DATA-INFORMED EMULATORS FOR MULTI-PHYSICS SIMULATIONS

Hannah Lu,[1] Dinara Ermakova,[2] Haruko Murakami Wainwright,[2,3] Liange Zheng,[3] & Daniel M. Tartakovsky[1,*]

[1]*Department of Energy Resources Engineering, Stanford University, Stanford, CA 94305, USA*

[2]*Department of Nuclear Engineering, University of California Berkeley, Berkeley, CA 94720, USA*

[3]*Earth and Environmental Sciences Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA*

*Address all correspondence to: Daniel M. Tartakovsky, Department of Energy Resources Engineering, Stanford University, Stanford, CA 94305, USA; Tel.: +1 650 498 8791; Fax: +1 650 725 2099, E-mail: tartakovsky@stanford.edu

Original Manuscript Submitted: 4/1/2021; Final Draft Received: 8/19/2021

Machine learning techniques are powerful tools for construction of emulators for complex systems. We explore different machine learning methods and conceptual methodologies, ranging from functional approximations to dynamical approximations, to build such emulators for coupled thermal, hydrological, mechanical, and chemical processes that occur near an engineered barrier system in the nuclear waste repository. Two nonlinear approximators, random forests and neural networks, are deployed to capture the complexity of the physics-based model and to identify its most significant hydrological and geochemical parameters. Our emulators capture the temporal evolution of the uranium distribution coefficient of the clay buffer and identify its functional dependence on these key parameters. The emulators' accuracy is further enhanced by assimilating relevant simulated predictors and clustering strategy. The relative performance of random forests and neural networks shows the advantage of ensemble learning in the random forests algorithm, especially for highly nonlinear problems with limited data.

KEY WORDS: *random forest, neural network, clustering, distribution coefficient*

1. INTRODUCTION

Predictive capabilities are critical for sustainable utilization of subsurface environment, improving the access to energy resources, and developing cost-effective and environmentally safe operations. Examples of such predictive modeling include nuclear waste disposal (Bea et al., 2013), geological CO_2 sequestration (Audigane et al., 2007), geothermal reservoirs (Xiong et al., 2013), and subsurface contamination and remediation (Steefel et al., 2015). A recent focus has been to incorporate multiscale and multiphysics models, which typically comprise a large number of coupled (nonlinear) ordinary and partial differential equations. A representative example

is thermal-hydrological-mechanical-chemistry (THMC) models (Rutqvist et al., 2014; Steefel et al., 2015, 2005; Zheng et al., 2017) used to represent, e.g., the changes in flow characteristics due to subsurface evolution caused by thermal and chemical processes. Despite continuing advances in software and hardware development, including high-performance computing, multiphysics simulations with large degrees of freedom remain a demanding and elusive task. That is especially so in sensitivity analysis, uncertainty quantification, and inverse modeling, where many simulation runs are required.

Model reduction techniques can significantly reduce the (prohibitively) high computational cost of physics-based simulations, while capturing key features of the underlying dynamics. Such techniques have been used extensively in subsurface applications (Barthelemy and Haftka, 1993; Forrester and Keane, 2009; Lucia et al., 2004; Razavi et al., 2012; Saridakis and Dentsoras, 2008; Schmit Jr. and Farshi, 1974; Simpson et al., 2001) and can be grouped in two general classes. The first is physics-based reduced-order models (ROMs), which seek to map a high-dimensional model onto a meaningful representation of reduced dimensionality; in this context, dimensionality refers to the number of degrees of freedom in a discretized numerical model. A prime example of this class is proper orthogonal decomposition (POD) (Kerschen et al., 2005; Rowley, 2005), which is grounded in singular value decomposition (SVD). It obtains a ROM by projecting the dynamics of the full model onto the hyperplane using the basis extracted from the SVD analysis. The computational saving stems from replacing the high-dimensional full nonlinear system with its lower-dimensional counterpart for future prediction. Such ROMs are physics based in the sense that they inherit the dynamic operator from the projection.

The second class of ROMs are emulators or surrogates. Instead of reducing a model's dimensionality, these methods aim to reduce its complexity by learning the dynamics of the state variables or quantities of interest directly from the full model's output and/or observational data. These data-informed and equation-free ROMs are built by using such machine learning techniques as Gaussian process regression (Pau et al., 2013; Rasmussen, 2003), dynamic mode decomposition (DMD) (Kutz et al., 2016; Schmid, 2010), random forest (RF) (Booker and Woods, 2014; Naghibi et al., 2016), and neural networks (NN) (Hesthaven and Ubbiali, 2018; Qin et al., 2019).

Construction of both types of ROMs for multiphysics (e.g., THMC) problems faces several challenges. First, once discretized in space, complex multiphysics problems result in huge systems of nonlinear ordinary differential equations for which the projection-based techniques become unfeasible. Although POD can be combined with the empirical interpolation method (Chaturantabut and Sorensen, 2010; Maday and Mula, 2013) in order to handle nonlinearities, it still requires one to solve for a large number of state variables from the projected system. Second, the time evolution of a quantity of interest, expressed as a function of the parameters and simulated predictors, is usually highly nonlinear. It becomes challenging for conventional approximators like Gaussian Process and polynomial regression to capture the dynamics precisely. Third, the computational cost of the full model is so high that only a limited amount of high-fidelity data is available for training, which poses a great challenge of overfitting. For these and other reasons, surrogate models for complex coupled processes (Bianchi et al., 2016) are scarce.

Driven by the practical considerations mentioned above, we focus on the data-informed/equation-free emulators. Specifically, we investigate the performance of RF- and NN-based emulators for complex multiphysics problems. These two surrogates aim to directly predict the dynamics of quantities of interest without having to deal with the full set of state variables and the governing nonlinear equations. Among the many machine learning tools, we choose RF and NN because

they are known as robust universal nonlinear approximators that place no formal constraints on data. Conventional regression methods, such as polynomial regression and Gaussian process regression, are expected to fail for complex multiphysics problems because the correlation and/or smoothness conditions they place on the data are seldom satisfied.

Recent theoretical and computational developments in machine learning (e.g., regularization, cross-validation, and bootstrap aggregating) enhance the generalizability of RF and NN, enabling them to handle "small data." Since nonlinear systems often exhibit dramatic changes in the relationship between parameters and target variables, we utilize clustering tools to identify the threshold behaviors and design more efficient training strategies. Finally, to boost the computational efficiency of the emulator training, we deploy the machine learning toolboxes keras and scikit-learn.

We use the thermal-hydrological-chemistry (THC) model of an engineered-barrier system at a hypothetical nuclear waste disposal site (Ermakova et al., 2020) to illustrate the performance of our RF and NN surrogates. While the model, which consists of a large number of coupled partial differential equations, describes the spatiotemporal evolution of multiple physicochemical state variables, a quantity of interest is the uranium distribution coefficient K_{d} for the buffer material. The novelty of our study is two-fold. From the methodological prospective, we improve the prediction accuracy of our emulators by splitting the training based on the geochemical features. The thresholds of the geochemical features are identified by a cluster analysis on training data. Afterward, we train each cluster to handle highly nonlinear and nonmonotonic functions $K_{\mathrm{d}} = K_{\mathrm{d}}(t)$ by taking advantage of the RF and NN approximators. From the applications perspective, although RF and NN have been used before as emulators of relatively simple subsurface models (Booker and Woods, 2014; Naghibi et al., 2016; Zhou and Tartakovsky, 2020), we are not aware of their use for such complex multiphysics phenomena as THC.

In Section 2, we provide a brief description of the THC model (Ermakova et al., 2020) and identify a relevant quantity of interest. In Section 3, we detail the general methodology for construction of the RF and NN emulators. The accuracy and robustness of these two surrogates in the THC context are investigated in Section 4. Main conclusions drawn from this study are summarized in Section 5.

2. PROBLEM FORMULATION

We consider multiphysics simulations of a phenomenon that is described by N'_{sv} state variables $\mathbf{s}(\mathbf{x},t) = \{s_1,\dots,s_{N'_{\mathrm{sv}}}\}$, varying in space $\mathbf{x} \in \mathcal{D}$ and time $t \in [0,T]$ throughout the simulation domain $\mathcal{D}$ during the simulation time interval $[0,T]$. The spatiotemporal evolution of these state variables is described by a system of coupled partial differential equations

$$\frac{\partial s_i}{\partial t} = \mathcal{N}_i(\mathbf{s};\mathbf{p}), \quad (\mathbf{x},t) \in \mathcal{D}\times(0,T]; \quad i = 1,\dots,N'_{\mathrm{sv}}, \tag{1}$$

where $\mathcal{N}_i$ are (nonlinear) differential operators that contain spatial derivatives, and $\mathbf{p} = \{p_1,\dots,p_{N_{\mathrm{par}}}\}$ is a set of N_{par} parameters that might vary within space and time $(\mathbf{x},t)$ and be dependent on $\mathbf{s}$. Problems of this kind have to be solved numerically, which requires a discretization of the spatial domain $\mathcal{D}$ into N_{el} elements (or nodes) and the simulation time horizon $[0,T]$ into N_{st} time steps. Consequently, the numerical solution of Eq. (1) gives a set of $N_{\mathrm{sv}} = N'_{\mathrm{sv}} \times N_{\mathrm{el}}$ discretized state variables, $\mathbf{s}_{kn} = \mathbf{s}(\mathbf{x}_k,t_n) = [s_1(\mathbf{x}_k,t_n),\cdots,s_{N'_{\mathrm{sv}}}(\mathbf{x}_k,t_n)]^\top$ with $k = 1,\dots,N_{\mathrm{el}}$ and $n = 1,\dots,N_{\mathrm{st}}$.

More often than not, this model output has to be postprocessed to compute N_Q quantities of interest (QoIs) $\mathbf{Q} = \{Q_1, \ldots, Q_{N_Q}\}$, such that $Q_i = \mathcal{M}_i(\mathbf{s}_{kn})$ for $i = 1, \ldots, N_Q$, $k = 1, \ldots, N_{\text{el}}$, and $n = 1, \ldots, N_{\text{st}}$. The maps $\mathcal{M}_i$ can represent, e.g., numerical approximations of the integrals over $\mathcal{D}$ or a streamline. In any simulation of practical significance, $N_Q \ll N_{\text{sv}}$ which makes QoIs much easier to visualize and comprehend than the raw output $\mathbf{s}_{kn}$. Surrogate modeling aims to derive relationships $\mathbf{Q} = \mathbf{Q}(\mathbf{p}; \mathbf{x}, t)$ directly, bypassing the need to compute $\mathbf{s}(\mathbf{x}, t)$ first.

To make the exposition concrete, we ground our analysis in the THC model for reactive transport of uranium (U) within an engineered-barrier system at a hypothetical nuclear waste disposal site (Ermakova et al., 2020). A key component of the system is a clay-based buffer surrounding waste canisters, since clay has high sorption capacity for many radionuclides. The clay properties change over time partly due to thermal and hydrological processes, which may reduce the sorption capacity or degrade the barrier function. The model consists of $N'_{\text{sv}} = 22$ partial differential equations for the N'_{sv} state variables $\mathbf{s}(\mathbf{x}, t)$, representing fluid pressure, saturation, temperature, the concentrations of primary species associated with uranium surface complexation, and the concentrations of around 80 geochemical complexes. In the TOUGHREACT simulations (Xu et al., 2014) of this problem, the one-dimensional domain $\mathcal{D}$ is discretized into $N_{\text{el}} = 206$ elements, and the simulation time horizon $T = 10^3$–10^5 years into $N_{\text{st}} = 7444$ time steps. Although the THC model can be developed for a single canister, it is not possible to extend this full model for the entire repository.

The single QoI from this computation Q_1 is the average distribution coefficient K_{d} of U across the buffer, defined as

$$K_{\text{d}} = \frac{(\text{total mass of U sorbed})}{(\text{total mass of U in solution})}. \tag{2}$$

This QoI is to be used in a site-scale assessment model. Our goal is to build a surrogate,

$$K_{\text{d}} = f(t; \mathbf{p}), \quad t \in [0, T], \quad \mathbf{p} \in \Gamma \subset \mathbb{R}^{N_{\text{par}}}, \tag{3}$$

i.e., to "learn" the functional form of $f(\cdot;\cdot)$, from N_{MC} solutions of Eq. (1) obtained for N_{MC} different combinations of the parameters $\mathbf{p}$. The training data are obtained by postprocessing the mth model run ($m = 1, \ldots, N_{\text{MC}}$) to evaluate temporal snapshots (at times $t_1, \ldots, t_{N_{\text{st}}}$) of the corresponding realization of the distribution $K_{\text{d}}^{(m)}(t)$ from the mth parameter sample $\mathbf{p}^{(m)}$. Although the THC model [Eq. (1)] contains over two hundred parameters, the model predictions are relatively insensitive to all but seven of them (Ermakova et al., 2020). Therefore, with a slight abuse of notation, we set $N_{\text{par}} = 7$ in Eq. (3) while keeping the rest of the model parameters fixed. The seven input parameters used in our examples are collated in Table 1. The task of learning the function $f(t;\cdot)$ is complicated by the high degree of nonlinearity of $K_{\text{d}}^{(m)}(t)$ and by the high sensitivity of $K_{\text{d}}^{(m)}(t)$ to the inputs $\mathbf{p}$, i.e., by its variability from one value of m to another (Fig. 1).

The possible time dependence of the distribution coefficient K_{d} stems from its definition as a map $\mathcal{M}$ of some of the state variables from the set $\mathbf{s}(\mathbf{x}, t)$ or their simulated predictors $\boldsymbol{\gamma}(\mathbf{x}, t) = \{\gamma_1, \gamma_2, \ldots\}$. In the THC model, these simulated predictors are the pore water composition expressed in term of its pH, $\gamma_1(t) \equiv \text{pH}(t)$, and calcium ion concentration, $\gamma_2(t) \equiv [\text{Ca}^{2+}](t)$, both averaged over the space domain $\mathcal{D}$. This observation might lead one to attempt to construct an emulator in the form $K_{\text{d}} = g(\boldsymbol{\gamma}(t); \mathbf{p})$. This formulation is useful when the overall performance model can simulate the regional groundwater chemistry (such as pH

TABLE 1: Hydrological and geochemical properties of the engineered-barrier system, based on Ermakova et al. (2020)

p	Parameter ID	Physical meaning	Reference value	Min value	Max value
p_1	ilsoh	Adsorption surface area on illite (cm^2/g) in $\log_{10}$ scale—combined parameter for: illite strong site adsorption zone surface area—high sorption affinity; illite weak site adsorption zone surface area—lower sorption affinity	5	3	6
p_2	smsoh	Adsorption surface area on smectite (cm^2/g) in $\log_{10}$ scale—combined parameter for: smectite strong site adsorption zone surface area—high sorption affinity; smectite weak site adsorption zone surface area—lower sorption affinity	5	3	6
p_3	pH	Initial pore water pH	7.96	9	7
p_4	Ca^{2+}	Initial Ca^{2+} concentration in $\log_{10}$ scale in the aqueous phase	-1.66	-3	-1
p_5	smectite	Volume fraction of smectite	0.92	0.3	0.95
p_6	illite	Volume fraction of illite	0.0001	0.01	0.2
p_7	calcite	Volume fraction of calcite	0.01	0.01	0.03

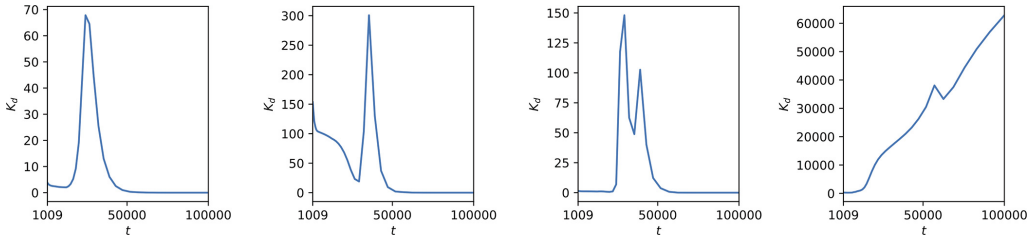

FIG. 1: Temporal variability of the distribution coefficient $K_{\mathrm{d}}^{(m)}$ for four combinations of the input parameters $\mathbf{p}$ indexed by m in equation Eq. (7). These realizations of K_{d} are evaluated by postprocessing the output of the THC model, based on Ermakova et al. (2020).

and Ca^{2+}) but not the uranium geochemistry. The observed behavior of $K_{\mathrm{d}}^{(m)}(t)$ and $\boldsymbol{\gamma}^{(m)}(t)$ falsifies this hypothesis in all realizations m of the input parameters (Fig. 2): one set of values of γ_1 and γ_2 can correspond to two different K_{d} values, failing the vertical line tests. This suggests that a surrogate aiming to incorporate the simulated predictors $\boldsymbol{\gamma}(t)$ must include an explicit dependence on time t,

$$K_{\mathrm{d}} = g(t, \boldsymbol{\gamma}(t); \mathbf{p}). \tag{4}$$

Another hypothesis is that the present state of K_{d} depends not on the present time t but on the whole history of its evolution up to that time. This possible temporal nonlocality of K_{d} can be captured by surrogates (see Appendix A for details):

$$\frac{\mathrm{d}K_{\mathrm{d}}}{\mathrm{d}t} = \mathcal{F}(K_{\mathrm{d}}, t; \mathbf{p}), \quad K_{\mathrm{d}}(t = 0; \mathbf{p}) = K_{\mathrm{d}}^0(\mathbf{p}), \tag{5}$$

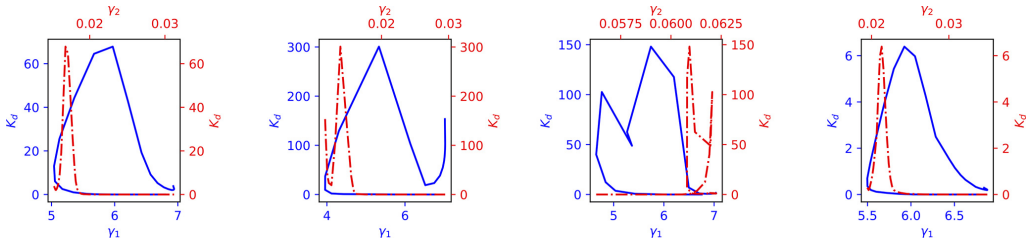

FIG. 2: Values of $K_{\mathrm{d}}^{(m)}(t)$ vs. $\gamma_1^{(m)}(t) \equiv \mathrm{pH}(t)$, and of $K_{\mathrm{d}}(t)$ vs. $\gamma_2^{(m)}(t) \equiv [\mathrm{Ca}^{2+}](t)$, at the same times t, for four combinations of the parameters $\mathbf{p}$ indexed by m. These realizations of K_{d} and $\boldsymbol{\gamma}$ are evaluated by postprocessing the output of the THC model, based on Ermakova et al. (2020).

and

$$\frac{\mathrm{d}K_{\mathrm{d}}}{\mathrm{d}t} = \mathcal{G}(K_{\mathrm{d}}, \boldsymbol{\gamma}(t); \mathbf{p}), \quad K_{\mathrm{d}}(t=0;\mathbf{p}) = K_{\mathrm{d}}^0(\mathbf{p}). \tag{6}$$

If the surrogates [Eqs. (3) and (4)] are thought of as function approximations of K_{d}, then Eqs. (5) and (6) represent their respective dynamic counterparts. These dynamic approximations aim to learn not only K_{d} but also its rate of change at any time t. The explicit dependence of $\mathcal{F}$ on t is, once again, dictated by empirical evidence: for any realization m of the parameters $\mathbf{p}$, plotting $\mathrm{d}K_{\mathrm{d}}^{(m)}/\mathrm{d}t$ against $K_{\mathrm{d}}^{(m)}$ at the same times t, we found their relation to be multivalued (similar to Fig. 2). A numerical approximation of Eqs. (5) and (6) is provided in Appendix A.

The machine-learning techniques used to learn the function f and the functionals g, $\mathcal{F}$, and $\mathcal{G}$ in Eqs. (3)–(6) are presented below.

3. METHODOLOGY

The temporal evolution of K_{d} is highly nonlinear and can be qualitatively dissimilar for different input parameters $\mathbf{p}$ (Fig. 1). Therefore, we deploy RF (Section 3.2) and NN (Section 3.3) to construct the emulator of $K_{\mathrm{d}}(t;\mathbf{p})$. These machine-learning techniques are known to be better nonlinear function approximators than Gaussian-process emulators, polynomial regression, etc. To substantiate this claim, we demonstrated the poor performance of the Gaussian-process emulator (see Fig. B1 in Appendix B).

3.1 Data Preprocessing

Our data come from multiple realizations of the THC model [Eq. (1)] obtained for different combinations of the seven input parameters $\mathbf{p}$, i.e., the sample points $\mathbf{p}^{(m)}$ are drawn from

$$p_i^{(m)} \in \left\{ p_i^{\min}, \frac{2p_i^{\min} + p_i^{\max}}{3}, \frac{p_i^{\min} + 2p_i^{\max}}{3}, p_i^{\max} \right\}, \quad i \in \{1, \cdots, N_{\mathrm{par}} = 7\}. \tag{7}$$

However, some combinations are not physical, leading to false runs in the simulator. For example, output pH values were above 14, or the simulation did not reach the final time step due to lack of convergence. The false runs are dropped from the dataset. Then several parameter samples are drawn by perturbation around the sample pool in Eq. (7) to generate more $K_{\mathrm{d}}^{(m)}$ temporal evolution of desirable shapes. This adjustment aims to enrich the dataset with more balanced temporal variability of $K_{\mathrm{d}}^{(m)}$.

The preprocessing of these data consist of the following steps.

3.1.1 Normalization of Input and Output

Normalization is a technique often used to prepare the data for machine learning. It is necessary for stable convergence and better accuracy when "features," e.g., the input parameters $\mathbf{p}$ in Table 1, have vastly different ranges. Therefore, we rescale the parameters $\mathbf{p} \in \Gamma$ to $\tilde{\mathbf{p}} \in [-1,1]^{N_{\text{par}}}$ as the first step of data preparation, i.e., $\tilde{p}_i = \mathcal{R}_i(p_i)$ for $i = 1, \ldots, N_{\text{par}}$, where $\mathcal{R}_i$ is the rescaling map. If measurements of the simulated predictors $\boldsymbol{\gamma}(t)$ are available, we normalize them with their initial values in $\mathbf{p}$, i.e., $\tilde{\gamma}_1(t) = \mathcal{R}_3(\gamma_1(t))$ and $\tilde{\gamma}_2(t) = \mathcal{R}_4(\gamma_2(t))$. In what follows, we drop the tilde to simplify the notation.

Different values of the input parameters $\mathbf{p}$ can yield an orders-of-magnitude shift in the range of $K_{\text{d}}(t)$ (Fig. 1). To rescale K_{d} and to preserve the positivity, we consider $\ln K_{\text{d}}$ instead of K_{d} in the training and testing.

3.1.2 Decomposition of Parameter Space

Despite the nonlinearity of the K_{d} time series, one can still discern several distinct parameter regimes, i.e., the subdomains of the ($N_{\text{par}} = 7$)-dimensional parameter space $[-1,1]^{N_{\text{par}}}$. Low values of the initial pH (parameter p_3) create high K_{d} values at early reaction times [e.g., Fig. 1(b)]. A combination of high illite site density (p_6), smectite site density (p_5), and initial calcium concentration (p_4) with middle range pH (p_3) leads to K_{d} increasing over the observation time [e.g., Fig. 1(d)]. The majority of the $K_{\text{d}}(t)$ shapes is visually Gaussian [e.g., Figs. 1(a) and 1(c)].

Machine-learning tools for classification include k-means, support vector machines, and Gaussian mixtures. Conventional k-means techniques perform poorly on time series data because the Euclidean distance metric is not invariant to time shifts, while most time-series data hold such invariants. Therefore, we use k-means with dynamic time warping (DTW) (Tavenard et al., 2020) to deal with time shifts and gather time series of similar shapes. Figure 3 shows the results with $k = 2$ clusters. Collating the combinations of the parameters $\mathbf{p}$ that lead to the $K_{\text{d}}^{(m)}$ membership in cluster 1, we identify the corresponding parameter subspace of the region in $[-1,1]^7$, defined by a combination of high p_2 and middle range of p_3. The number of clusters is problem dependent; it is determined from the observation of the samples and the consideration of the data size. Our experiments with $(k = 3)$- and $(k = 4)$-means clustering showed a less satisfactory performance than $(k = 2)$-means clustering. That is because a larger number of clusters ($k = 3$ and 4) leads to less distinct cluster features and smaller training datasets.

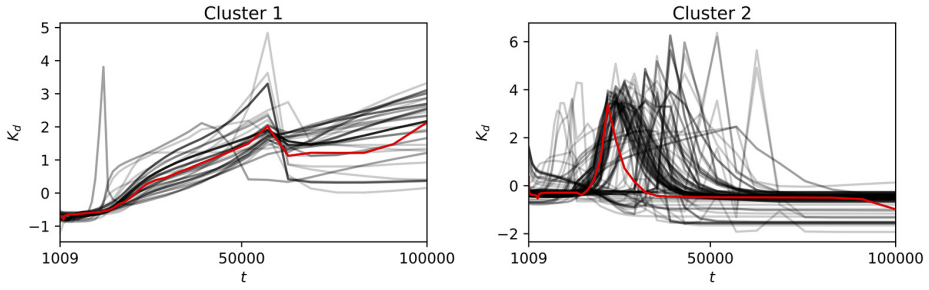

FIG. 3: Results of classification with $(k = 2)$-means clustering with dynamic time warping (DTW). Each subfigure represents (rescaled) $K_{\text{d}}^{(m)}$ time series from a given cluster and their centroid (in red).

Depending on the input parameter combination, the difference between the lowest and highest output values can be as high as nine orders of magnitude for K_d due to the wide input parameter ranges. In a majority of the cases, depending on the input values of pH (p_3) and adsorption surface area on smectite (p_2), and a combination of both, K_d may have an increasing shape. A high p_2, and $7 < \mathrm{pH} < 9$ may result in a constant level of K_d or slightly increasing K_d toward the end of the simulation cycle. This results from the increased concentration of bicarbonate ions (HCO_3–) at $7 < \mathrm{pH} < 9$, which leads to the formation of aqueous complexes with U(VI), and high adsorption surface area on smectite reduces the mobility of U(VI) and, as a result, increases K_d—cluster 1. The neutral or acidic (7.0) or high initial pH and adsorption surface area on smectite lower or equal to 10,000 may lead to an increase of contaminant in aqueous form and a decrease in K_d toward the end of the simulation cycle—cluster 2.

Based on this clustering observation, we construct the emulators that are trained on each cluster separately. During the test stage, we first determine the test sample's membership in one of the two clusters (based on values of the parameters p_2, p_3, and p_4) and then use the corresponding emulator to predict the temporal evolution of K_d. Our numerical experiments show that the emulators trained on the clustered data have better accuracy than their counterparts trained on the unclustered data.

3.1.3 Measurement of Performance

The data set consists of time series of $K_\mathrm{d}^{(m)}$ for N_MC parameter samples $\mathbf{p}^{(m)}$, with $m = 1, \ldots, N_\mathrm{MC}$. For each $K_\mathrm{d}^{(m)}$ time series, M snapshots are recorded. The times of these snapshots are logarithmically distributed from 10^3 years to 10^5 years in order to better capture the intense reactions at the beginning. To evaluate the performance of the constructed emulators, we reserve N_test pairs of $\mathbf{p}^{(m)}$ and the corresponding $K_\mathrm{d}^{(m)}$ time series for validation, while the rest ($N_\mathrm{train} = N_\mathrm{MC} - N_\mathrm{test}$) of the input-output pairs, $\mathbf{p}^{(m)}$ and $K_\mathrm{d}^{(m)}(t)$, are used for training. Membership in the test set $\mathcal{S}_{N_\mathrm{test}}$ is determined by randomly drawing N_test input-output pairs from the total of N_MC pairs, and the training set is randomly shuffled to reduce the bias caused by sequential ordering of the data. The accuracy of an emulator prediction of the mth member from the test set $\mathcal{S}_{N_\mathrm{test}}$ is measured by the relative L_2 norm,

$$\varepsilon_m = \sqrt{\frac{\sum_{k=1}^{M}\left[\ln K_\mathrm{d}^{(m)}(t_k) - \ln \tilde{K}_\mathrm{d}^{(m)}(t_k)\right]^2}{\sum_{k=1}^{M}\left[\ln K_\mathrm{d}^{(m)}(t_k)\right]^2}}, \quad \mathbf{p}^{(m)} \in \mathcal{S}_{N_\mathrm{test}}, \tag{8}$$

where $\tilde{K}_\mathrm{d}(t_k; \mathbf{p}^{(m)})$ is the prediction obtained by the RF or NN emulator.

3.2 Random Forest

Random forest (RF) belongs to the group of ensemble learning (Basu et al., 2018; Breiman, 2001). A group of regression trees (Breiman et al., 1984) are constructed at training time, and the output of RF is the mean prediction of the individual trees, illustrated in Fig. 4. Regression trees are known as "weak learners" in the sense that they have low bias but very high variance, especially for deep trees. In small-data problems, regression trees are seldom accurate due to their high variance (Friedman et al., 2001), known as "overfitting" issue in the machine learning community. To overcome this issue, RF employs multiple regression trees training on different

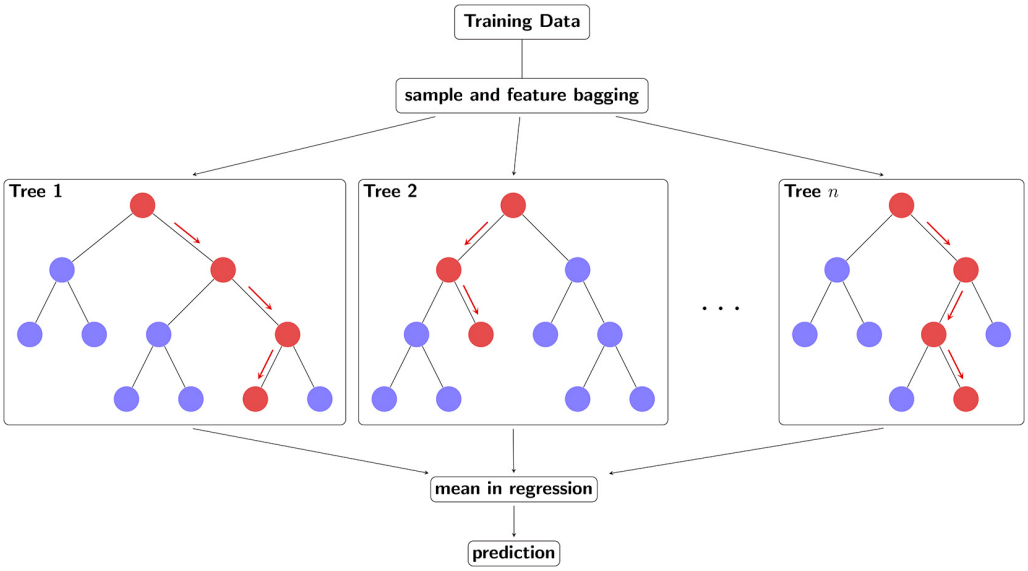

FIG. 4: Random forest learning

parts of the same training dataset [known as "sample bagging" strategy (Ho, 2002)] so that the average of all trees becomes a "strong learner" with effectively reduced variance and more accurate learning performance. The input-output data pairs need to be rearranged in the following format:

$$\text{input } \mathbf{X} = [X_1, \cdots, X_s]^\top \in \mathbb{R}^s \rightarrow \text{output } Y \in \mathbb{R}. \tag{9}$$

The rank of the importance of each input element $X_i, i = 1, \cdots, s$ is evaluated by importance scores (Zhu et al., 2015) during the fitting process of the RF. The details of RF algorithm can be found in the textbook (Friedman et al., 2001). We use the RF implementation of the RandomForestRegressor toolbox in sklearn package (Pedregosa et al., 2011). Details of implementation are illustrated in Section 4.

3.3 Neural Networks

Artificial neural networks are powerful and robust data-driven modeling tools, especially for nonlinear problems. In conventional notation, the input-output ($\mathbf{X}$-$\mathbf{Y}$) maps are approximated by a neural network $\mathcal{N}_\Theta$:

$$\mathbf{Y} \approx \mathcal{N}_\Theta(\mathbf{X}), \quad \mathbf{X} \in \mathbb{R}^s, \quad \mathbf{Y} \in \mathbb{R}^r, \tag{10}$$

where Θ is the parameter set including all the parameters in the network. A simple example is a linear input-output relation $\mathcal{N}_\Theta = \mathbf{W}$, where $\mathbf{W}$ is an $s \times r$ matrix of weights whose numerical values are obtained by minimizing the discrepancy between $\mathbf{Y}^{(m)}$ and $\mathcal{N}_\Theta(\mathbf{X}^{(m)})$, i.e., the following mean squared loss function:

$$L(\Theta) = \frac{1}{N_{\text{train}}} \sum_{m=1}^{N_{\text{train}}} \|\mathbf{Y}^{(m)} - \mathcal{N}_\Theta(\mathbf{X}^{(m)})\|^2, \tag{11}$$

where $\|\cdot\|$ denotes the vector 2-norm hereafter. The performance of this linear regression is likely to be suboptimal, especially for highly nonlinear problems like ours. Thus, one replaces $\mathcal{N}_\Theta = \mathbf{W}$ with a nonlinear model $\mathcal{N}_\Theta = \sigma \circ \mathbf{W}$ in which the prescribed function σ operates on each element of $\mathbf{WX}$. Popular choices for these so-called activation functions include sigmoid, hyperbolic tangent, rectified linear unit (ReLU), etc. In our numerical tests, we use ReLU as the activation function, i.e., $\sigma(X) = \max(0, X)$. The nonlinear model $\mathcal{N}_\Theta = \sigma \circ \mathbf{W}$ constitutes a single fully connected "layer" in a network. It has been established that such fully connected NN are universal approximators (Hornik, 1991; Pinkus, 1999).

A (deep) fully connected NN comprising $L \geq 3$ "layers" is constructed by a repeated application of the activation function to the input,

$$\mathcal{N}_\Theta = (\sigma_L \circ \mathbf{W}_{L-1}) \circ \cdots \circ (\sigma_2 \circ \mathbf{W}_1). \tag{12}$$

In general, different activation functions might be used in one network, and the last σ_L is an identity function, i.e., $\sigma_L(X) = X$. The layers except the input and output layers, are called "hidden layers." The parameter set $\Theta = \{\mathbf{W}_1, \cdots, \mathbf{W}_{L-1}\}$ consists of the weights $\mathbf{W}_l$ connecting the neurons from lth to $(l+1)$st layers. The weights $\mathbf{W}_1$ form a $s \times n_2$ matrix, $\mathbf{W}_2$ form a $n_3 \times n_2$ matrix, $\cdots$, and W_{L-1} is a $n_{L-1} \times r$ matrix, where the integers $n_l, l = 2, \cdots, L-1$ represent the number of neurons in each hidden layer. An example of a 3-layer fully connected NN architecture is shown in Fig. 5. As before, the fitting parameters Θ are obtained by minimizing Eq. (11). In practice, L_2 norm of the weights is added to the loss function Eq. (11) with small hyperparameter λ to avoid overfitting. The learning performance is evaluated by the prediction of $\mathcal{N}_\Theta(\mathbf{X}^{(m)})$ compared to $\mathbf{Y}^{(m)}$ in the test set. NN can be easily implemented using TensorFlow Keras application programming interface (API) (Chollet et al., 2015). The implementation details are illustrated in Section 4.

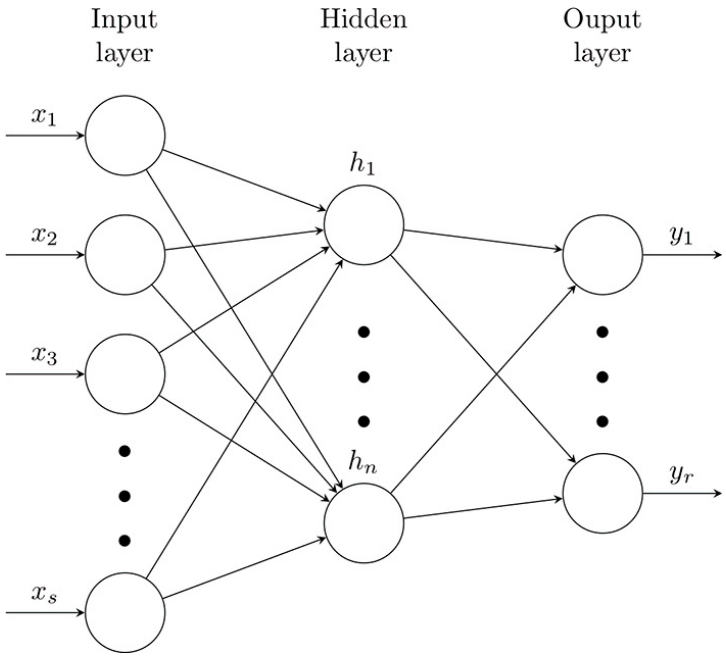

FIG. 5: An example of 3-layer fully connected NN architecture visualization

4. NUMERICAL RESULTS AND DISCUSSION

Our data set consists of $N_{\text{MC}} = 172$ input-output pairs, $\{\mathbf{p}^{(m)}, \boldsymbol{\gamma}^{(m)}(t), K_{\text{d}}^{(m)}(t)\}_{m=1}^{N_{\text{MC}}}$. Each output $K_{\text{d}}^{(m)}(t)$ consists of $M = 50$ snapshots collected at times $t_1, \ldots, t_M$. These data sets are split into the training and testing data sets consisting of $N_{\text{train}} = 166$ and $N_{\text{test}} = 6$ input-output pairs, respectively. The selection of both observation times $\{t_k\}_{k=1}^{M}$ and membership in the training and testing sets follows the procedures described in Section 3.1.

The $(k = 2)$-means with DTW classifier (section 3.1.2) identifies $N_1 = 123$ and $N_2 = 49$ members in clusters 1 and 2, respectively (Fig. 3). Thus, during subgroup learning, we have $N_1^{\text{train}} = 119$, $N_1^{\text{test}} = 4$ and $N_2^{\text{train}} = 47$, $N_2^{\text{test}} = 2$.

RF is implemented using the machine-learning package `scikit-learn`. The forest comprises $N_{\text{est}} = 1000$ regression trees. The maximum depth of each tree is not preset. The nodes are expanded until either all leaves are pure (i.e., cannot be split further) or all leaves contain less than 1 sample data. Bootstrap strategy is implemented by drawing the total number of training samples with replacement to fit each tree. In `RandomForestRegressor` toolbox, the implementation is to set $N_{\text{est}} = 1000$ with all other default values.

NN is implemented using `TensorFlow Keras` API. In NN, the neural network consists of $L = 7$ layers with $n_l = n_h$ $(l = 2, \ldots, L-1)$ neurons in each hidden layer. All the weights are initialized with `He initialization` (He et al., 2015), and all the biases are initialized to be zeros. To avoid overfitting, each layer is penalized by L_2 regularization of strength λ. The training data set is divided into mini batches of size 10, and the model is trained for 5000 epochs, after which the decrease in loss function is saturated. Minimization of the loss function Eq. (11) is done with the `Adam` algorithm, starting with learning rate α. If the monitored validation loss stagnates in a "patience" E_p epochs, then the learning rate is reduced by a factor of β until its preset minimum value. The hyperparameters, n_h, λ, α, and β, and the corresponding learning rate schedule need to be fine-tuned and thus are problem dependent.

4.1 Emulators in Eq. (3): Function Approximation without Observables

To construct the emulator Eq. (3), training of the RF [Eq. (9)] uses $N = N_{\text{train}} \times M$ input features $\mathbf{X} \in \mathbb{R}^8$ and the same number of output targets $Y \in \mathbb{R}$,

$$\{\mathbf{X}^{(i)}\}_{i=1}^{N} = \{t_k, \mathbf{p}^{(m)}\}_{(m,k)=(1,1)}^{(N_{\text{train}},M)} \quad \text{and} \quad \{Y^{(i)}\}_{i=1}^{N} = \{\ln K_{\text{d}}^{(m)}(t_k)\}_{(m,k)=(1,1)}^{(N_{\text{train}},M)}. \tag{13}$$

The testing dataset is arranged identically. The resulting RF-based emulator is denoted by f_{RF}, such that $\ln K_{\text{d}}(t) = f_{\text{RF}}(t; \mathbf{p})$. As a curious aside, we found that the importance scores of features $t, p_1, \ldots, p_7$, computed by the RF, coincide with their rankings obtained via the global sensitivity analysis (Ermakova et al., 2020).

The NN-based emulator Eq. (3), trained on the data in Eq. (13), is denoted by f_{NN1}, such that $\ln K_{\text{d}}(t) = f_{\text{NN1}}(t; \mathbf{p})$. As an alternative, we also build an NN-based emulator Eq. (3) that is trained on N_{train} input features $\mathbf{X} \in \mathbb{R}^7$ and output targets $\mathbf{Y} \in \mathbb{R}^M$,

$$\{\mathbf{X}^{(i)}\}_{i=1}^{N_{\text{train}}} = \{\mathbf{p}^{(m)}\}_{m=1}^{N_{\text{train}}}, \quad \{\mathbf{Y}^{(i)}\}_{i=1}^{N_{\text{train}}} = \{\ln K_{\text{d}}^{(m)}(t_1), \ldots, \ln K_{\text{d}}^{(m)}(t_M)\}_{m=1}^{N_{\text{train}}}. \tag{14}$$

We denote this NN-based emulator Eq. (3) by f_{NN2}, such that $\{\ln K_{\text{d}}(t_1), \ldots, \ln K_{\text{d}}(t_M)\} = f_{\text{NN2}}(\mathbf{p})$. We consider the emulators constructed with and without the clustering of the input data.

The left column of Fig. 6 demonstrates that, when trained on the data without clustering, all three emulators of type Eq. (3) yield satisfactory predictions of the distribution coefficient

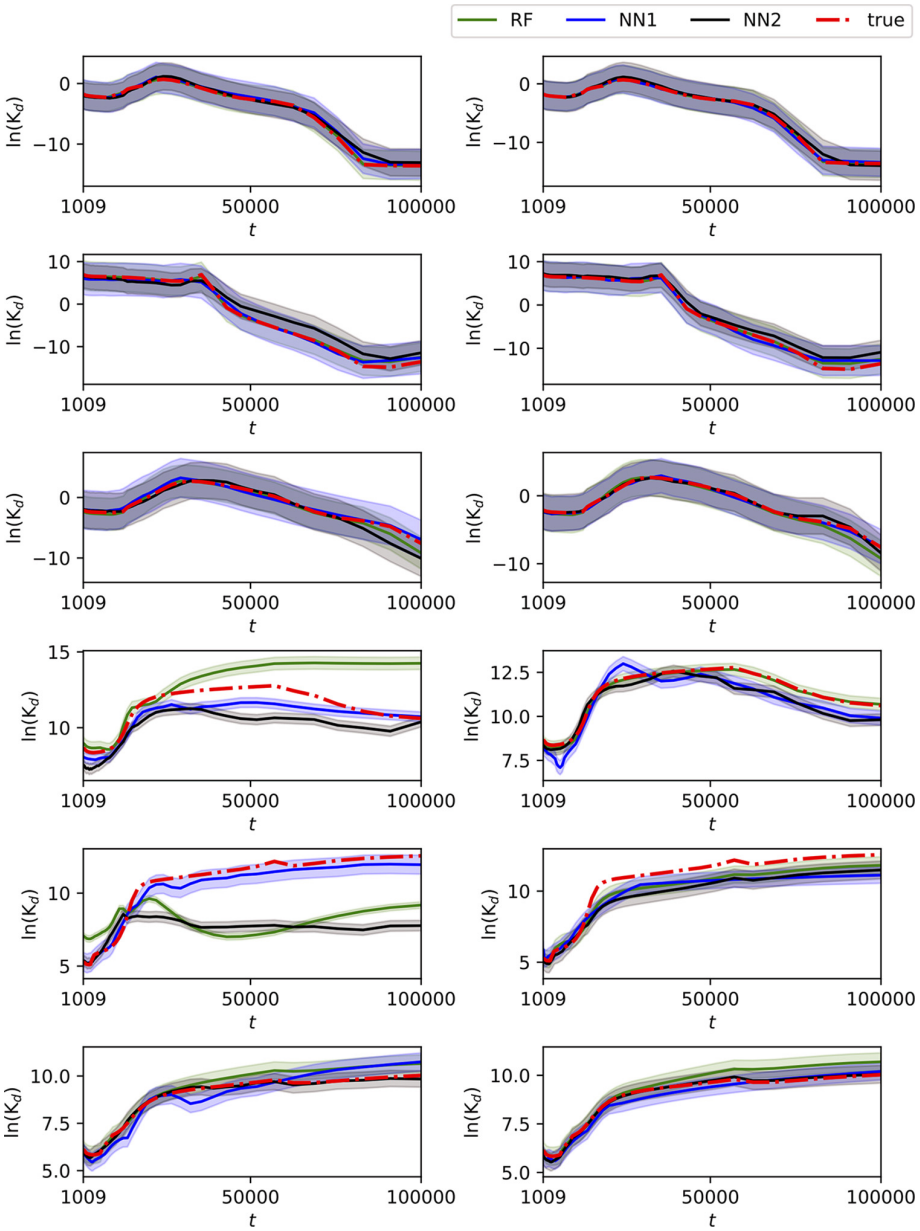

FIG. 6: RF- and NN-based emulators Eq. (3) without (left column) and with (right column) data clustering. For learning with two-class clustering, the top four graphs correspond to cluster 1, and the remaining bottom two to cluster 2, with both clusters identified in Fig. 3. The light green/blue/black region indicates 95% confidence interval.

$K_{\rm d}(t;\mathbf{p})$ in all but a few anomalous cases. This demonstrates the ability of the RF and NN emulators to capture the most common features from the data. The prior clustering of this data enables the RF and NN emulators to predict the anomalies as well (the right column of Fig. 6).

It leads to significant improvement in the emulators' accuracy. The light green/blue/black regions indicate 95% confidence intervals. The variances of the three emulators are comparable and case dependent. On average, NN2 has a slightly larger variance than the other two because the input-output formulation of NN2 does not consider the correlation between the t and $K_{\rm d}(t)$ time series. The size of the training data for NN2 is also much smaller, which may introduce higher variance in the prediction. The prior clustering does not show significant effects on the variance.

In Table 2, we report the relative errors ε_i of the three emulators, rendering this assessment more quantitative. Without clustering, NN1 outperforms the other two methods on half of the test cases and on an average of all 6 tests. NN2 performs worst both in most individual cases and in the average evaluation. With clustering, the error in each test case for every method drops significantly. On average, RF and NN2 improve their accuracy by nearly 50%, respectively. With the help of clustering, RF outperforms the other two methods and provides the most accurate predictions, as also shown in Fig. 6. There is only one test case, $i = 5$, where NN1 without clustering outperforms NN1 with clustering. It can be explained by the fact that NN is a highly nonlinear approximator with many hyperparameters (e.g., learning rate) and randomness (e.g., initialization). The NN surrogates are also heavily dependent on the training dataset. Therefore, it could happen for this particular test case that NN1 without clustering performs better than with clustering. Nevertheless, the prediction from NN1 with clustering still captures the increasing feature of $K_{\rm d}$ within reasonable error tolerance. The average performance of the test set still shows significant improvement when the clustering is used.

4.2 Emulators in Eq. (4): Function Approximation with Observables

To construct the emulator Eq. (4), training of the RF [Eq. (9)] uses $N = N_{\text{train}} \times M$ input features $\mathbf{X} \in \mathbb{R}^{10}$ and the same number of output targets $Y \in \mathbb{R}$,

$$\{\mathbf{X}^{(i)}\}_{i=1}^{N} = \{t_k, \boldsymbol{\gamma}(t_k), \mathbf{p}^{(m)}\}_{(m,k)=(1,1)}^{(N_{\text{train}},M)}, \quad \{Y^{(i)}\}_{i=1}^{N} = \{\ln K_{\rm d}^{(m)}(t_k)\}_{(m,k)=(1,1)}^{(N_{\text{train}},M)}. \tag{15}$$

The testing dataset is arranged identically. The resulting RF-based emulator is denoted by $f_{\rm RF}$, such that $\ln K_{\rm d}(t) = g_{\rm RF}(t, \boldsymbol{\gamma}(t); \mathbf{p})$.

The NN-based emulator Eq. (4), trained on the data in [Eq. (15)], is denoted by $g_{\rm NN1}$, such that $\ln K_{\rm d} = g_{\rm NN1}(t, \boldsymbol{\gamma}(t); \mathbf{p})$. As before, we also construct an alternative NN-based emulator Eq. (4) that is trained on N_{train} input features $\mathbf{X} \in \mathbb{R}^{3M+7}$ and output targets $\mathbf{Y} \in \mathbb{R}^{M}$,

$$\begin{aligned} \{\mathbf{X}^{(i)}\}_{i=1}^{N_{\text{train}}} &= \{t_1, \ldots, t_M, \boldsymbol{\gamma}(t_1), \ldots, \boldsymbol{\gamma}(t_M), \mathbf{p}^{(m)}\}_{m=1}^{N_{\text{train}}}, \\ \{\mathbf{Y}^{(i)}\}_{i=1}^{N_{\text{train}}} &= \{\ln K_{\rm d}^{(m)}(t_1), \ldots, \ln K_{\rm d}^{(m)}(t_M)\}_{m=1}^{N_{\text{train}}}. \end{aligned} \tag{16}$$

TABLE 2: Relative errors ε_i for the ith sample and their sample averages for the RF- and NN-based emulators Eq. (3) trained on the data without and with clustering

Clustering	Method	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$	Average
No	RF	0.0082	0.0447	0.1512	0.1107	0.2883	0.0349	0.1063
	NN1	0.0701	0.1015	0.1478	0.0655	0.0439	0.0517	0.0801
	NN2	0.1029	0.1717	0.2421	0.1142	0.2745	0.0158	0.1535
Yes	RF	0.0055	0.0433	0.1595	0.0096	0.0721	0.0365	0.0544
	NN1	0.0410	0.0700	0.0833	0.0630	0.0900	0.0326	0.0633
	NN2	0.0706	0.1212	0.0836	0.0424	0.1019	0.0228	0.0738

We denote this NN-based emulator Eq. (4) by g_{NN2}, such that $\{\ln K_{\text{d}}(t_1), \ldots, \ln K_{\text{d}}(t_M)\} = g_{\text{NN2}}(t_1, \ldots, t_M, \boldsymbol{\gamma}(t_1), \ldots, \boldsymbol{\gamma}(t_M); \mathbf{p})$. Again, we consider the emulators constructed with and without the clustering of the input data.

The performance of these emulators is visually similar to that of their counterparts in Fig. 6. The errors ε_i reported in Table 3 demonstrate that the addition of the simulated predictors, i.e., the use of emulators Eq. (4) instead of Eq. (3), yields more accurate predictions. The accuracy of each method with simulated predictors and no clustering is already comparable to the accuracy (reported in Table 2) of each method without simulated predictors but with clustering. This justifies the strong correlations between the simulated predictors and K_{d}. Similarly as before, clustering helps in improving the accuracy of each method (though less significantly). RF outperforms the other two methods by nearly 50%, on average, with or without clustering.

4.3 Emulators in Eq. (5): Dynamic Approximation without Observables

Construction of the RF- and NN-based emulators Eq. (5) relies on the same input-output training set. It consists of $N = N_{\text{train}} \times (M-1)$ input features $\mathbf{X} \in \mathbb{R}^9$ and output targets $Y \in \mathbb{R}$,

$$\begin{aligned} \{\mathbf{X}^{(i)}\}_{i=1}^{N} &= \{\ln K_{\text{d}}^{(m)}(t_k), t_k, \mathbf{p}^{(m)}\}_{(m,k)=(1,1)}^{(N_{\text{train}}, M-1)}, \\ \{Y^{(i)}\}_{i=1}^{N} &= \{\ln K_{\text{d}}^{(m)}(t_{k+1}) - \ln K_{\text{d}}^{(m)}(t_k)\}_{(m,k)=(1,1)}^{(N_{\text{train}}, M-1)}. \end{aligned} \tag{17}$$

In this case, the NN learning process coincides with ResNet (He et al., 2016). The resulting RF- and NN-based emulators are denoted by $\mathcal{F}_\alpha$, where $\alpha =$ RF and NN, respectively (see Appendix A for details).

The testing on the data set $\mathcal{S}_{N_{\text{test}}}$ is carried out as follows. Given the parameters $\mathbf{p}^{(m)}$ and the corresponding values $\ln K_{\text{d}}^{(m)}(t_1)$ from $\mathcal{S}_{N_{\text{test}}}$, the values of $\ln K_{\text{d}}^{(m)}$ at later times $(t_2, \ldots, t_M)$ are computed iteratively as

$$\ln K_{\text{d}}^{(m)}(t_{k+1}) = \ln K_{\text{d}}^{(\text{m})}(t_k) + \mathcal{F}_\alpha(\ln K_{\text{d}}^{(\text{m})}(t_k), t_k; \mathbf{p}^{(m)}), \quad k = 1, \ldots, M-1, \tag{18}$$

where $\alpha =$ RF and NN. These predicted values are then compared with the data, and the corresponding errors ε_m are computed.

The predicted distribution coefficients $K_{\text{d}}^{(m)}(t) \in \mathcal{S}_{N_{\text{test}}}$ exhibit the qualitative behavior similar to that in Fig. 6. As before, the training with prior clustering improves the the emulators' accuracy. Although the iterative procedure [Eq. (18)] adds up the error at every time step, one still maintains good accuracy. The errors ε_m in Table 4 show that the performances of RF and

TABLE 3: Relative errors ε_i for the ith sample and their sample averages for the RF- and NN-based emulators Eq. (4) trained on the data without and with clustering

Clustering	Method	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$	Average
No	RF	0.0042	0.0303	0.0660	0.0952	0.0368	0.0026	0.0392
	NN1	0.0264	0.0643	0.0803	0.1323	0.1416	0.0185	0.0772
	NN2	0.0497	0.0506	0.1277	0.0898	0.0590	0.0398	0.0694
Yes	RF	0.0037	0.0317	0.0649	0.0335	0.0571	0.0026	0.0322
	NN1	0.0320	0.0557	0.0593	0.0480	0.0376	0.1187	0.0585
	NN2	0.0449	0.0527	0.1324	0.0207	0.0605	0.0363	0.0579

TABLE 4: Relative errors, ε_i, for the ith sample and their sample-averages for the RF- and NN-based emulators Eq. (5) trained on the data without and with clustering

Clustering	Method	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$	Average
No	RF	0.0190	0.0921	0.1814	0.0394	0.2390	0.0273	0.0997
	NN	0.0642	0.1167	0.1109	0.1022	0.1063	0.0408	0.0902
Yes	RF	0.0087	0.0686	0.1615	0.0589	0.0448	0.0305	0.0622
	NN	0.0397	0.0792	0.1360	0.0829	0.0429	0.0207	0.0669

NN have about the same accuracy in the test average error. Comparing Table 2 with Table 4, we observe that emulator Eq. (3) and emulator Eq. (5) have comparable learning performances.

4.4 Emulators in Eq. (6): Dynamic Approximation with Observables

Construction of the RF- and NN-based emulators Eq. (6) relies on the same input-output training set. It consists of $N = N_{\text{train}} \times (M-1)$ input features $\mathbf{X} \in \mathbb{R}^{12}$ and output targets $Y \in \mathbb{R}$,

$$\begin{aligned} \{\mathbf{X}^{(i)}\}_{i=1}^{N} &= \{\ln K_{\text{d}}^{(m)}(t_k), \boldsymbol{\gamma}^{(m)}(t_k), \boldsymbol{\gamma}^{(m)}(t_{k+1}) - \boldsymbol{\gamma}^{(m)}(t_k), \mathbf{p}^{(m)}\}_{(m,k)=(1,1)}^{(N_{\text{train}},M-1)}, \\ \{Y^{(i)}\}_{i=1}^{N} &= \{\ln K_{\text{d}}^{(m)}(t_{k+1}) - \ln K_{\text{d}}^{(m)}(t_k)\}_{(m,k)=(1,1)}^{(N_{\text{train}},M-1)}. \end{aligned} \tag{19}$$

The resulting RF- and NN-based emulators are denoted by $\mathcal{G}_\alpha$, where $\alpha =$ RF and NN, respectively (see Appendix A for details).

The testing on the data set $\mathcal{S}_{N_{\text{test}}}$ follows the procedure described in the previous section. Given the parameters $\mathbf{p}^{(m)}$, the corresponding set of observables $\boldsymbol{\gamma}^{(m)}(t_1), \ldots, \boldsymbol{\gamma}^{(m)}(t_M)$, and the corresponding values $\ln K_{\text{d}}^{(m)}(t_1)$ from $\mathcal{S}_{N_{\text{test}}}$, the values of $\ln K_{\text{d}}^{(m)}$ at later times $(t_2, \ldots, t_M)$ are computed iteratively as

$$\begin{aligned} \ln K_{\text{d}}^{(\text{m})}(t_{k+1}) = \ln K_{\text{d}}^{(\text{m})}(t_k) \\ + \mathcal{G}_\alpha(\ln K_{\text{d}}^{(\text{m})}(t_k), \boldsymbol{\gamma}^{(m)}(t_k), \boldsymbol{\gamma}^{(m)}(t_{k+1}) - \boldsymbol{\gamma}^{(m)}(t_k); \mathbf{p}^{(m)}), \end{aligned} \tag{20}$$

for $k = 1, \ldots, M-1$. These predicted values are then compared with the data, and the corresponding errors ε_m are computed.

The predicted distribution coefficients $K_{\text{d}}^{(m)}(t) \in \mathcal{S}_{N_{\text{test}}}$ are shown in Fig. 7. The true K_{d} lies in the 95% confidence interval of either RF or NN, even with no clustering. The accuracy is further enhanced by the prior clustering, as more clearly shown in the quantitative error evaluations in Table 5. Although the improvement by using clustering is not significant due to the relatively high accuracy of the emulators without clustering, RF-based emulators Eq. (6) with prior clustering provide the most accurate predictions. There are no significant differences between the variances of the two emulators and between the variances of predictions with/without clustering.

4.5 Inter-Method Comparison

The numerical experiments reported in Sections 4.1–4.4 demonstrate a comparable performance of the RF- and NN-based emulators, with the former having better accuracy for some test samples and the latter for the others. The data clustering strategy improves the accuracy in all scenarios. We summarize the observations and comparisons from the following three perspectives.

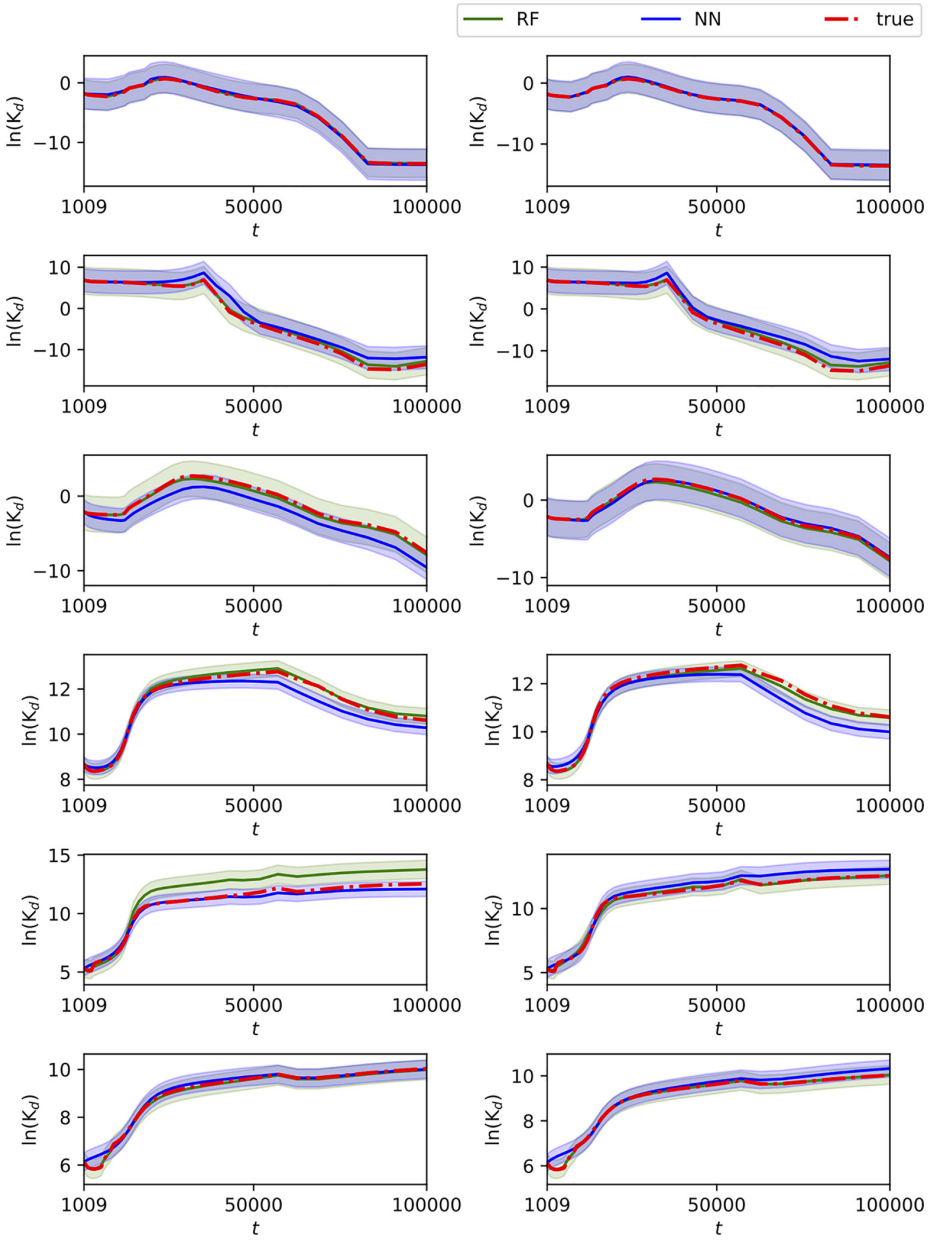

FIG. 7: RF- and NN-based emulators Eq. (6) without (left column) and with (right column) data clustering. For the learning with two-class clustering, the top four graphs correspond to cluster 1 and the remaining bottom two to cluster 2, with both clusters identified in Fig. 3. The light green/blue/black region indicates 95% confidence interval.

4.5.1 RF- and NN-Based Emulators

When no data clustering is performed, the RF-based emulators outperform the NN-based ones in terms of average accuracy. This is due to the power of ensemble learning strategy in the RF

TABLE 5: Relative errors ε_i for the ith sample and their sample averages for the RF- and NN-based emulators Eq. (6) trained on the data without and with clustering

Clustering	Method	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$	Average
No	RF	0.0062	0.0409	0.0851	0.0079	0.0912	0.0051	0.0394
	NN	0.0502	0.1504	0.4108	0.0217	0.0278	0.0330	0.1157
Yes	RF	0.0053	0.0513	0.0867	0.0090	0.0179	0.0032	0.0289
	NN	0.0270	0.1319	0.0769	0.0287	0.0378	0.0400	0.0570

algorithm. NNs are designed for big-data problems and, under those conditions, perform well as surrogates (Tripathy and Bilionis, 2018; Zhu et al., 2019). For small-data problems like ours, high variance in a single NN emulator can comprise the prediction accuracy. One can improve the NN emulator by borrowing ideas from RF to construct ensemble neural networks (Izmailov et al., 2018; Krogh and Vedelsby, 1995). This involves training multiple NN models, instead of a single NN model, and combining the predictions from these models. Studies on ensemble learning (Opitz and Maclin, 1999; Polikar, 2006; Rokach, 2010) show that this strategy not only reduces the variance of predictions but also can result in predictions that are better than any single model.

4.5.2 The NN1 and NN2 Designs of NN-Based Emulators

The NN1 considers a scalar prediction with time as an input variable, while NN2 considers a time series vector as the target variable. These two input-output designs of NNs approximate different mappings. In the absence of a theoretical explanation of which strategy is to be preferred, we relied on numerical experimentation. We found NN1 to be more stable, i.e., its prediction accuracy is not much affected by the random initialization or train-test splitting. This can be attributed to two factors. First, the input-output design of NN1 enables more data than NN2 (M times more data). Hence, the NN2 prediction may have higher variance and thus be less stable. Second, the NN2 formulation does not consider the correlation between the t and $K_\mathrm{d}(t)$ time series. This shortcoming can be overcome by the introduction of input-simulated predictors, which account for time correlations, as we have done in Sections 4.2 and 4.4. Also, NN1 has better applicability when the emulator is used by another model in which time is usually an input variable.

4.5.3 Function Approximation vs. Dynamic Formulation

The RF- and NN-based emulators based on function approximation (Sections 4.1 and 4.2) provide a local representation of the distribution coefficient $K_\mathrm{d}(t)$. Their counterparts based on dynamic formulation (Sections 4.3 and 4.4) are nonlocal, i.e., account for the memory effect. Since the "true" evolution of K_d is unknown, one cannot determine the preferred formulation without further investigating the dominant dynamics of the upscaled system. Our numerical experiments demonstrate comparable prediction results in both formulations. Other data sets, generated at different scales, may show different relative performance. Although dynamics formulation would be favorable for real-time estimation, it might not make a significant difference since our predictions are long term, and direction observations over time are not possible.

4.5.4 Impact of Simulated Predictors

The incorporation of simulated predictors, e.g., time series of pH and $[Ca^{2+}]$ used in our examples, significantly improves the learning performance. Since the simulated predictors depend on the geochemical and transport processes, they provide certain information about the full multiphysics system. Therefore, measurements of most correlated simulated predictors play a key role in forecasting upscaled quantities of interest.

5. CONCLUSIONS

Development of reduced-order models for reactive transport models remains an open challenge due to nonlinearity and parameter interactions. We constructed RF- and NN-based emulators to represent the buffer-averaged distribution coefficient K_{d} as a function of input parameters $\mathbf{p}$ and time t. To the best of our knowledge, ours are the first successful emulators for such systems that are both accurate and computationally efficient.

We explored two formulations of the RF- and NN-based emulators of $K_{\mathrm{d}}(t;\mathbf{p})$: function approximation and dynamic approximation. We also introduced two strategies to boost the learning performance of all the emulators considered. The first relies on k-means clustering with dynamic time warping of the temporal data. The second incorporates geochemical simulated predictors, e.g., time series of pore water pH and calcium concentration, into the learning process.

The emulators provide orders-of magnitude computational speedup: the average simulation time for one run of the reactive transport model is about 26 hours, while the training time for a well designed RF- or NN-based emulator is within 10 minutes [on a machine with Intel(R) Core(TM) i7-6700 at 3.40 GHz processor].

Our use of an emulator can be thought of as numerical upscaling, which is distinct from theoretical upscaling (Korneev and Battiato, 2016; Lichtner and Tartakovsky, 2003; Neuman and Tartakovsky, 2009). A good emulator provides not only a meaningful representation of a complex system but also a bridge connecting high-fidelity (e.g., pore-scale or fine-resolution) models and their low-fidelity (e.g., field-scale) counterparts. In the future, we will implement our K_{d} emulators in a performance assessment (PA) model. That effort would consist of the use of the temporally variable K_{d} of U(VI) as input parameters in PFLOTRAN simulations of the PA model. In addition, we will test our surrogate models further, improve their accuracy, build in uncertainty quantifications, and account for the interactions among parameters.

ACKNOWLEDGMENTS

Funding for this work was provided by the Spent Fuel and Waste Science and Technology, Office of Nuclear Energy, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231, with Lawrence Berkeley National Laboratory. Research at Stanford was supported in part by Air Force Office of Scientific Research under award No. FA9550-18-1-0474 and the Advanced Research Projects Agency-Energy (ARPA-E), U.S. Department of Energy, under Award No. DE-AR0001202; and a gift from Total. There are no data sharing issues since all of the numerical information is provided in the figures produced by solving the equations in the paper.

REFERENCES

Audigane, P., Gaus, I., Czernichowski-Lauriol, I., Pruess, K., and Xu, T., Two-Dimensional Reactive Transport Modeling of CO_2 Injection in a Saline Aquifer at the Sleipner Site, North Sea, *Am. J. Sci.*, vol. **307**, no. 7, pp. 974–1008, 2007.

Barthelemy, J.F.M. and Haftka, R.T., Approximation Concepts for Optimum Structural Design—A Review, *Struct. Optimiz.*, vol. **5**, no. 3, pp. 129–144, 1993.

Basu, S., Kumbier, K., Brown, J.B., and Yu, B., Iterative Random Forests to Discover Predictive and Stable High-Order Interactions, *Proc. Nat. Acad. Sci.*, vol. **115**, no. 8, pp. 1943–1948, 2018.

Bea, S.A., Wainwright, H., Spycher, N., Faybishenko, B., Hubbard, S.S., and Denham, M.E., Identifying Key Controls on the Behavior of an Acidic-U (VI) Plume in the Savannah River Site Using Reactive Transport Modeling, *J. Contam. Hydrol.*, vol. **151**, pp. 34–54, 2013.

Bianchi, M., Zheng, L., and Birkholzer, J.T., Combining Multiple Lower-Fidelity Models for Emulating Complex Model Responses for CCS Environmental Risk Assessment, *Int. J. Greenhouse Gas Contr.*, vol. **46**, pp. 248–258, 2016.

Booker, D. and Woods, R., Comparing and Combining Physically-Based and Empirically-Based Approaches for Estimating the Hydrology of Ungauged Catchments, *J. Hydrol.*, vol. **508**, pp. 227–239, 2014.

Breiman, L., Random Forests, *Mach. Learn.*, vol. **45**, no. 1, pp. 5–32, 2001.

Breiman, L., Friedman, J., Stone, C.J., and Olshen, R.A., *Classification and Regression Trees*, Boca Raton, FL: CRC press, 1984.

Chaturantabut, S. and Sorensen, D.C., Nonlinear Model Reduction via Discrete Empirical Interpolation, *SIAM J. Sci. Comput.*, vol. **32**, no. 5, pp. 2737–2764, 2010.

Chollet, F., *Keras*, accessed from https://keras.io, 2015.

Ermakova, D., Wainwright, H., Zheng, L., Shirley, I., and Lu, H., Global Sensitivity Analysis for U(VI) Transport for Integrating Coupled Thermal–Hydrological–Chemical Processes Models into Performance Assessment Model, *ASME J. Nuclea. Rad. Sci.*, vol. **7**, no. 4, p. 041902, 2020.

Forrester, A.I.J. and Keane, A.J., Recent Advances in Surrogate-Based Optimization, *Progr. Aerosp. Sci.*, vol. **45**, nos. 1-3, pp. 50–79, 2009.

Friedman, J., Hastie, T., and Tibshirani, R., *The Elements of Statistical Learning*, vol. **1**, New York, NY: Springer, 2001.

He, K., Zhang, X., Ren, S., and Sun, J., Delving Deep into Rectifiers: Surpassing Human-Level Performance on Imagenet Classification, in *Proc. of the IEEE Int. Conf. on Computer Vision*, pp. 1026–1034, 2015.

He, K., Zhang, X., Ren, S., and Sun, J., Deep Residual Learning for Image Recognition, in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Hesthaven, J.S. and Ubbiali, S., Nonintrusive Reduced Order Modeling of Nonlinear Problems Using Neural Networks, *J. Comput. Phys.*, vol. **363**, pp. 55–78, 2018.

Ho, T.K., A Data Complexity Analysis of Comparative Advantages of Decision Forest Constructors, *Pattern Anal. Appl.*, vol. **5**, no. 2, pp. 102–112, 2002.

Hornik, K., Approximation Capabilities of Multilayer Feedforward Networks, *Neural Net.*, vol. **4**, no. 2, pp. 251–257, 1991.

Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., and Wilson, A.G., Averaging Weights Leads to Wider Optima and Better Generalization, 2018. arXiv: 1803.05407

Kerschen, G., Golinval, J.C., Vakakis, A.F., and Bergman, L.A., The Method of Proper Orthogonal Decomposition for Dynamical Characterization and Order Reduction of Mechanical Systems: An Overview,

Nonlinear Dyna., vol. **41**, nos. 1-3, pp. 147–169, 2005.

Korneev, S. and Battiato, I., Sequential Homogenization of Reactive Transport in Polydisperse Porous Media, *Multiscale Model. Simul.*, vol. **14**, no. 4, pp. 1301–1318, 2016.

Krogh, A. and Vedelsby, J., Neural Network Ensembles, Cross Validation, and Active Learning, *Adv. Neural Inform. Proc. Sys.*, pp. 231–238, 1995.

Kutz, J.N., Brunton, S.L., Brunton, B.W., and Proctor, J.L., *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*, Philadelphia, PA: SIAM, 2016.

Lichtner, P.C. and Tartakovsky, D.M., Stochastic Analysis of Effective Rate Constant for Heterogeneous Reactions, *Stoch. Environm. Res. Risk Assess.*, vol. **17**, no. 6, pp. 419–429, 2003.

Lucia, D.J., Beran, P.S., and Silva, W.A., Reduced-Order Modeling: New Approaches for Computational Physics, *Progr. Aerosp. Sci.*, vol. **40**, nos. 1-2, pp. 51–117, 2004.

Maday, Y. and Mula, O., A Generalized Empirical Interpolation Method: Application of Reduced Basis Techniques to Data Assimilation, in *Analysis and Numerics of Partial Differential Equations*, Berlin, Germany: Springer, pp. 221–235, 2013.

Naghibi, S.A., Pourghasemi, H.R., and Dixon, B., GIS-Based Groundwater Potential Mapping Using Boosted Regression Tree, Classification and Regression Tree, and Random Forest Machine Learning Models in Iran, *Environ. Monitor. Assess.*, vol. **188**, no. 1, p. 44, 2016.

Neuman, S.P. and Tartakovsky, D.M., Perspective on Theories of Nonfickian Transport in Heterogeneous Media, *Adv. Water Res.*, vol. **32**, no. 5, pp. 670–680, 2009.

Opitz, D. and Maclin, R., Popular Ensemble Methods: An Empirical Study, *J. Artif. Intel. Res.*, vol. **11**, pp. 169–198, 1999.

Pau, G.S.H., Zhang, Y., and Finsterle, S., Reduced Order Models for Many-Query Subsurface Flow Applications, *Comput. Geosci.*, vol. **17**, no. 4, pp. 705–721, 2013.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E., Scikit-Learn: Machine Learning in Python, *J. Mach. Learn. Res.*, vol. **12**, pp. 2825–2830, 2011.

Pinkus, A., Approximation Theory of the MLP Model in Neural Networks, *Acta Numer.*, vol. **8**, no. 1, pp. 143–195, 1999.

Polikar, R., Ensemble Based Systems in Decision Making, *IEEE Circ. Sys. Mag.*, vol. **6**, no. 3, pp. 21–45, 2006.

Qin, T., Chen, Z., Jakeman, J., and Xiu, D., Data-Driven Learning of Nonautonomous Systems, 2020. arXiv: 2006.02392

Qin, T., Wu, K., and Xiu, D., Data Driven Governing Equations Approximation Using Deep Neural Networks, *J. Comput. Phys.*, vol. **395**, pp. 620–635, 2019.

Rasmussen, C.E., Gaussian Processes in Machine Learning, *Summer School on Machine Learning*, Berlin, Germany: Springer, pp. 63–71, 2003.

Razavi, S., Tolson, B.A., and Burn, D.H., Review of Surrogate Modeling in Water Resources, *Water Res. Res.*, vol. **48**, no. 7, 2012.

Rokach, L., Ensemble-Based Classifiers, *Artif. Intel. Rev.*, vol. **33**, nos. 1-2, pp. 1–39, 2010.

Rowley, C.W., Model Reduction for Fluids, Using Balanced Proper Orthogonal Decomposition, *Int. J. Bifurcat. Chaos*, vol. **15**, no. 3, pp. 997–1013, 2005.

Rutqvist, J., Zheng, L., Chen, F., Liu, H.H., and Birkholzer, J., Modeling of Coupled Thermo-Hydro-Mechanical Processes with Links to Geochemistry Associated with Bentonite-Backfilled Repository Tunnels in Clay Formations, *Rock Mech. Rock Eng.*, vol. **47**, no. 1, pp. 167–186, 2014.

Saridakis, K.M. and Dentsoras, A.J., Soft Computing in Engineering Design—A Review, *Adv. Eng. Inform.*, vol. **22**, no. 2, pp. 202–221, 2008.

Schmid, P.J., Dynamic Mode Decomposition of Numerical and Experimental Data, *J. Fluid Mech.*, vol. **656**, pp. 5–28, 2010.

Schmit, L., Jr. and Farshi, B., Some Approximation Concepts for Structural Synthesis, *AIAA J.*, vol. **12**, no. 5, pp. 692–699, 1974.

Simpson, T.W., Poplinski, J., Koch, P.N., and Allen, J.K., Metamodels for Computer-Based Engineering Design: Survey and Recommendations, *Eng. Comput.*, vol. **17**, no. 2, pp. 129–150, 2001.

Steefel, C.I., Appelo, C.A.J., Arora, B., Jacques, D., Kalbacher, T., Kolditz, O., Lagneau, V., Lichtner, P.C., Mayer, K.U., Meeussen, J.C.L., Reactive Transport Codes for Subsurface Environmental Simulation, *Comput. Geosci.*, vol. **19**, no. 3, pp. 445–478, 2015.

Steefel, C.I., DePaolo, D.J., and Lichtner, P.C., Reactive Transport Modeling: An Essential Tool and a New Research Approach for the Earth Sciences, *Earth Planet. Sci. Lett.*, vol. **240**, nos. 3-4, pp. 539–558, 2005.

Tavenard, R., Faouzi, J., Vandewiele, G., Divo, F., Androz, G., Holtz, C., Payne, M., Yurchak, R., Rußwurm, M., Kolar, K., and Woods, E., Tslearn, a Machine Learning Toolkit for Time Series Data, *J. Mach. Learn. Res.*, vol. **21**, no. 118, pp. 1–6, 2020.

Tripathy, R.K. and Bilionis, I., Deep UQ: Learning Deep Neural Network Surrogate Models for High Dimensional Uncertainty Quantification, *J. Comput. Phys.*, vol. **375**, pp. 565–588, 2018.

Xiong, Y., Fakcharoenphol, P., Winterfeld, P., Zhang, R., and Wu, Y.S., Coupled Geomechanical and Reactive Geochemical Model for Fluid and Heat Flow: Application for Enhanced Geothermal Reservoir, *SPE Reservoir Characterization and Simulation Conf. and Exhibition*, Society of Petroleum Engineers, 2013.

Xu, T., Sonnenthal, E., Spycher, N., and Zheng, L., TOUGHREACT V3. 0-OMP Reference Manual: A Parallel Simulation Program for Non-Isothermal Multiphase Geochemical Reactive Transport, University of California, Berkeley, CA, 2014.

Zheng, L., Rutqvist, J., Xu, H., and Birkholzer, J.T., Coupled THMC Models for Bentonite in an Argillite Repository for Nuclear Waste: Illitization and Its Effect on Sweling Stress under High Temperature, *Eng. Geol.*, vol. **230**, pp. 118–129, 2017.

Zhou, Z. and Tartakovsky, D.M., Markov Chain Monte Carlo with Neural Network Surrogates: Application to Contaminant Source Identification, *Stoch. Env. Res. Risk Assess.*, 2020.

Zhu, R., Zeng, D., and Kosorok, M.R., Reinforcement Learning Trees, *J. Am. Stat. Assoc.*, vol. **110**, no. 512, pp. 1770–1784, 2015.

Zhu, Y., Zabaras, N., Koutsourelakis, P.S., and Perdikaris, P., Physics-Constrained Deep Learning for High-Dimensional Surrogate Modeling and Uncertainty Quantification without Labeled Data, *J. Comput. Phys.*, vol. **394**, pp. 56–81, 2019.

APPENDIX A. NUMERICAL APPROXIMATIONS OF $K_\mathbf{d}$ DYNAMICS

Our goal is to provide an accurate approximation to the true solutions of Eqs. (5) and (6) at prescribed times $\{t_1, \ldots, t_M\}$. Since the analysis of Eq. (5) follows directly from that of Eq. (6), we show the latter in detail and provide only the final result for the former.

Without loss of generality, we use constant time step

$$\Delta t = t_{k+1} - t_k, \quad \text{for} \quad k = 0, \ldots, M-1. \tag{A.1}$$

For each time interval $[t_k, t_{k+1}]$, with $k = 0, \ldots, M-1$, we first seek a first-order local parameterization for the simulated predictors $\boldsymbol{\gamma}(t)$:

$$\boldsymbol{\gamma}(t_k + \tau; \mathbf{p}) \approx \boldsymbol{\gamma}(t_k; \mathbf{p}) + [\boldsymbol{\gamma}(t_{k+1}; \mathbf{p}) - \boldsymbol{\gamma}(t_k; \mathbf{p})]\tau, \quad \tau \in [0, \Delta t]. \tag{A.2}$$

Then a global parameterization is constructed as

$$\boldsymbol{\gamma}(t; \mathbf{p}) \approx \hat{\boldsymbol{\gamma}}(t; \mathbf{p}) \equiv \sum_{k=0}^{M-1} [\boldsymbol{\gamma}(t_k; \mathbf{p}) + (\boldsymbol{\gamma}(t_{k+1}; \mathbf{p}) - \boldsymbol{\gamma}(t_k; \mathbf{p}))(t - t_k)]\mathbb{I}_{[t_k, t_{k+1}]}(t), \tag{A.3}$$

where the indicator function $\mathbb{I}$ is defined by

$$\mathbb{I}_{[t_k, t_{k+1}]}(t) = \begin{cases} 1 & \text{if } t \in [t_k, t_{k+1}], \\ 0 & \text{otherwise.} \end{cases} \tag{A.4}$$

If the simulated predictors $\boldsymbol{\gamma}(t; \mathbf{p})$ are measured either continuously or at more than the two endpoints during all time intervals $[t_k, t_{k+1}]$, then higher-order local parameterizations can be constructed (Qin et al., 2020).

Substituting Eq. (A.3) into Eq. (6) and using a first-order approximation of the derivative in the latter, we obtain

$$K_\mathrm{d}(t_{k+1}; \mathbf{p}) = K_\mathrm{d}(t_k; \mathbf{p}) + \Delta t \hat{\mathcal{G}}(K_\mathrm{d}(t_k), \hat{\boldsymbol{\gamma}}(t_k), \hat{\boldsymbol{\gamma}}(t_{k+1}) - \hat{\boldsymbol{\gamma}}(t_k); \mathbf{p}), \tag{A.5}$$

for $k = 0, \ldots, M-1$. Here the target of our approximation $\hat{\mathcal{G}} : \mathbb{R} \times \mathbb{R}^2 \times \mathbb{R}^2 \times \mathbb{R}^7 \to \mathbb{R}$ is a fully discretized numerical evaluation of $\mathcal{G}$ with the choice of local parameterization [Eq. (A.2)].

Similarly, Eq. (5) is approximated with

$$K_\mathrm{d}(t_{k+1}; \mathbf{p}) = K_\mathrm{d}(t_k; \mathbf{p}) + \hat{\mathcal{F}}(K_\mathrm{d}(t_k), t_k, \Delta t; \mathbf{p}), \quad k = 0, \ldots, M-1. \tag{A.6}$$

The Δt-flow maps $\hat{\mathcal{F}}$ and $\hat{\mathcal{G}}$ are the target functions for our RF and NN learning methods. The emulators for these flow maps, denoted by $\mathcal{G}_\alpha$ and $\mathcal{F}_\alpha$ (with $\alpha = \mathrm{RF}, \mathrm{NN}$) in Eqs. (18) and (20), are employed iteratively as surrogates to approximate K_d values at times $\{t_1, \ldots, t_M\}$.

APPENDIX B. GP PERFORMANCE

We use a vanilla GP with common kernels provided in the Python subroutine sklearn. Figure B1 demonstrates that it yields the predictions of $K_\mathrm{d}(t)$ that fall outside the 95% confidence intervals. While more advanced GP variants may improve the learning performance, their exploration lies outside the scope of this work.

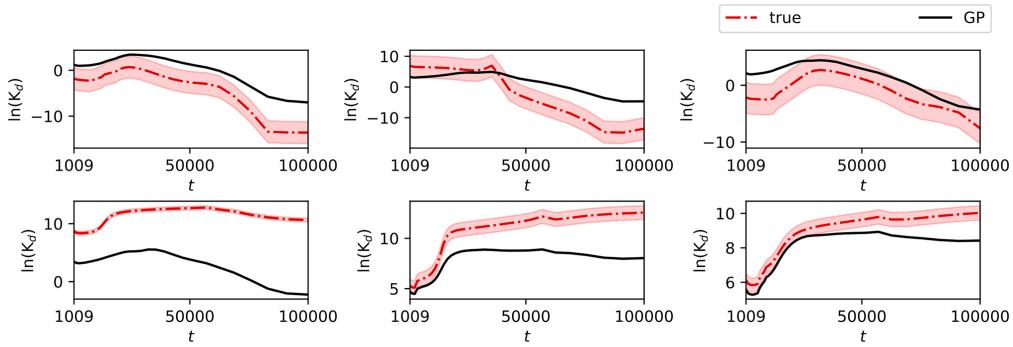

FIG. B1: GP prediction results: optimal radial basis function kernel with correlation length 0.522. The light red region indicates 95% confidence interval.