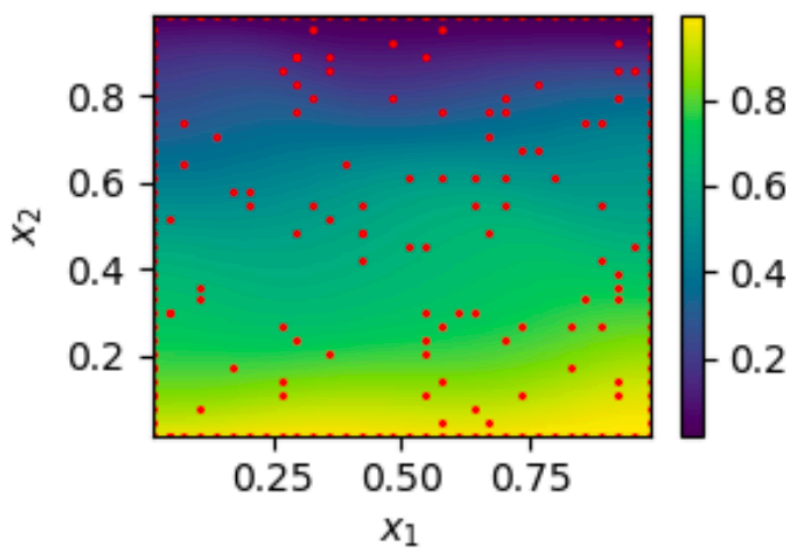
(a) Reference $K(\mathbf{x})$



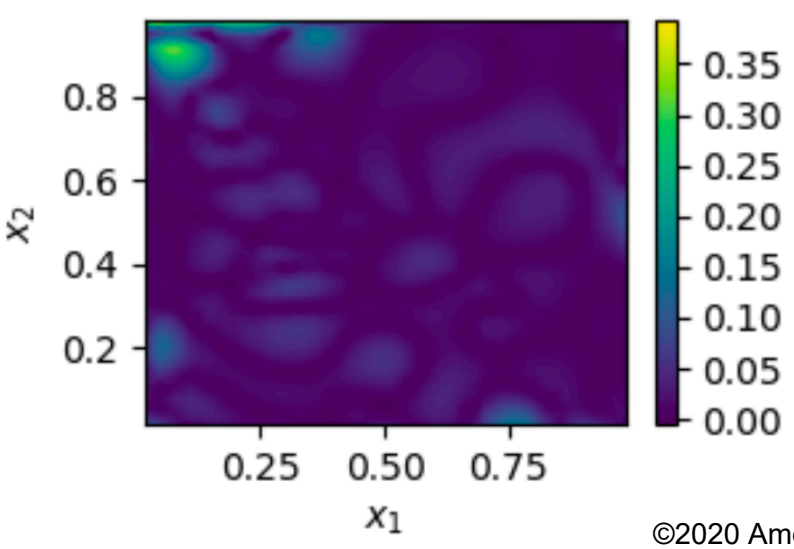
(b) Reference $u(\mathbf{x})$



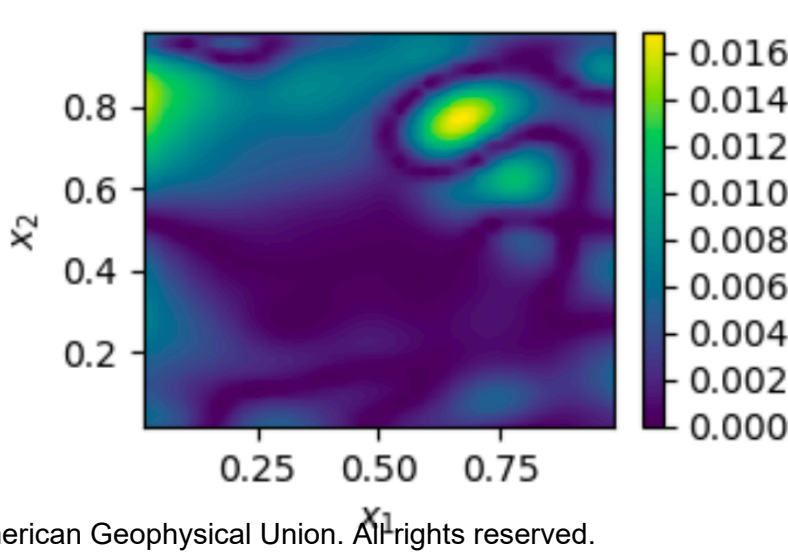
(c) $\hat{K}(\mathbf{x})$



(d) $\hat{u}(\mathbf{x})$

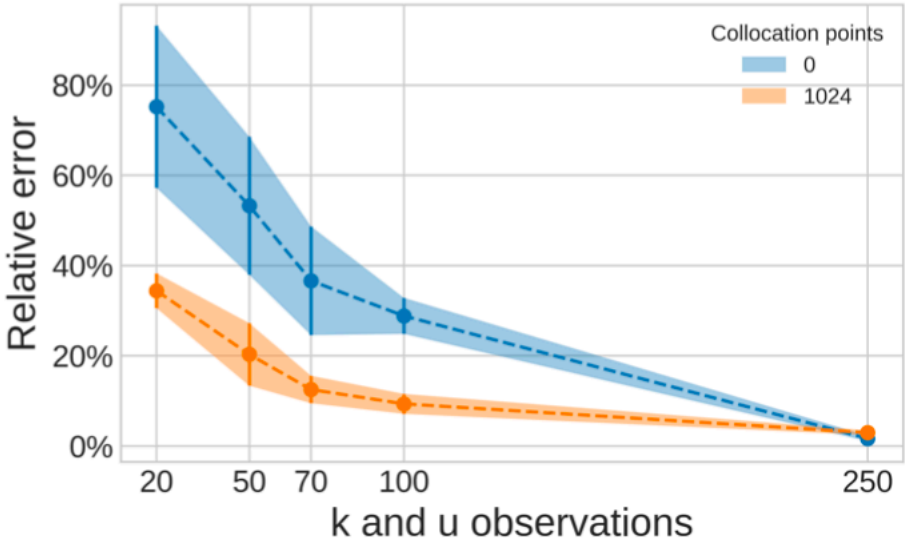


(e) $|K(\mathbf{x}) - \hat{K}(\mathbf{x})|$



(f) $|u(\mathbf{x}) - \hat{u}(\mathbf{x})|$

Figure 4.



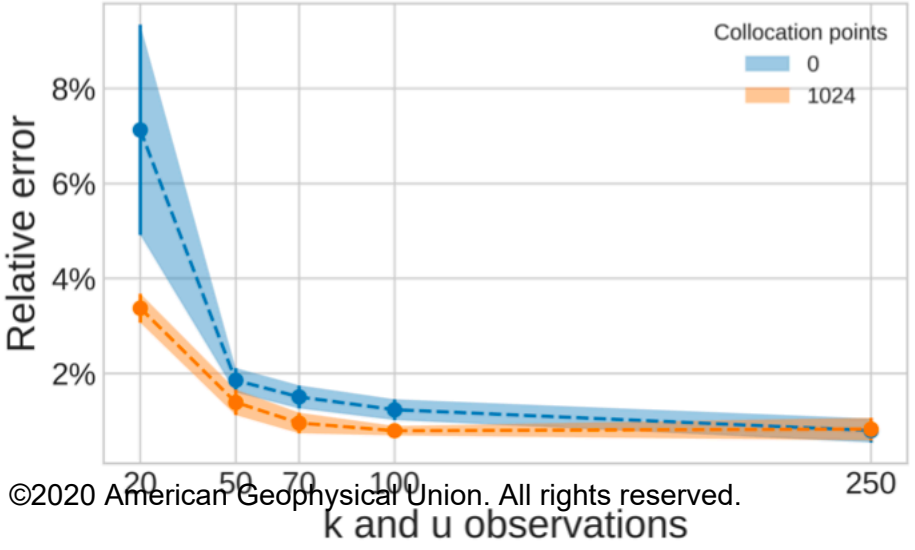
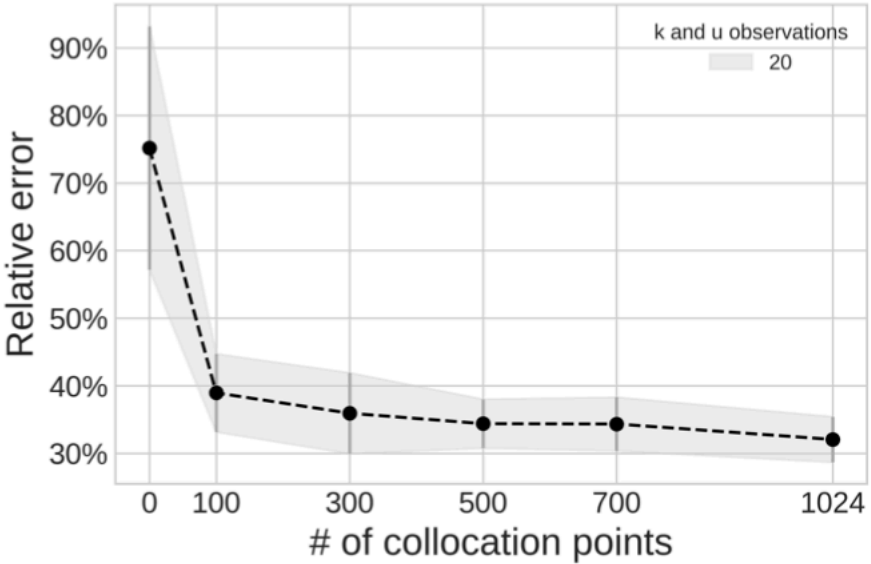
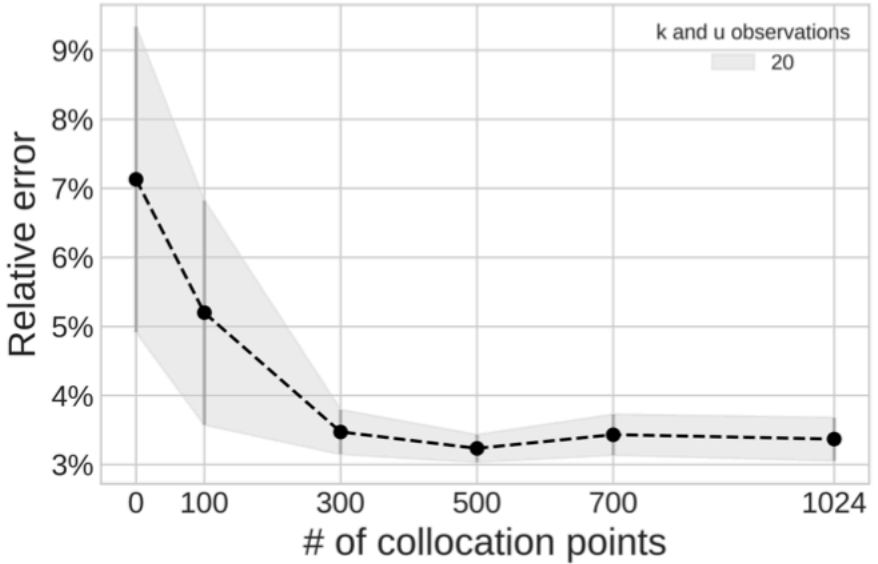


Figure 5.

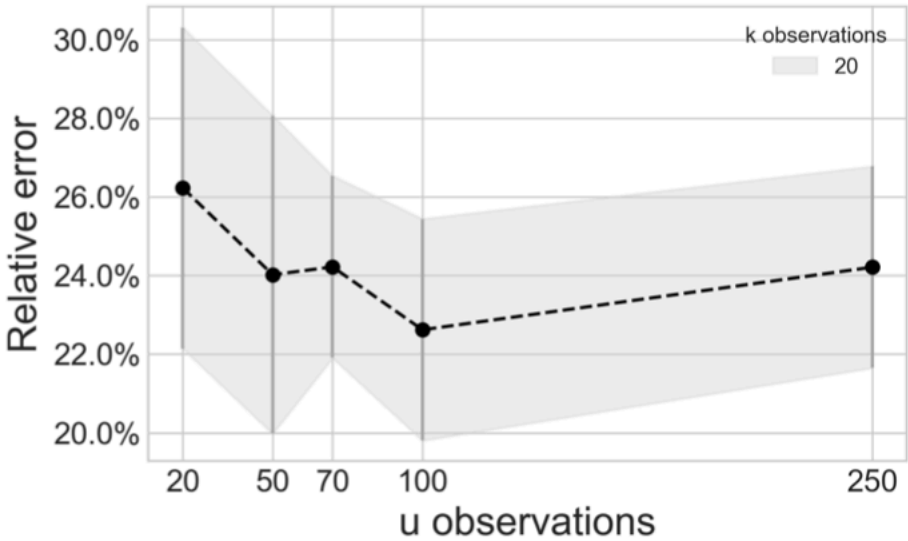


(a) $\bar{\varepsilon}_K$ and σ_{ε_K}

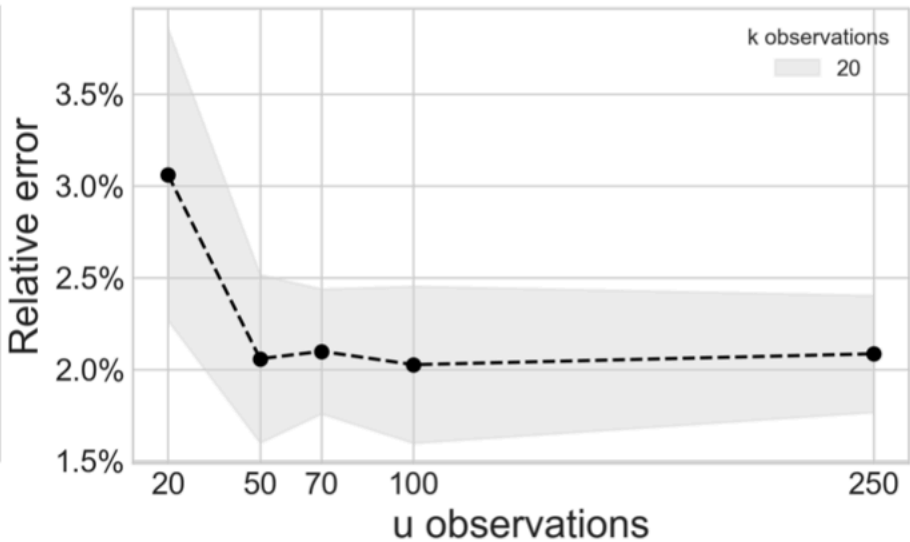


(b) $\bar{\varepsilon}_u$ and σ_{ε_u}

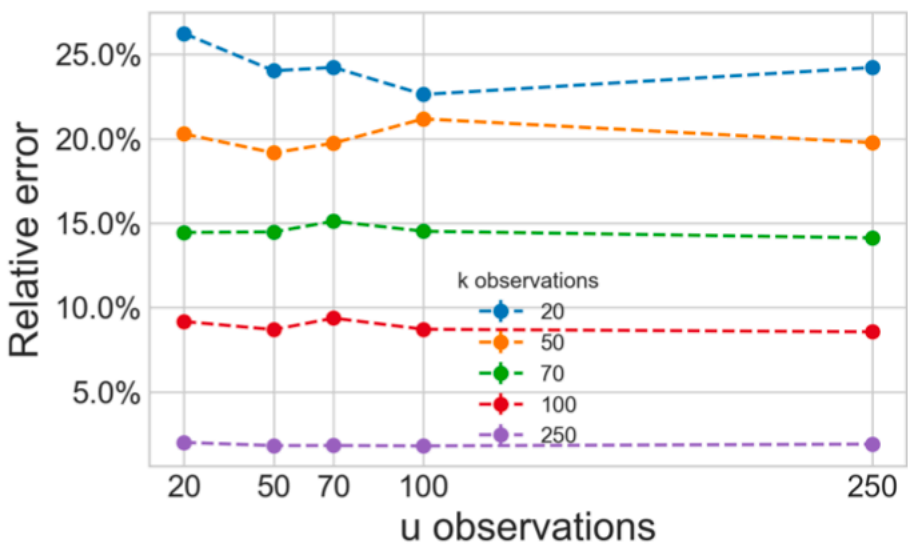
Figure 6.



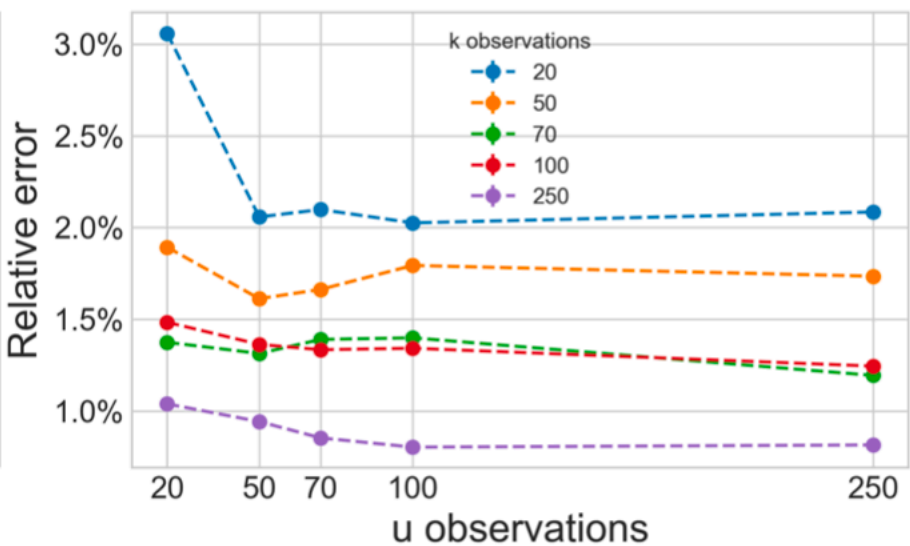
(a) $\overline{\varepsilon}_K$ and σ_{ε_K}



(b) $\overline{\varepsilon}_u$ and σ_{ε_u}

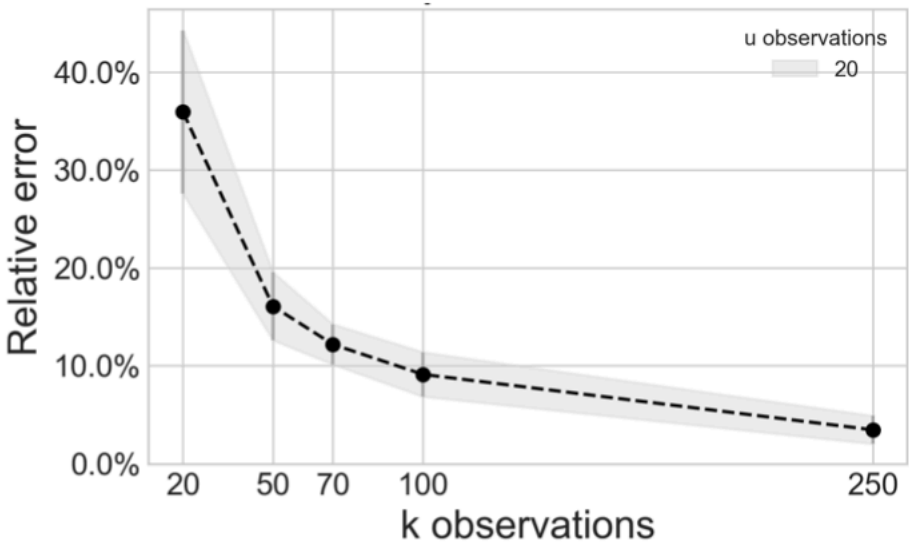


(c) $\overline{\varepsilon}_K$

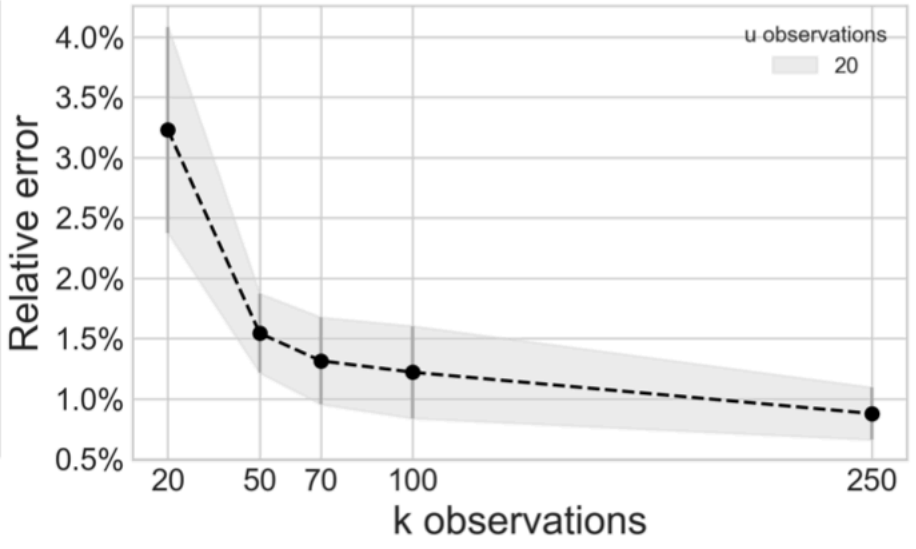


(d) $\overline{\varepsilon}_u$

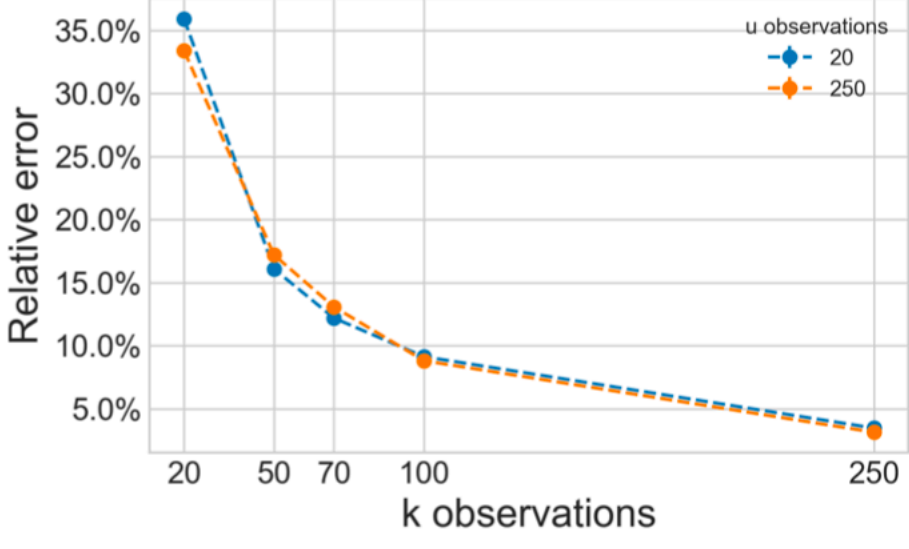
Figure 7.



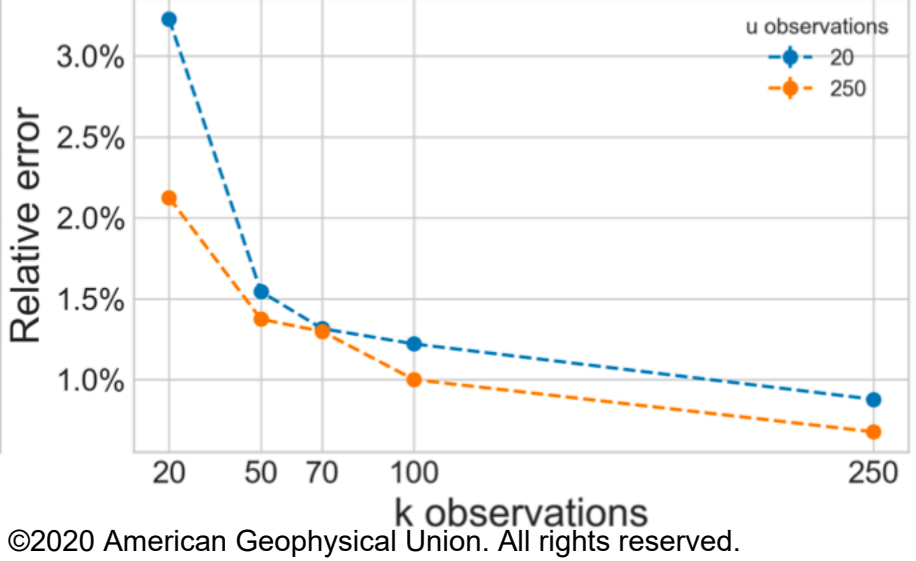
(a) $\overline{\varepsilon}_K$ and σ_{ε_K}



(b) $\overline{\varepsilon}_u$ and σ_{ε_u}



(c) $\overline{\varepsilon}_K$



(d) $\overline{\varepsilon}_u$

Figure 8.

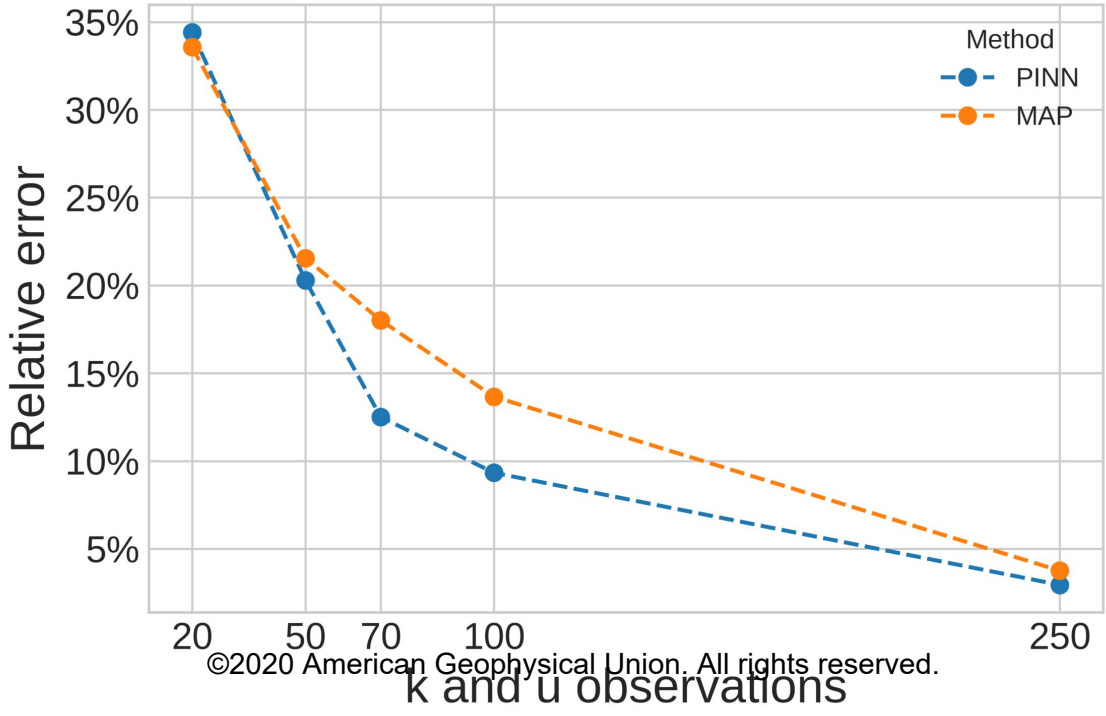


Figure 9.

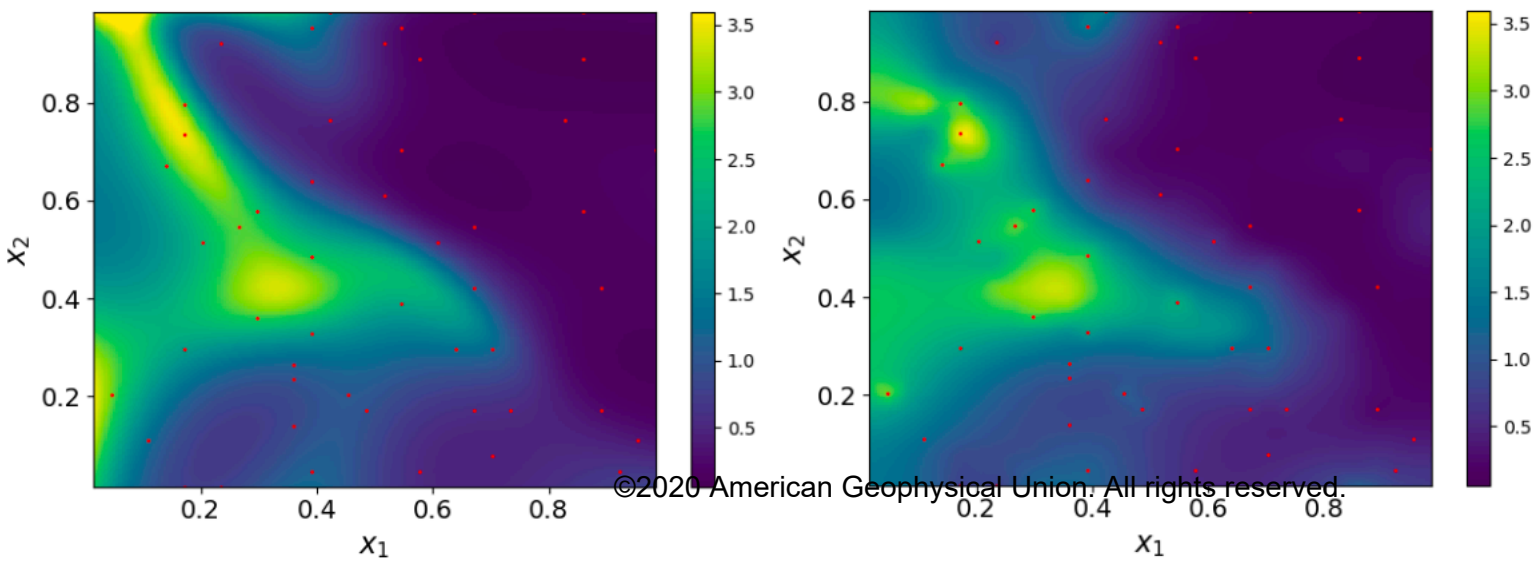


Figure 10.

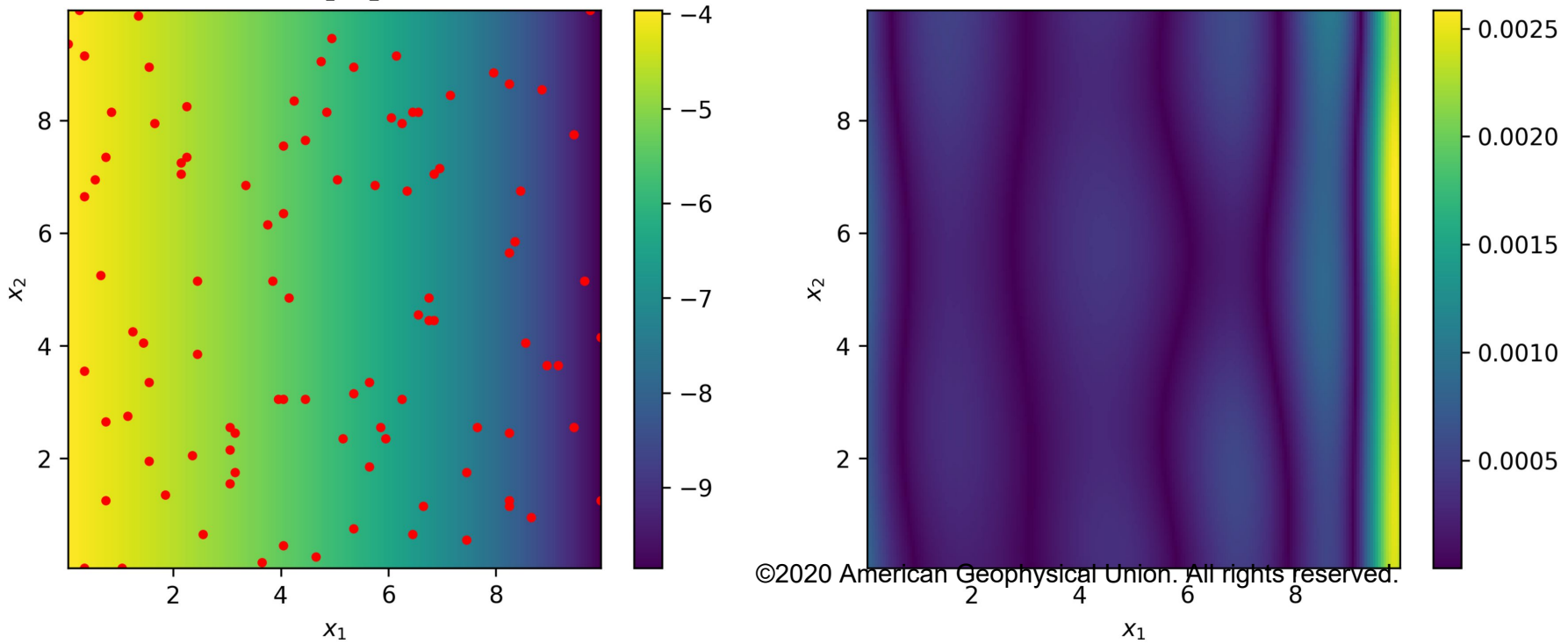


Figure 11.

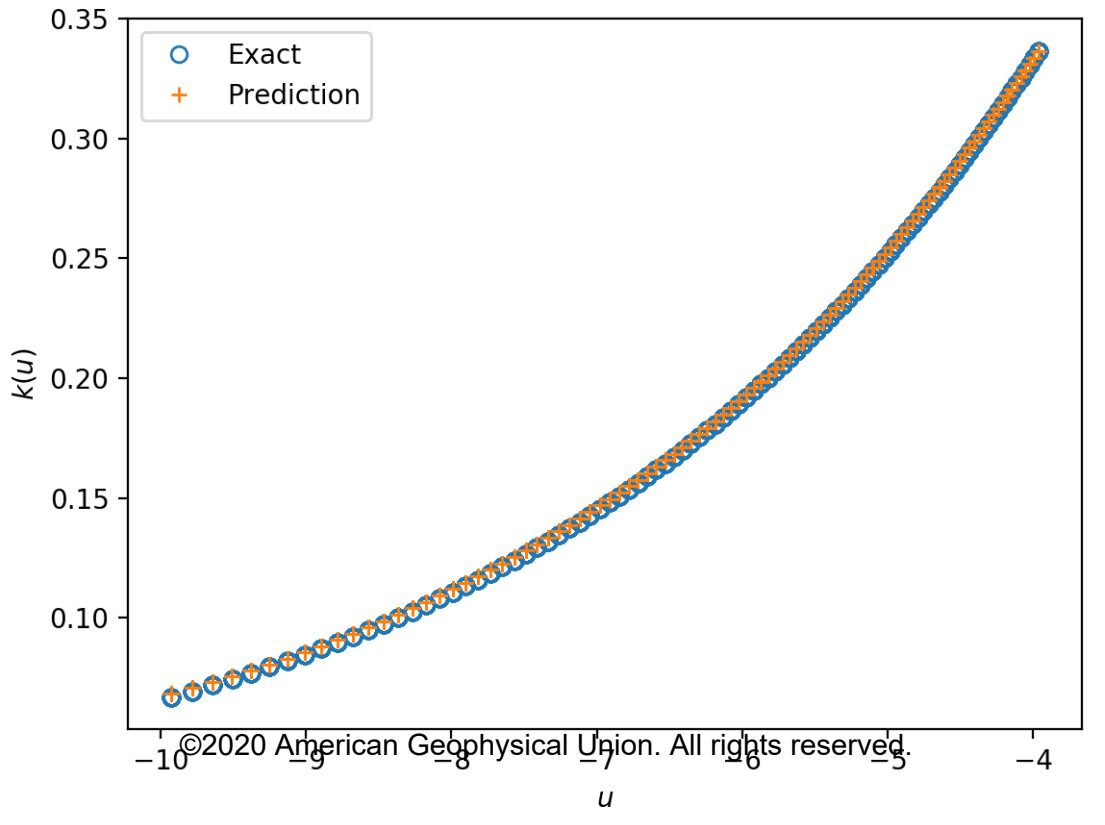


Figure 12.

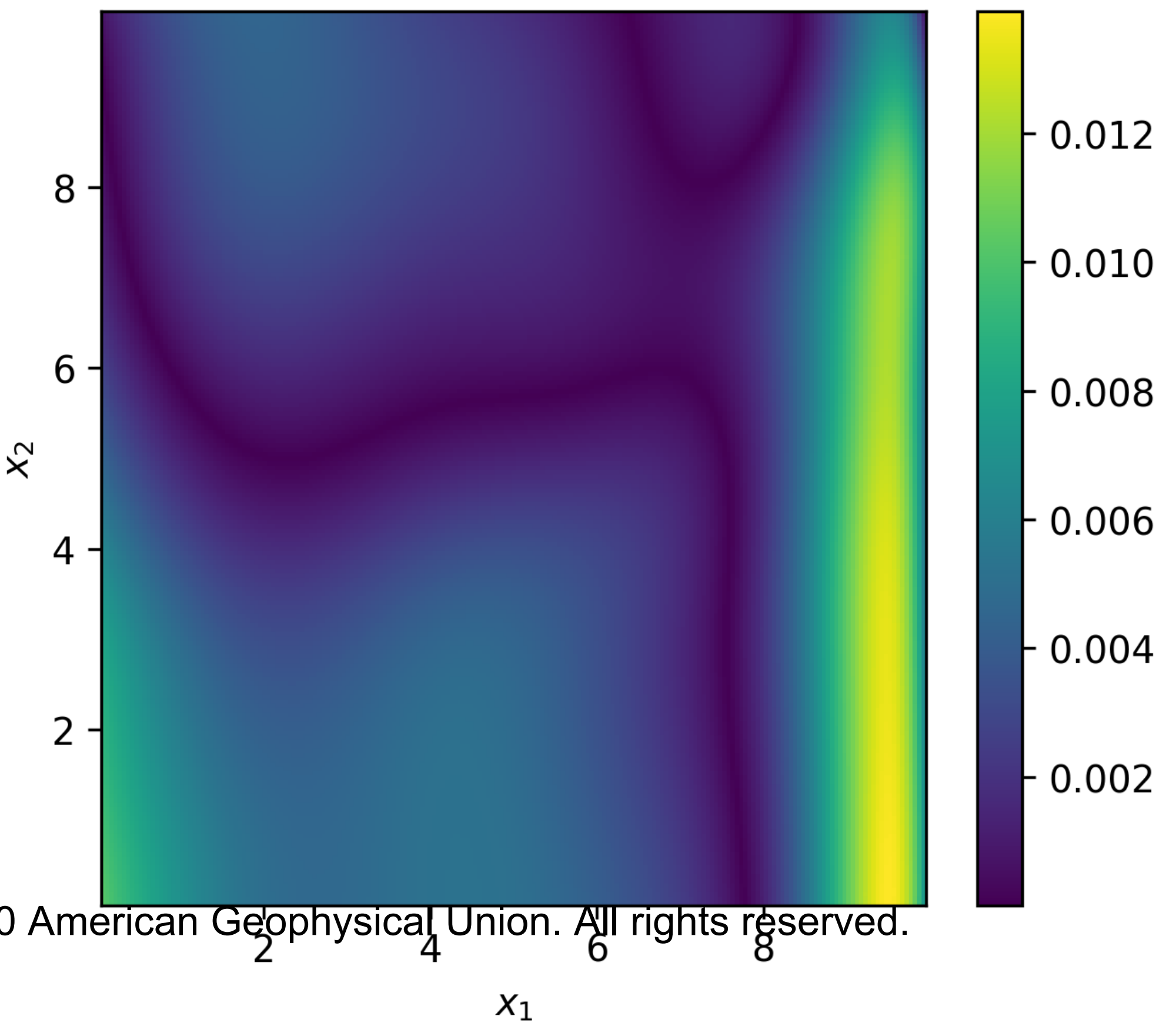


Figure 13.