Hyperparameter Optimization for Randomized Algorithms: A Case Study on Random Features

Oliver R. A. Dunbar ODUNBAR@CALTECH.EDU
Division of Geological and Planetary Sciences
California Institute of Technology
Pasadena, CA 91125, USA

Nicholas H. Nelsen NNELSEN@CALTECH.EDU
Department of Computing and Mathematical Sciences
California Institute of Technology
Pasadena, CA 91125, USA

Maya Mutic MMUTIC@PRINCETON.EDU
Andlinger Center for Energy and the Environment
Princeton University
Princeton, NJ 08554, USA

Abstract

Randomized algorithms exploit stochasticity to reduce computational complexity. One important example is random feature regression (RFR) that accelerates Gaussian process regression (GPR). RFR approximates an unknown function with a random neural network whose hidden weights and biases are sampled from a probability distribution. Only the final output layer is fit to data. In randomized algorithms like RFR, the hyperparameters that characterize the sampling distribution greatly impact performance, yet are not directly accessible from samples. This makes optimization of hyperparameters via standard (gradient-based) optimization tools inapplicable. Inspired by Bayesian ideas from GPR, this paper introduces a random objective function that is tailored for hyperparameter tuning of vector-valued random features. The objective is minimized with ensemble Kalman inversion (EKI). EKI is a gradient-free particle-based optimizer that is scalable to high-dimensions and robust to randomness in objective functions. A numerical study showcases the new black-box methodology to learn hyperparameter distributions in several problems that are sensitive to the hyperparameter selection: two global sensitivity analyses, integrating a chaotic dynamical system, and solving a Bayesian inverse problem from atmospheric dynamics. The success of the proposed EKI-based algorithm for RFR suggests its potential for automated optimization of hyperparameters arising in other randomized algorithms.

Keywords: random features, Gaussian process regression, hyperparameter learning, ensemble Kalman inversion, Bayesian inverse problems

1. Introduction

Nonconvex optimization is ubiquitous in machine learning. In deep learning, it most frequently arises when training a neural network. This involves adjusting the weights and biases of the network to minimize a loss function. Alternatively, one can greatly simplify this optimization task by replacing the inner network weights with fixed samples from a probability distribution (Huang et al., 2006; Scardapane and Wang, 2017) and only learning the neural network's linear output layer. With a suitable choice of loss function, such as a

(possibly regularized) least squares functional, this optimization problem becomes a convex program. In the least squares example, its explicit solution involves the inversion of a linear system. Such an approach reduces approximation power, but does not have to deal with the challenges that come with nonconvexity. A large body of empirical evidence demonstrates that these convex reductions display remarkable performance in streaming large data sets at moderate processing and memory costs during training, partly because they do not require backpropagation (Chattopadhyay et al., 2020; Le et al., 2013; Sun et al., 2015; Bollt, 2021). Performance on (often high-dimensional) problems with little data available for training is less explored (Gauthier et al., 2021; Lanthaler and Nelsen, 2023).

Naturally, there is a hidden price attached to such randomized approximations: one must pose additional structure on the problem in the form of the probability distribution μ from which the neural network weights are sampled. Critically, approximation accuracy is strongly tied to the choice of μ. Unsystematic choice of μ may misrepresent performance or decrease robustness and usability. Indeed, practitioners often resort to unsystematic approaches such as manual tuning, grid searches, or fixed samples that lead to user bias, scalability issues, and overfitting, respectively. One systematic method that the present paper advocates for is to (*i*) choose a u-parametrized family $\{\mu_u\}_{u\in\mathcal{U}}$ of probability distributions, (*ii*) design a regularized stochastic optimization problem for the hyperparameter u, and (*iii*) optimize over $u \in \mathcal{U}$ to achieve consistent performance over different random realizations of the algorithm. Stage (*iii*) will be referred to as *tuning* or *calibration* for the remainder of this paper. The simplicity of the convex optimization of final layer weights is now contrasted by a nonconvex stochastic optimization task for the optimal hyperparameters.

Motivated by scientific and engineering contexts, in this paper we instantiate the preceding systematic optimization pipeline for the specific machine learning task of function emulation from a small noisy data set. We apply the framework to one randomized algorithm, the method of *random features* (RFs, Rahimi and Recht, 2007). We benchmark the RF approach with the current state-of-the-art tool for such tasks, the kernel-based method of *Gaussian processes* (GPs, Williams and Rasmussen, 2006). The underlying hyperparameter tuning problem for the RF method involves optimally adapting the RF sampling distribution μ_u to data. This is equivalent to learning a data-adapted kernel or Bayesian prior distribution. Indeed, RFs, which in special cases can be viewed as neural networks with randomly sampled weights (e.g., "extreme learning machines" from Huang et al. (2006)), by construction can also be viewed as low-rank approximations to GPs. A broad chronology of the developments and performance gains from RF approximation of kernel and GP methods can be found in the work of Liu et al. (2022).

The majority of existing studies do not systematically optimize the RF sampling distribution μ_u. Of those that do, the approach of Yang et al. (2015) is closely related to our own. The authors adopt a GP perspective on RFs by deriving an objective function from empirical Bayes arguments. The work fixes the number and specific realizations of the sampled features. Although this produces accurate results, their approach only fits to the individual sample and not its law, which prohibits future resampling and flexible computational cost adjustments. Other fixed-feature approaches use random projections (Hamid et al., 2014) or discrete kernel alignment (Sinha and Duchi, 2016) to create an optimal weighting of a pre-generated set of RFs. More modern methods learn a generative model for the feature

distribution (Li et al., 2019; Falk et al., 2022); this results in more flexible RF methods. Another line of research derives explicit formulas for the "optimal" RF distribution based on minimization of upper bounds on the approximation error (Kammonen et al., 2020, 2023). The authors then approximately sample from this inaccessible distribution with Markov chain Monte Carlo (MCMC) methods. To summarize, most existing works that do adopt systematic hyperparameter calibration pipelines must exploit special properties and structure of the distribution family $\{\mu_u\}$. In contrast, the present paper introduces a widely applicable hyperparameter learning methodology that treats $\{\mu_u\}$ in a black-box manner.

The optimization workhorse of this paper is the ensemble Kalman inversion (EKI) algorithm (Iglesias et al., 2013; Schillings and Stuart, 2017; Calvello et al., 2025), one of a family of derivative-free algorithms designed to solve nonlinear optimization problems that arise from Bayesian inversion. It can be configured to solve for both a maximum likelihood estimator or a maximum a posteriori estimator. For linear inversion with Gaussian priors and Gaussian observational noise models, the EKI algorithm converges to the true point estimator of the Bayesian posterior distribution. As a particle-based method, EKI can be derived from the ensemble Kalman filtering (EnKF) literature (Chen and Oliver, 2012; Iglesias et al., 2013) and, along with its variants, has been used successfully in a wide range of optimization and inversion tasks (Böttcher, 2023; Cleary et al., 2021; Dunbar et al., 2021, 2022a,b; Huang et al., 2022; Kovachki and Stuart, 2019; Xiao et al., 2016). In part, this success is due to the method's robustness to rough optimization landscapes and the inheritance of modifiers (e.g., localization, sampling error correction, covariance inflation) and variants (e.g., square root inversion, unscented inversion, sparsity-promoting inversion) from the decades-long history of EnKF in meteorological applications. Since EKI treats the hyperparameter-to-output map as a black-box (in particular, without access to derivatives of the map), it allows for easy prototyping and modularity over different families of features, hyperparameters, and objective functions. On the other hand, algorithms that make use of higher-order derivative information tend to converge faster than those that do not. However, the derivative-free EKI algorithm is still known to approximate a gradient descent dynamic and hence acceleration methods such as momentum are readily applicable to EKI (Kovachki and Stuart, 2019, Section 4).

1.1 Contributions

The primary purpose of this paper is to build a robust optimization framework for the automated calibration of hyperparameters that appear in randomized algorithms. The approach developed in this work is derivative-free, which enables a black-box treatment of hyperparameters at the level of probability distributions instead of at the level of individual samples. Highlighting our framework in the specific context of RF algorithms, in this paper we make the following contributions.

(C1) Building off of empirical Bayes, we introduce a stochastic objective function for RF distribution hyperparameters that is specifically tailored to the EKI algorithm.

(C2) We demonstrate that optimized hyperparameters obtained from moderate numbers of EKI iterations are not overfit to specific feature samples. This offers new flexibility such as being able to use a different numbers of features for hyperparameter optimization than for downstream tasks such as prediction.

(C3) We contextualize the proposed machine learning tools with the requirements of common scientific or engineering model emulation tasks. In particular, the following constraints are imposed on the learning tasks:

(i) the amount of available data is severely limited, as these must be generated by scientific computer codes that may be expensive to run;

(ii) inputs and outputs are both multidimensional and correlated, as often such relationships are unknown a priori;

(iii) output observations are corrupted with noise.

This is in contrast to much of the broader machine learning literature, which is more concerned with scaling performance on massive data sets and more reliant on approaches such as exact diagonalization to tackle multi-output problems.

(C4) We test the new machine learning tools on three scientifically-minded benchmark problems. The tools are used to emulate scalar-valued functions for global sensitivity analysis, integrate the state a chaotic Lorenz 63 dynamical system, and accelerate a Bayesian parameter estimation problem from the atmospheric sciences by emulating the underlying forward map. In all three applications, we compare our approach with a current gold-standard GP-based methodology.

(C5) We document and actively maintain all tools and examples in open-source, registered, Julia-language GitHub repositories (Dunbar et al., 2022b, 2024) with documentation at:

(i) (EKI) `clima.github.io/EnsembleKalmanProcesses.jl/dev/`,

(ii) (RF) `clima.github.io/RandomFeatures.jl/dev/`,

(iii) (Examples) `clima.github.io/CalibrateEmulateSample.jl/dev/`.

1.2 Outline

The remainder of this paper is organized as follows. Section 2 introduces Bayesian regression with RFs from a GP perspective for both scalar-valued and vector-valued function learning settings. Section 3 motivates the RF hyperparameter learning problem from empirical Bayes ideas. It also presents the black-box EKI optimizer that is championed in this work and a random objective function that is amenable to EKI. Section 4 applies the tuned RF emulators to three scientific applications. The paper concludes with a final discussion in Section 5. Appendix A contains additional details regarding the numerical experiments from Section 4.

2. Bayesian Regression with Random Features

In this section, we formulate a supervised learning problem in the framework of RFs. To this end, let $\mathcal{X} \subseteq \mathbb{R}^d$ denote the input space and $\mathcal{Y}$ denote the output space. Suppose that we have access to N pairs of data $\{(x_n, y_n)\}_{n=1}^N \subset \mathcal{X} \times \mathcal{Y}$. To set the notation, let $X \coloneqq \{x_n\}_{n=1}^N$, $Y \coloneqq \{y_n\}_{n=1}^N$, and $D_N \coloneqq (X, Y)$. Write $[N] \coloneqq \{1, 2, \ldots, N\}$. Although our

framework can handle both random or deterministic input data X, from here onward we assume that X is fixed to ease notation and simplify the exposition. The goal is to train a model $F\colon \mathcal{X} \to \mathcal{Y}$ to both "fit" the observed data D_N and generalize well, that is, $F(x) \approx y$ in an appropriate sense both for $x \in X$ and $x \notin X$. With only finite data and without further assumptions on how X is related to Y, the learning problem is ill-posed. Learning algorithms impose prior knowledge, regularization, or constraints to obtain a stable solution. Subsection 2.1 first presents the *Gaussian process regression* (GPR) solution approach and then its approximation by RFs, *random feature regression* (RFR), in a scalar output setting, that is, $\mathcal{Y} = \mathbb{R}$. This setup is then generalized in Subsection 2.2 to the more realistic setting of vector-valued learning, where now $\mathcal{Y} = \mathbb{R}^p$ and output space correlations must be accounted for. Extensions to infinite-dimensional $\mathcal{X}$ and $\mathcal{Y}$ are also possible (Lanthaler and Nelsen, 2023; Nelsen and Stuart, 2021) but are not considered here.

2.1 Scalar-Valued Learning

We now consider the output space $\mathcal{Y} = \mathbb{R}$ and thus real-valued regression. This is a classical problem and various estimators exist for it, ranging from those based on parametric models or nonparametric models to those that return a single point solution or an entire family of possible solutions in the form of a probability distribution. In this paper, we focus on the latter class of so called Bayesian estimators. GPR is one canonical example from this class.

2.1.1 Gaussian Process Regression

We begin by prescribing the following statistical model for the data D_N:

$$y_n = F(x_n) + \eta_n, \quad \text{where} \quad \eta_n \overset{\text{i.i.d.}}{\sim} \mathsf{N}(0, \sigma^2) \quad \text{for all} \quad n \in [N]. \tag{1}$$

Here $E \coloneqq \{\eta_n\}_{n=1}^N$ represents noisy observations in the form of independent and identically distributed (i.i.d.) real-valued zero mean Gaussian random variables with common variance σ^2, and F determines the input-output relationship. The ideal situation is the well-specified setting[1] in which the observed D_N is actually generated according to (1) for some ground truth function $F = F^\star$.

GPR proceeds by imposing additional structure on (1) in the form of a prior probability distribution over the function F. That is, we model F as a GP, independent of E, given by

$$F \sim \mathsf{GP}\big(\overline{F}, K\big). \tag{2}$$

This means that $F\colon \mathcal{X} \to \mathbb{R}$ is a *random function* with mean function $\overline{F}\colon \mathcal{X} \to \mathbb{R}$ and symmetric covariance function $K\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that for any $T \in \mathbb{N}$ and any distinct points $Z \coloneqq \{z_t\}_{t=1}^T \subset \mathcal{X}$, the finite-dimensional marginals $F(Z) \coloneqq (F(z_1), F(z_2), \ldots, F(z_T))^\top \in \mathbb{R}^T$ are distributed according to

$$F(Z) \sim \mathsf{N}\big(\overline{F}(Z), K(Z,Z)\big), \tag{3}$$

where the mean vector and positive semidefinite covariance matrix of the multivariate Gaussian distribution in (3) are given by

1. The observed data D_N may not necessarily follow the likelihood model (1) in reality. For example the noise may not be i.i.d. Gaussian or may appear multiplicatively. From this perspective, $\sigma > 0$ may be viewed as a tunable hyperparameter of the GPR algorithm and not fixed in advance by the data.

$$\overline{F}(Z) \coloneqq \big(\overline{F}(z_1), \ldots, \overline{F}(z_T)\big)^\top \in \mathbb{R}^T \quad \text{and} \quad K(Z,Z) \coloneqq \big[K(z_i, z_j)\big]_{i,j\in[N]} \in \mathbb{R}^{T\times T}. \tag{4}$$

Typically the prior mean $\overline{F}$ is set to zero. The choice of covariance function K greatly influences prediction accuracy and is informed by prior knowledge about the problem, such as smoothness, sparsity, and lengthscales.

To solve the original regression problem, GPR simply applies Bayes' rule by conditioning the prior function F on the observed data D_N to obtain the *posterior distribution*

$$F \,|\, D_N \sim \mathsf{GP}\big(\overline{F}^{(N)}, K^{(N)}\big), \tag{5}$$

where the posterior mean function and posterior covariance function are given by[2]

$$\begin{aligned} \overline{F}^{(N)}(x) &= \overline{F}(x) + K(x,X)\big(K(X,X) + \sigma^2 I_N\big)^{-1}\big(Y - \overline{F}(X)\big) \quad \text{and} \\ K^{(N)}(x,x') &= K(x,x') - K(x,X)\big(K(X,X) + \sigma^2 I_N\big)^{-1} K(X,x') \end{aligned} \tag{6}$$

for any $x \in \mathcal{X}$ and $x' \in \mathcal{X}$ (Williams and Rasmussen, 2006). Here $I_N \in \mathbb{R}^{N\times N}$ represents the N-dimensional identity and

$$K(X,x) \coloneqq \big(K(x_1,x), \ldots, K(x_N,x)\big)^\top \in \mathbb{R}^N \tag{7}$$

is the vector of cross covariances between the training inputs and point $x \in \mathcal{X}$. We identify $K(x,X) \coloneqq K(X,x)^\top \in \mathbb{R}^{1\times N}$ as a row vector. The whole posterior distribution (5) itself is the GPR estimator. The posterior mean is a standard point predictor and can be related to deterministic kernel methods. The posterior covariance enables the quantification of uncertainty in the prediction and beyond. The underlying regression problem is actually regularized by the prescription of the prior distribution over F in the sense that the Bayesian inversion $D_N \mapsto F \,|\, D_N$ is a stable mapping (Stuart, 2010). The posterior also being Gaussian is due to the fact that both the likelihood (1) and prior (2) were assumed Gaussian.[3]

The closed form expressions for the posterior mean and covariance functions in (6) are what standard GPR software packages implement in practice. For $x \in \mathcal{X}$, these implementations involve solving the linear systems

$$\big(K(X,X) + \sigma^2 I_N\big)\alpha = \big(Y - \overline{F}(X)\big) \quad \text{and} \quad \big(K(X,X) + \sigma^2 I_N\big)\alpha(x) = K(X,x) \tag{8}$$

for coefficients $\alpha \in \mathbb{R}^N$ and $\alpha(x) \in \mathbb{R}^N$, respectively. We observe here that the linear algebraic complexity falls into three stages: *space* (i.e., memory storage), *offline* (i.e., operations on stored objects), and *online* (i.e., operations for evaluation at new $x \in \mathcal{X}$). GPR is a nonparametric method that requires access to the full input data X to evaluate the posterior mean and covariance at online cost $\mathcal{O}(N)$. Storing X requires $\mathcal{O}(dN)$ memory. Since

2. The posterior mean formula here is simply *kernel ridge regression* (Kanagawa et al., 2018) with a specific ridge penalty parameter depending on σ^2 and N.
3. One can also perform GPR with non-Gaussian likelihoods. However, the posterior will no longer be Gaussian in this case and the closed form expressions (6) for the mean and covariance will no longer be valid. More sophisticated approaches to compute the posterior, such as MCMC, would be required.

$(K(X,X)+\sigma^2 I_N)$ is a dense matrix in general, it requires $\mathcal{O}(N^2)$ storage in space. Often applications require the solution of the linear systems (8) many times with the same system matrix. In this case, we use a Cholesky factorization that has an offline cost of $\mathcal{O}(N^3)$ operations. Then the cost of obtaining α in (8) is reduced to $\mathcal{O}(N^2)$. Similarly, for a new input $x \in \mathcal{X}$, we can use the precomputed Cholesky factor to solve for $\alpha(x)$ with cost $\mathcal{O}(N^2)$. Nevertheless, for problems with large N, such costs make GPR infeasible as presented. This motivates randomized approximations that reduce computational complexity.

2.1.2 Random Feature Regression

Several scalable approximations to GPR have been proposed in the literature. These include, but are not limited to, sparse variational approaches based on inducing points (Hensman et al., 2015, 2018; Titsias, 2009), Nyström subsampling (Sun et al., 2015), and RFs (Rahimi and Recht, 2007). We focus on RFs and in particular the method of RFR. We interpret RFR as an approximation to GPR obtained by first replacing the GP prior covariance K with a low-rank approximation K_M and then performing exact GPR inference (5) with this low-rank covariance function. We now describe this procedure in detail.

To begin, let Θ be a set and $\varphi\colon \mathcal{X}\times\Theta\to\mathbb{R}$ be a feature map. Let μ be a probability distribution over Θ. The choice of pair (φ,μ) defines the particular RF method. From this pair, define a GP prior covariance function $K\colon \mathcal{X}\times\mathcal{X}\to\mathbb{R}$ for x and x' in $\mathcal{X}$ by

$$K(x,x') \coloneqq \mathbb{E}_{\theta\sim\mu}\big[\varphi(x;\theta)\varphi(x';\theta)\big]. \tag{9}$$

A covariance defined this way is always symmetric and positive semidefinite by construction. Conversely, any given GP covariance function K can be written in the form (9) for some set Θ and RF pair (φ,μ).[4] In practice, many popular covariance functions fit such a form (Rudi and Rosasco, 2017, Supplement E). The functions $\varphi(\,\cdot\,;\theta)$ with $\theta\sim\mu$ are also called RFs and play a role similar to that of feature maps in the kernel methods literature.

So far, the kernel K is completely deterministic. RF methods apply a Monte Carlo approximation to the integral in (9), replacing the full expectation with the empirical average

$$K_M(x,x') \coloneqq \frac{1}{M}\sum_{m=1}^{M}\varphi(x;\theta_m)\varphi(x';\theta_m), \quad \text{where} \quad \theta_m \overset{\text{i.i.d.}}{\sim} \mu, \tag{10}$$

for $x\in\mathcal{X}$ and $x'\in\mathcal{X}$ and some $M\in\mathbb{N}$. This is a random finite rank prior covariance function. RFR is then simply the process of applying Bayes' rule to the RF prior $\mathsf{GP}(\overline{F},K_M)$ under the likelihood model (1) to obtain the RF posterior $\mathsf{GP}(\overline{F}_M^{(N)},K_M^{(N)})$. Here, the RF posterior mean $\overline{F}_M^{(N)}$ and RF posterior covariance $K_M^{(N)}$ are given by (6) except with all instances of K replaced by K_M. In terms of sample paths, we write $F_M\sim\mathsf{GP}(\overline{F},K_M)$ for the RF prior and $F_M\,|\,D_N\sim\mathsf{GP}(\overline{F}_M^{(N)},K_M^{(N)})$ for the RF posterior. Although it appears that RFR is just GPR with a special choice of prior, from an implementation and practical point of view it is provably beneficial to view RFR as an entirely distinct methodology (Rahimi and Recht, 2008a,b). Indeed, RFR can be interpreted as a finite rank Bayesian linear

4. Indeed, by definition $\mathsf{GP}(0,K)$ satisfies $K(x,x')=\mathbb{E}_{F\sim\mathsf{GP}(0,K)}[F(x)F(x')]$. To satisfy (9), take Θ to be a sufficiently smooth function space, $\mu=\mathsf{GP}(0,K)$, and $\varphi(x;\theta)=F(x)$ with $\theta=F\sim\mu$.

model in weight space (Bishop and Nasrabadi, 2006, Section 3.3), which simplifies sampling of the posterior distribution. Moreover, in what follows we show that RFR involves different system matrices that are cheaper to invert and coefficients that have different interpretations than those in appearing in GPR. We leave a rigorous analysis of the convergence of RFR to GPR to future work.

We now follow through the linear algebra to define the system matrices that characterize the posterior mean and covariance of RFR. To this end, we adopt the notation

$$\begin{aligned} \Phi_M(x) &:= \big(\varphi(x;\theta_1),\dots,\varphi(x;\theta_M)\big) \in \mathbb{R}^{1\times M} \quad \text{and} \\ \Phi_M(X) &:= \big[\varphi(x_n;\theta_m)\big]_{n\in[N],m\in[M]} \in \mathbb{R}^{N\times M} \end{aligned} \tag{11}$$

for $x \in \mathcal{X}$. Thus $K_M(x,x') = \Phi_M(x)\Phi_M(x')^\top/M$. We observe from (6) and (8) that $\overline{F}_M^{(N)}(x) = \overline{F}(x) + K_M(x,X)\alpha = \overline{F}(x) + M^{-1}\Phi_M(x)\beta$, where

$$\beta := \Phi_M(X)^\top\alpha \in \mathbb{R}^M. \tag{12}$$

This shows that the difference between the posterior mean and the prior mean belongs to the linear span of the RFs $\{\varphi(\,\cdot\,;\theta_m)\}_{m=1}^M$. We see that the random neural network example from Section 1 holds whenever the RFs $\varphi(\,\cdot\,;\theta)$ take the form of a hidden neuron with sampled weights $\theta \sim \mu$ (or combinations thereof). The coefficients β represent the final linear output layer weights in this analogy. It remains to derive a computationally tractable equation for β. To do this, left multiply the first equation in (8) by $\Phi_M(X)^\top$ to obtain

$$\left(\frac{1}{M}\Phi_M(X)^\top\Phi_M(X) + \sigma^2 I_M\right)\beta = \Phi_M(X)^\top\big(Y - \overline{F}(X)\big). \tag{13}$$

Thus, obtaining β only requires the inversion of an $M \times M$ matrix instead of an $N \times N$ one. This is advantageous whenever $M \ll N$.

We similarly seek to write the RF posterior covariance function in the span of the RFs. First, we project the second equation in (8) to obtain the parametrized $M \times M$ system

$$\left(\frac{1}{M}\Phi_M(X)^\top\Phi_M(X) + \sigma^2 I_M\right)\beta(x') = \left(\frac{1}{M}\Phi_M(X)^\top\Phi_M(X)\right)\Phi_M(x')^\top. \tag{14}$$

for $\beta(x') := \Phi_M(X)^\top\alpha(x') \in \mathbb{R}^M$. From the second equation in (6), we deduce that $K_M^{(N)}(x,x') = K_M(x,x') - M^{-1}\Phi_M(x)\beta(x') = M^{-1}\Phi_M(x)(\Phi_M(x')^\top - \beta(x'))$. However, realizing that the system matrix on the left hand side of (14) is actually the precision matrix of the RF weight space posterior (Bishop and Nasrabadi, 2006, Section 3.3) reveals an even more efficient form of $K_M^{(N)}$. This form requires the solution $\widehat{\beta}(x')$ of

$$\left(\frac{1}{M}\Phi_M(X)^\top\Phi_M(X) + \sigma^2 I_M\right)\widehat{\beta}(x') = \Phi_M(x')^\top \tag{15}$$

and use of the matrix identity $I - (A+\sigma^2 I)^{-1}A = \sigma^2(A+\sigma^2 I)^{-1}$ for symmetric positive semidefinite A. We summarize the closed form formulas for the RF posterior mean and covariance functions in terms of the coefficients β from (13) and $\widehat{\beta}(x')$ from (15) as

$$\overline{F}_M^{(N)}(x) = \overline{F}(x) + \frac{1}{M}\Phi_M(x)\beta = \overline{F}(x) + \frac{1}{M}\sum_{m=1}^{M}\beta_m\varphi(x;\theta_m) \quad \text{and}$$
$$K_M^{(N)}(x,x') = \frac{\sigma^2}{M}\Phi_M(x)\widehat{\beta}(x') = \frac{\sigma^2}{M}\sum_{m=1}^{M}\big(\widehat{\beta}(x')\big)_m\,\varphi(x;\theta_m) \tag{16}$$

for any x and x' in $\mathcal{X}$.

We now assess the computational complexity for RFR. Since RFR is a parametric method of estimation, we can discard the data X from memory once the dense matrix $\Phi_M(X) \in \mathbb{R}^{N\times M}$ is stored with cost $\mathcal{O}(NM)$. The new system matrix in (13) leads to offline complexity $\mathcal{O}(M^3)$ for a Cholesky factorization. Subsequent solves for β in (13) offline and $\widehat{\beta}(x)$ in (14) online both have complexity $\mathcal{O}(M^2)$. The online cost to evaluate the RF posterior mean and covariance functions is $\mathcal{O}(M)$. Thus, RFR is computationally advantageous to GPR if the RF regression error is comparable to that of GPR even with $M \ll N$. This is indeed true. Theoretical analysis establishes that RFR delivers the same asymptotic error rate as GPR with only $M = \mathcal{O}(\sqrt{N})$ RFs (Lanthaler and Nelsen, 2023).

We conclude this subsection with an example of a commonly used RF pair (φ, μ).

Example 1 (Random Fourier Features for RBF Kernel) *The pair* (φ, μ) *with*

$$x \mapsto \varphi(x;\theta) \coloneqq \sqrt{2}\cos(a^\top x + b) \quad \textit{and} \quad \theta = (a,b) \sim \mu \coloneqq \mathsf{N}(0,\ell^{-2}I) \otimes \mathsf{Unif}(-\pi,\pi) \tag{17}$$

satisfies the identity (9) *for the squared exponential covariance function*

$$(x,x') \mapsto K(x,x') = \exp\left(-\frac{\|x-x'\|_2^2}{2\ell^2}\right). \tag{18}$$

The lengthscale $\ell > 0$ *is the only hyperparameter appearing in this kernel.*

More details about random Fourier features are provided in Section 3.4.

2.2 Vector-Valued Learning

In this subsection, we summarize the Bayesian regression framework in the setting of vector-valued outputs. Let $\mathcal{Y} = \mathbb{R}^p$ for some $p \in \mathbb{N}$. We prescribe the likelihood model

$$y_n = F(x_n) + \eta_n, \quad \text{where} \quad \eta_n \overset{\text{i.i.d.}}{\sim} \mathsf{N}(0,\Sigma) \quad \text{for all} \quad n \in [N]. \tag{19}$$

Here $\Sigma \in \mathbb{R}^{p\times p}$ is the symmetric positive definite covariance matrix of the Gaussian noise. It is common to model the noise by $\Sigma = \sigma^2 I_p$, but non-isotropic covariances are also natural depending on the application. The function F in (19) maps $\mathcal{X}$ to $\mathbb{R}^p$. Independent GP or RF priors on F lead to GP or RF posteriors, respectively. We first describe the GP setting before moving on to the computationally advantageous RF setting.

2.2.1 Vector-Valued Gaussian Process Regression

We model F with an $\mathbb{R}^p$-valued GP prior $F \sim \mathsf{GP}(\overline{F}, K)$. The mean function $\overline{F}\colon \mathcal{X} \to \mathbb{R}^p$ is vector-valued. More importantly, the covariance function $K\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}^{p\times p}$ is matrix-valued in order to model correlations in the output space. Several examples of matrix- or operator-valued covariance kernels may be found in the work of Brault et al. (2016); Kadri et al. (2016); Micchelli and Pontil (2004, 2005); Nelsen and Stuart (2021).

Under this $\mathbb{R}^p$-valued GP prior, the resulting posterior $F \,|\, D_N \sim \mathsf{GP}(\overline{F}^{(N)}, K^{(N)})$ under the likelihood (19) is also an $\mathbb{R}^p$-valued GP with mean and covariance functions

$$\begin{aligned} \overline{F}^{(N)}(x) &= \overline{F}(x) + K(x,X)\big(K(X,X) + B_\Sigma\big)^{-1}\big(Y - \overline{F}(X)\big) \quad \text{and} \\ K^{(N)}(x,x') &= K(x,x') - K(x,X)\big(K(X,X) + B_\Sigma\big)^{-1}K(X,x') \end{aligned} \tag{20}$$

for any $x \in \mathcal{X}$ and $x' \in \mathcal{X}$. These equations are interpreted in block form. In particular, $B_\Sigma \coloneqq \operatorname{blockdiag}(\Sigma, \ldots, \Sigma) \in (\mathbb{R}^{p\times p})^{N\times N}$, $K(X,X) \coloneqq \big[K(x_i,x_j)\big]_{i,j\in[N]} \in (\mathbb{R}^{p\times p})^{N\times N}$, and

$$K(X,x') \coloneqq \big(K(x_1,x'), \ldots, K(x_N,x')\big)^\top \in \big(\mathbb{R}^{p\times p}\big)^N. \tag{21}$$

The factor $K(x,X) = K(X,x)^\top \in (\mathbb{R}^{p\times p})^{1\times N}$ is interpreted as a "row vector" with matrix entries. The concatenated data $Y = \{y_n\}_{n=1}^N$ and prior mean entries $\overline{F}(X)$ are viewed as length N vectors with each entry taking value in $\mathbb{R}^p$. The formulas (20) follow from whitening (19) by pre-multiplying by $\Sigma^{-1/2}$ and then applying an existing white noise result (Owhadi, 2022, Theorem 5.3).

Although (20) visually mimics its scalar-valued counterpart (6), the computational effort is greatly magnified in the vector-valued setting. Effectively, the sample size N is replaced by Np in all scalar-valued output space GPR complexity estimates. Indeed, (20) requires solving Np by Np linear systems of equations instead of N by N ones. Table 1 summarizes upper bounds on the space, offline, and online costs of the approach. This poor scaling with output space dimension p has severely limited the practical utility of vector-valued GPR to date. Indeed, existing studies usually only work with separable (or even scalar) matrix-valued kernels, which suffer from a simplistic rank-one tensor structure (Kadri et al., 2016; Owhadi, 2022), or diagonal kernels, which ignore correlations in the output space (Cleary et al., 2021). Vector-valued RFs enable scalable approximate GP inference without neglecting correlations in the output space.

2.2.2 Vector-Valued Random Feature Regression

Vector-valued RFR greatly alleviates the cost of learning with matrix-valued covariance kernels. The method is characterized by a vector-valued feature map $\varphi\colon \mathcal{X} \times \Theta \to \mathbb{R}^p$ and a probability distribution μ over Θ. We work with RF kernels of the form

$$K(x,x') \coloneqq \mathbb{E}_{\theta\sim\mu}\Big[\varphi(x;\theta)\varphi(x';\theta)^\top\Big] \quad \text{and} \quad K_M(x,x') \coloneqq \frac{1}{M}\sum_{m=1}^M \varphi(x;\theta_m)\varphi(x';\theta_m)^\top, \tag{22}$$

where $\theta_m \overset{\text{i.i.d.}}{\sim} \mu$ for $m \in [M]$ and x and x' belong to $\mathcal{X}$. The vector-valued RFR method proceeds by assigning a RF prior distribution $F_M \sim \mathsf{GP}(\overline{F}, K_M)$ to F in (19). The RF posterior is then given by the vector-valued Gaussian process $\mathsf{GP}(\overline{F}_M^{(N)}, K_M^{(N)})$, where

$$\begin{aligned} \overline{F}_M^{(N)}(x) &= \overline{F}(x) + \frac{1}{M}\sum_{m=1}^{M} \beta_m \varphi(x;\theta_m) \quad \text{and} \\ K_M^{(N)}(x,x') &= \frac{1}{M}\sum_{m=1}^{M} \varphi(x;\theta_m)\big(\widehat{\beta}(x')\big)_m \end{aligned} \tag{23}$$

for any x and x' in $\mathcal{X}$. The coefficients $\beta \in \mathbb{R}^M$ and $\widehat{\beta}(x') \in \mathbb{R}^{M\times p}$ solve the linear systems

$$\begin{aligned} \left(\frac{1}{M}\Phi_M(X)^\top B_\Sigma^{-1}\Phi_M(X) + I_M\right)\beta &= \Phi_M(X)^\top B_\Sigma^{-1}\big(Y - \overline{F}(X)\big) \quad \text{and} \\ \left(\frac{1}{M}\Phi_M(X)^\top B_\Sigma^{-1}\Phi_M(X) + I_M\right)\widehat{\beta}(x') &= \Phi_M(x')^\top, \end{aligned} \tag{24}$$

respectively. The term $\Phi_M(x') \in \mathbb{R}^{p\times M}$ is analogous to the first line of (11). The formulas in (24) are derived using an approach similar to that in Subsection 2.1.2, except now with an additional left multiplication by $B_\Sigma^{-1/2}$. More precisely, we define (24) by

$$\begin{aligned} \sum_{m=1}^{M}\left(\frac{1}{M}\sum_{n=1}^{N}\varphi(x_n;\theta_\ell)^\top\Sigma^{-1}\varphi(x_n;\theta_m) + \delta_{\ell m}\right)\beta_m &= \sum_{n=1}^{N}\varphi(x_n;\theta_\ell)^\top\Sigma^{-1}\big(y_n - \overline{F}(x_n)\big) \quad \text{and} \\ \sum_{m=1}^{M}\left(\frac{1}{M}\sum_{n=1}^{N}\varphi(x_n;\theta_\ell)^\top\Sigma^{-1}\varphi(x_n;\theta_m) + \delta_{\ell m}\right)\big(\widehat{\beta}(x')\big)_m &= \varphi(x';\theta_\ell)^\top, \end{aligned} \tag{25}$$

where $\ell \in [M]$ and $\delta_{\ell m} = 1$ if $\ell = m$ and equals zero otherwise. The Σ-weighted system Gram matrices in (25) are still only of size M by M. These are much cheaper to invert than the vector-valued GPR system matrices whenever $M \ll Np$; this is often the case in practice. Table 1 summarizes the computational tradeoffs. We observe that the complexity of vector-valued GPR depends directly on Np, with cubic scaling in Np offline and quadratic scaling online for storage. On the other hand, the complexity of RFR depends instead on M instead of Np, with similar scaling in this variable. Thus, RFR is computationally advantageous whenever $M \ll Np$, while still maintaining the same average prediction accuracy as vector-valued GPR (Lanthaler and Nelsen, 2023).

3. Hyperparameter Learning for Random Feature Regression

Recall that in the development of Section 2, the RF pair (φ, μ) is assumed given. An often overlooked practical consideration of RFR—or other algorithms based on RFs—is how to choose the pair (φ, μ). This is the natural analog to choosing the covariance kernel K in GPR. To address this kernel selection problem, we begin in Subsection 3.1 by adapting the popular empirical Bayes approach for learning priors in GPR to the stochastic setting of RFR. This approach comes with several unique challenges that we address with the derivative-free optimization method known as EKI. We describe the EKI algorithm in Subsection 3.2. In Subsection 3.3, we cast the covariance kernel learning problem as a statistical

Table 1: Computational complexity of important operations in GPR and RFR. The integers N, M, d, and p denote the number of data points, number of random features, dimension of $\mathcal{X}$, and dimension of $\mathcal{Y}$, respectively. "Space" represents a memory cost, "Offline" represents a one time precomputation, and "Online" represents a cost at every new input x and x'.

Operation	Cost Type	GPR	RFR
Store dense matrix	Space	$\mathcal{O}\big((Np)^2\big)$	$\mathcal{O}\big(M^2\big)$
Cholesky factorization	Offline	$\mathcal{O}\big((Np)^3\big)$	$\mathcal{O}\big(M^3\big)$
Evaluate $\overline{F}^{(N)}(x)$	Online	$\mathcal{O}\big((Np)pd\big)$	$\mathcal{O}\big(Mpd\big)$
Evaluate $\overline{K}^{(N)}(x,x')$	Online	$\mathcal{O}\big(Np(Np+pd)\big)$	$\mathcal{O}\big(Mp(M+p+d)\big)$

inverse problem so that it may be solved with EKI. Subsection 3.4 instantiates the methodology with an expressive family of parametrized distributions $\{\mu_u\}$ that also performs well in practice. Subsection 3.5 acknowledges the limitations of the proposed framework.

3.1 Empirical Bayes Motivation

Suppose that $\{\mu_u\}_{u\in\mathcal{U}}$ is a parametrized family of RF distributions over some hyperparameter set $\mathcal{U}$. Our framework is still valid if φ is also a parametrized family, but for simplicity we assume that the feature map φ is fixed. Let $K^{(u)}\colon \mathcal{X}\times\mathcal{X}\to\mathbb{R}^{p\times p}$ be the matrix-valued kernel defined by

$$K^{(u)}(x,x') \coloneqq \mathbb{E}_{\theta\sim\mu_u}\Big[\varphi(x;\theta)\varphi(x';\theta)^\top\Big] \tag{26}$$

for x and x' in $\mathcal{X}$ and parametrized by $u\in\mathcal{U}$. Under the vector-valued likelihood (19), a fully Bayesian approach to performing GPR with the kernel (26) is to build a hierarchical prior model by first assigning a prior distribution to the hyperparameter u and then taking the distribution of $F\,|\,u$ to be a centered Gaussian process with covariance function $K^{(u)}$. This defines a prior distribution for the joint vector (F,u). The posterior is then obtained as usual by conditioning (F,u) on the observed data Y. In general, however, this posterior is non-Gaussian and requires expensive sampling algorithms to access it.

To overcome this, the empirical Bayes methodology takes an optimization approach (Chen et al., 2021; Naslidnyk et al., 2023). Assuming that u is finite-dimensional and the prior on u is uninformative, such as a uniform distribution, empirical Bayes maximizes the probability density of $u\,|\,Y$, which is the joint posterior $(F,u)\,|\,Y$ marginalized over F. By Bayes' rule, the density of $u\,|\,Y$ is proportional to the marginal likelihood $Y\,|\,u$. Under our Gaussian data likelihood (19), $Y\,|\,u\sim\mathsf{N}(0,K^{(u)}(X,X)+B_\Sigma)$ so that empirical Bayes minimizes the negative log marginal likelihood

$$L^{(\mathrm{EB})}(u) \coloneqq Y^\top\big(K^{(u)}(X,X)+B_\Sigma\big)^{-1}Y + \log\det\big(K^{(u)}(X,X)+B_\Sigma\big) \tag{27}$$

over $u\in\mathcal{U}$. This inspires an empirical RF approximation to the true covariance matrices appearing in (27). Let $K_M^{(u)}$ be as in (22) with $\mu=\mu_u$. Then define the function

$$\widetilde{L}_M^{(\mathrm{EB})}(u) \coloneqq Y^\top\big(K_M^{(u)}(X,X)+B_\Sigma\big)^{-1}Y + \log\det\big(K_M^{(u)}(X,X)+B_\Sigma\big). \tag{28}$$

For sufficiently large M, we expect $\widetilde{L}_M^{(\mathrm{EB})}$ to be close to $L^{(\mathrm{EB})}$ in some reasonable sense.

Unfortunately, (28) as written is not efficiently computable. We would like to write it in terms of the smaller RF gram matrix $G_M^{(u)} \in \mathbb{R}^{M\times M}$ defined by

$$G_M^{(u)} \coloneqq \left[\frac{1}{M}\sum_{n=1}^{N} \varphi\big(x_n;\theta_\ell^{(u)}\big)^\top \Sigma^{-1} \varphi\big(x_n;\theta_m^{(u)}\big)\right]_{\ell\in[M],m\in[M]}, \tag{29}$$

where $\theta_m^{(u)} \sim \mu_u$ are i.i.d. samples. To do this, notice that

$$\begin{aligned}\det\big(K_M^{(u)}(X,X)+B_\Sigma\big) &= \big(\det B_\Sigma^{1/2}\big)^2 \det\big(B_\Sigma^{-1/2}K_M^{(u)}(X,X)B_\Sigma^{-1/2}+I_{Np}\big) &(30)\\ &= (\det\Sigma)^N \det\Big(\frac{1}{M}B_\Sigma^{-1/2}\Phi_M^{(u)}(X)\Phi_M^{(u)}(X)^\top B_\Sigma^{-1/2}+I_{Np}\Big). &(31)\end{aligned}$$

This and application of the Weinstein–Aronszajn identity $\det(AB+I)=\det(BA+I)$ for AB and BA square and I of appropriate dimension yield

$$\log\det\big(K_M^{(u)}(X,X)+B_\Sigma\big) = N\log\det\Sigma + \log\det\big(G_M^{(u)}+I_M\big). \tag{32}$$

To simplify the first term in (28), let $\alpha^{(u)} \coloneqq (K_M^{(u)}(X,X)+B_\Sigma)^{-1}Y \in (\mathbb{R}^p)^N$. Then

$$Y^\top\big(K_M^{(u)}(X,X)+B_\Sigma\big)^{-1}Y = (\alpha^{(u)})^\top K_M^{(u)}(X,X)\alpha^{(u)} + (\alpha^{(u)})^\top B_\Sigma\alpha^{(u)}. \tag{33}$$

The calculations in Subsection 2.2.2 show that the RFR posterior mean coefficients $\beta = \beta^{(u)} \in \mathbb{R}^M$ from (25) satisfy $\beta_m^{(u)} = \sum_{n=1}^N \varphi(x_n;\theta_m^{(u)})^\top \alpha_n^{(u)}$. This implies that the first term on the right hand side of (33) equals $\|\beta^{(u)}\|_{\mathbb{R}^M}^2/M$. The second term may be written as $(\alpha^{(u)})^\top B_\Sigma\ B_\Sigma^{-1}\ B_\Sigma\ \alpha^{(u)} = \|B_\Sigma^{-1/2}(B_\Sigma\alpha^{(u)})\|_{(\mathbb{R}^p)^N}^2$. Using the fact that $B_\Sigma\alpha^{(u)} = Y - K_M^{(u)}(X,X)\alpha^{(u)}$, we thus define our practically computable random empirical Bayes objective function to be $L_M^{(\mathrm{EB})}(u) \coloneqq \widetilde{L}_M^{(\mathrm{EB})} - N\log\det(\Sigma)$, which equals

$$L_M^{(\mathrm{EB})}(u) = \frac{1}{M}\big\|\beta^{(u)}\big\|_{\mathbb{R}^M}^2 + \Big\|B_\Sigma^{-1/2}\big(Y-\overline{F}_M^{(N)}(X;u)\big)\Big\|_{(\mathbb{R}^p)^N}^2 + \log\det\big(G_M^{(u)}+I_M\big). \tag{34}$$

We subtracted off the term $N\log\det(\Sigma)$ (32) because it is independent of u. The map $\overline{F}_M^{(N)}(\,\cdot\,;u) = K_M^{(u)}(\,\cdot\,,X)\alpha^{(u)}$ in (34) is the RF posterior mean corresponding to feature pair (φ,μ_u).

Although now written in terms of efficiently computable quantities, the actual optimization of the objective function $L_M^{(\mathrm{EB})}$ is non-trivial. This is because we view the map $u\mapsto\mu_u$ as a black-box and hence each evaluation of the objective $L_M^{(\mathrm{EB})}(u)$ for new u requires resampling the M random variables $\{\theta_m^{(u)}\}$ i.i.d. from μ_u. Although one could attempt to reduce the effect of randomness by drawing enormous numbers of RFs, this will likely require $M \gg Np$ and be computationally infeasible to optimize. Instead, our approach is to use a statistical optimization tool which can handle stochastic model evaluations more naturally.

3.2 Ensemble Kalman Inversion

EKI (Iglesias et al., 2013; Schillings and Stuart, 2017; Calvello et al., 2025)—and its variants (Huang et al., 2022; Garbuno-Inigo et al., 2020a,b)—is a family of tools for stochastic optimization. More explicitly, EKI is a method for solving Bayesian inverse problems. In the following exposition, we first describe the basics of the algorithm to solve such problems. Then the following subsection proposes a new inverse problem whose solution is related to minimization of (34).

The underlying generic inverse problem setting is as follows. Assume a system produces observable output under a parameter-to-observable map $\mathcal{G}$ that is random with Gaussian marginals. That is, the relationship between the parameters u and observable z is given by

$$z = \mathcal{G}(u), \quad \text{where} \quad \mathcal{G}(u) \sim \mathsf{N}\big(\overline{\mathcal{G}}(u), \Gamma(u)\big). \tag{35}$$

In the preceding display, $\Gamma = \Gamma(u)$ is some (possibly u-dependent) covariance matrix, and the "overbar" notation $\overline{\mathcal{G}}$ represents the mean of the distribution of $\mathcal{G}$. Equivalently,

$$z = \overline{\mathcal{G}}(u) + \eta, \quad \text{where} \quad \eta \sim \mathsf{N}\big(0, \Gamma(u)\big). \tag{36}$$

The form (36) emphasizes that the relationship can be written in terms of a deterministic map $\overline{\mathcal{G}}$ plus additive noise. By endowing the input $u \in \mathcal{U}$ with a Bayesian prior distribution, and given a particular observation $z = z^\star$, we can informally state two optimization tasks (i.e., inverse problems): find an optimal $u^\star = u^{(\mathrm{MLE})}$ or $u^\star = u^{(\mathrm{MAP})}$, where

$$u^{(\mathrm{MLE})} \coloneqq \operatorname*{argmax}_{u \in \mathcal{U}} \log P(z^\star \,|\, u) \quad \text{and} \quad u^{(\mathrm{MAP})} \coloneqq \operatorname*{argmax}_{u \in \mathcal{U}} \big\{ \log P(z^\star \,|\, u) + \log P(u) \big\}. \tag{37}$$

Here $w \mapsto P(w)$ denotes the probability density function of a finite-dimensional random variable w. In (37), $P(z^\star \,|\, u)$ is the data likelihood for the observation $z^\star$ arising from any input u and $P(u)$ represents the prior on u. Commonly, optimization frameworks evaluate (36) to compute $P(z^\star \,|\, u)$ and so require approximation of $\overline{\mathcal{G}}(u)$, which is a computationally demanding task in many applications (e.g., via sampling). On the other hand, EKI solves for $u^\star$ from (35) and requires only evaluations of the parameter-to-observable map $u \mapsto \mathcal{G}(u)$.

We now describe EKI. The basic EKI algorithm makes a modeling assumption that is typical of the ensemble Kalman literature by insisting that the prior on parameter u is a Gaussian distribution $\mathsf{N}(m, C)$. Under this condition, we note that

$$u^{(\mathrm{MLE})} = \operatorname*{argmin}_{u \in \mathcal{U}} \big\| \Gamma(u)^{-\frac{1}{2}} \big(z^\star - \mathcal{G}(u) \big) \big\|^2 \quad \text{and} \tag{38a}$$

$$u^{(\mathrm{MAP})} = \operatorname*{argmin}_{u \in \mathcal{U}} \Big\{ \big\| \Gamma(u)^{-\frac{1}{2}} \big(z^\star - \mathcal{G}(u) \big) \big\|^2 + \big\| C^{-\frac{1}{2}} (m - u) \big\|^2 \Big\}, \tag{38b}$$

where the norms are induced by the Euclidean inner product. EKI represents the initial Gaussian distribution empirically from samples; these are called ensemble members. Let the size of the ensemble be denoted by J, which is a critical hyperparameter in EKI. The initial ensemble can be written for each member $j \in [J]$ as i.i.d. samples $u^{(j)}(t_0) \sim \mathsf{N}(m, C)$. Given a sequence $\{t_n\}$ of artificial time values, the Kalman-type update rule for EKI at iteration $n \in [n_{\mathrm{iter}} - 1]$ is given for each $j \in [J]$ by

$$u^{(j)}(t_{n+1}) = u^{(j)}(t_n) + C^{(u\mathcal{G})}(t_n)\big(C^{(\mathcal{G}\mathcal{G})}(t_n) + (\Delta t_{n+1})^{-1}\Gamma\big)^{-1}\big(z^{(j)}(t_{n+1}) - \mathcal{G}(u^{(j)}(t_n))\big), \tag{39a}$$

$$z^{(j)}(t_{n+1}) = z^\star + \xi_{n+1}^{(j)}, \quad \text{where} \quad \xi_{n+1}^{(j)} \sim \mathsf{N}(0, (\Delta t_{n+1})^{-1}\Gamma). \tag{39b}$$

The first equation updates the parameter and the second equation represents a noisy perturbation of the observation $z^\star$. For further interpretations and derivations of the EKI update formulas, see Sanz-Alonso et al. (2023, Chapter 13) and references therein. The time step Δt_{n+1} in the preceding display is usually constant, e.g., $\Delta t_{n+1} \equiv \Delta t = 1$. Writing $\overline{u(t)} \coloneqq \frac{1}{J}\sum_{j=1}^J u^{(j)}(t)$ and similarly[5] for $\overline{\mathcal{G}(u(t))}$, for any t the empirical covariance matrices in (39) are given by

$$C^{(u\mathcal{G})}(t) = \frac{1}{J}\sum_{j=1}^J \Big(u^{(j)}(t) - \overline{u(t)}\Big)\Big(\mathcal{G}(u^{(j)}(t)) - \overline{\mathcal{G}(u(t))}\Big)^\top \quad \text{and} \tag{40a}$$

$$C^{(\mathcal{G}\mathcal{G})}(t) = \frac{1}{J}\sum_{j=1}^J \Big(\mathcal{G}(u^{(j)}(t)) - \overline{\mathcal{G}(u(t))}\Big)\Big(\mathcal{G}(u^{(j)}(t)) - \overline{\mathcal{G}(u(t))}\Big)^\top. \tag{40b}$$

This completes the description of the basic EKI algorithm.

In the setting of a linear parameter-to-observable map $\mathcal{G}$, iterating (39) until time $t_{n_{\text{iter}}} = T = 1$ will approximate the posterior distribution of the Bayesian inverse problem (Calvello et al., 2025). That is, the mean and covariance of the final ensemble will accurately represent the posterior mean and covariance. In the nonlinear case, iterating until $T = 1$ (just one step if $\Delta t \equiv 1$) will approximate $u^{(\text{MAP})}$ in (38). In contrast, upon sending $t \to \infty$, the dynamic will instead approach $u^{(\text{MLE})}$—up to constraints imposed by the span of the initial ensemble. The ensemble is known to greatly underestimate posterior covariances (even at $T = 1$), and so these will be disregarded in this paper except in Section 4.3. Other variants of the ensemble Kalman methodology have been developed to overcome this limitation (Garbuno-Inigo et al., 2020a,b).

The algorithm (39) is the basic form of EKI. As well as previously mentioned variants, there are several features that can be added for additional computational advantages. Many of these may be found in the open-source software package by Dunbar et al. (2022b) and are used later on in the numerical experiments of the present paper. Detailed EKI configurations are provided for each upcoming experiment in Appendix A.

3.3 Recasting Optimization as Inversion

We construct a suitable inverse problem in the form (35) so that the stochastic optimization of the RF empirical Bayes objective function (34) becomes amenable to EKI. Take $\Omega \subseteq [N]$ and $\Omega^{\mathsf{c}} \subseteq [N]$ to be *training* and *validation* index sets, respectively. In what follows, the notation X_Ω denotes the set $\{x_i \in X \,|\, i \in \Omega\}$ and similarly for Y_Ω. The following items define the constituents of the proposed inversion framework:

5. Note the difference in notation: $\overline{\mathcal{G}}(u)$, the mean of the distribution of $\mathcal{G}$ evaluated at u, versus $\overline{\mathcal{G}(u(t))}$, the finite ensemble average of evaluations of $\mathcal{G}$ over $u^{(j)}(t)$ for $j \in [J]$.

(i) (prior) A prior probability distribution is taken over the hyperparameters $u \in \mathcal{U}$.[6] As a canonical example, consider a finite-dimensional vector hyperparameter u where the prior is given by independent probability distributions in each coordinate. These univariate distributions are chosen to ensure a broad spread of values, prevent blow-up during optimization, and enforce hard constraints (e.g., positivity) where needed.

(ii) (forward Map) Let $\overline{F}^{(N)}_{M,\Omega}(\,\cdot\,;u)$ be the RF posterior mean as in (27) except with the posterior obtained by conditioning on the smaller data set $(X_\Omega, Y_\Omega) \subset (X, Y)$. Similarly, let $\beta^{(u)}_\Omega \in \mathbb{R}^M$ be the coefficients (recall Equation 25) of the mean $\overline{F}^{(N)}_{M,\Omega}(\,\cdot\,;u)$. Finally, write $G^{(u)}_{M,\Omega}$ for the RF Gram matrix (29) except with the sum now over the set Ω instead of $[N]$. The forward map of the proposed inverse problem framework is then defined for each $u \in \mathcal{U}$ to be

$$\mathcal{G}(u) \coloneqq \begin{pmatrix} \overline{F}^{(N)}_{M,\Omega}(X_{\Omega^c};u) \\ \frac{1}{\sqrt{M}}\|\beta^{(u)}_\Omega\| \\ \sqrt{\log\det\big(G^{(u)}_{M,\Omega} + I_M\big)} \end{pmatrix}. \tag{41}$$

(iii) (observable) The corresponding "observable" quantity z (35) is

$$z \coloneqq \begin{pmatrix} Y_{\Omega^c} \\ 0 \\ 0 \end{pmatrix}, \tag{42}$$

which leads to the equation $z = \mathcal{G}(u)$. The zero entry in z for the log determinant component in $\mathcal{G}(u)$ is due to the Minkowski determinant theorem. Indeed, this theorem yields the lower bound

$$\begin{aligned} \det\big(G^{(u)}_{M,\Omega} + I_M\big)^{1/M} &\geq \det\big(G^{(u)}_{M,\Omega}\big)^{1/M} + \det(I_M)^{1/M} \\ &\geq \det(I_M)^{1/M} \\ &= 1 \end{aligned}$$

because $G^{(u)}_{M,\Omega}$ and I_M are positive-semidefinite. This implies that zero is a uniform lower bound for the log determinant term, i.e.,

$$\inf_{u\in\mathcal{U}}\Big\{\log\det\big(G^{(u)}_{M,\Omega} + I_M\big)\Big\} \geq 0. \tag{43}$$

In particular, the square root in the third component of (41) is well-defined.

(iv) (correlations) Let B^{c}_Σ denote the block diagonal matrix with $N - |\Omega|$ identical diagonal blocks containing the noise covariance matrix $\Sigma \in \mathbb{R}^{p\times p}$. The variability η of the artificial forward map evaluation $\mathcal{G}(u)$ (41) about its mean is assumed to be a Gaussian $\mathsf{N}(0, \Gamma(u))$ with covariance matrix[7]

6. The exact details may be found in Appendix A.1.
7. The zeros in (44) denote rectangular zero matrices of the appropriate size.

$$\Gamma(u) \coloneqq \mathrm{Cov}_{\mu_u^{\otimes M}}\big(\mathcal{G}(u)\big) + \begin{pmatrix} B_\Sigma^{\mathsf{c}} & 0 \\ 0 & I_2 \end{pmatrix}, \tag{44}$$

namely, a contribution from the variability within $\mathcal{G}$ and from the noise perturbing the labeled training data. In (44), the covariance operation on $\mathcal{G}(u)$ is taken with respect to the random features and is conditional on the realization of the noisy training data that underlie the regression problem. The 2 by 2 identity matrix in the lower right block is needed to maintain consistency with the ideal empirical Bayes objective (34) in the large feature limit, as discussed next.

With these four ingredients in hand, we may now directly apply the EKI framework from Subsection 3.2. Iterating the EKI algorithm with small enough Δt towards termination time $T = 1$ will minimize the functional $L_M^{(\mathrm{EKI})}$ defined by

$$L_M^{(\mathrm{EKI})}(u) \coloneqq \big\|\Gamma(u)^{-\frac{1}{2}}\big(z - \mathcal{G}(u)\big)\big\|^2 - 2\log P(u), \tag{45}$$

where $P(u)$ is the prior probability density of u; cf. (38). This is the weighted observable misfit added to a diminishing contribution from the prior on u as the number of iterations increases. The prior contribution ensures that desired constraints such as positivity hold. The implied EKI objective function (45) is related to the ideal empirical Bayes cost (34) in the infinite feature limit. Indeed, in this limit, the term $\mathrm{Cov}_{\mu_u^{\otimes M}}(\mathcal{G}(u))$ tends to zero so that for large enough M, it holds that

$$\begin{aligned} L_M^{(\mathrm{EKI})}(u) \approx \frac{1}{M}\big\|\beta_\Omega^{(u)}\big\|_{\mathbb{R}^M}^2 + \Big\|(B_\Sigma^{\mathsf{c}})^{-\frac{1}{2}}\big(Y_{\Omega^{\mathsf{c}}} - \overline{F}_{M,\Omega}^{(N)}(X_{\Omega^{\mathsf{c}}}; u)\big)\Big\|^2 \\ + \log\det\big(G_{M,\Omega}^{(u)} + I_M\big) - 2\log P(u). \end{aligned} \tag{46}$$

The terms on the right hand side of (46) in this large feature limit are structurally similar to those in the cost $L_M^{(\mathrm{EB})}$ (34) when $\Omega = \Omega^{\mathsf{c}} = [N]$. In practice, we prefer instead a construction similar to cross-validation: take a fixed disjoint partition $\bigcup_{k=1}^{\mathsf{K}} \Omega_k = [N]$, where $|\Omega_k| = N/\mathsf{K}$ and K divides N. We define a validation partition for an index $j \in [\mathsf{K}]$ as $\Omega^{\mathsf{c}} \coloneqq \Omega_j$; the remaining indices form the training partition $\Omega \coloneqq \bigcup_{k \neq j} \Omega_k$. Repeating this procedure by cycling through $j \in [\mathsf{K}]$, we may incorporate up to K partitions by stacking for each j the corresponding data z for each validation partition and comparing this to stacked forward map evaluations $\mathcal{G}(u)$ built from the associated training partition, plus additive Gaussian noise with block diagonal covariance matrix having Γ as each block. The introduction of these validation partitions improves computational efficiency, which scales with $|\Omega^{\mathsf{c}}| \leq N$ instead of N. With even two such partitions, we also observe improved convergence of the optimizer and improved accuracy in experiments. We hypothesize that cross-validation can improve the landscape of the objective function but leave further investigation to future work. Since (46) arises from a Bayesian inverse problem, it also requires a prior to regularize the inversion, and hence optimization; this contribution appears in $L_M^{(\mathrm{EKI})}$ and in (46) but not in $L_M^{(\mathrm{EB})}$. Although (46) is approximately valid for large numbers of random features, for efficiency purposes we minimize it with relatively few features in practice.[8]

8. The effect of varying the number of features during optimization is explored in Appendix A.3.1.

3.4 A Practical Feature Distribution Structure

In this subsection, we instantiate our general framework with an implementable parametrization of the RF pair (φ, μ_u). In all numerical experiments to follow, we employ real vector-valued random Fourier features (Rahimi and Recht, 2007) with Gaussian samples and an additional scaling parameter $\varsigma > 0$. Specifically, define for each $x \in \mathcal{X} \subset \mathbb{R}^d$ the $\mathbb{R}^p$-valued feature map

$$\varphi(x;\theta) \coloneqq \sqrt{\varsigma}\cos(\Xi x + B), \quad \text{where} \quad \theta = (\Xi, B) \sim \mathscr{D} \tag{47}$$

and $\mathscr{D} \coloneqq \mathsf{N}(0, C) \otimes \mathsf{Unif}([0, 2\pi]^p)$. The RF distribution $\mathscr{D}$ has high-dimensional hyperparameter $C \in \mathbb{R}^{dp\times dp}$, the covariance operator over the Gaussian-distributed matrix Ξ. We always reshape samples of Ξ from $\mathsf{N}(0, C)$ into a $p \times d$ matrix. To reduce the number of tunable parameters (which is $\mathcal{O}(d^2p^2)$) and computational effort, one can impose structure on C in various forms, for example, through Cholesky factorizations, diagonal approximations, low-rank parametrizations, or tensor products. We choose a low-rank perturbation representation (Ambikasaran et al., 2016) as follows. For some $r \in \mathbb{N}$, take $U \in \mathbb{R}^{dp\times r}$ and symmetric positive definite $S \in \mathbb{R}^{r\times r}$. Then, define

$$C \coloneqq \left(I_{dp} + USU^\top\right)\left(I_{dp} + USU^\top\right)^\top. \tag{48}$$

The task is then to jointly learn the hyperparameters (ς, U, S). We enforce that S be diagonal, leading to a moderate number of total tunable parameters $r(dp + r) + 1$. This is advantageous whenever $r \ll dp$. We refer to the preceding construction as a *nonseparable feature distribution* with rank r. We also consider $\Xi \sim \mathsf{MatrixNormal}(0, C_{\text{in}}, C_{\text{out}})$ (Dutilleul, 1999) and tune the pair $(C_{\text{in}}, C_{\text{out}}) \in \mathbb{R}^{d\times d} \times \mathbb{R}^{p\times p}$. This pair of covariances can be identified with a larger covariance $C \in \mathbb{R}^{dp\times dp}$ with Kronecker structure. This construction is referred to as a *separable feature distribution* with rank $(r_{\text{in}}, r_{\text{out}})$ and leads to $r_{\text{in}}(d + r_{\text{in}}) + r_{\text{out}}(p + r_{\text{out}}) + 1$ total tuneable parameters. A suitable rank is selected to produce emulators with minimal L^2 errors on a held-out test set in experiments.

There are many other options for parametrization beyond (vector-valued) random Fourier features of the form (47) and (48). For instance, one could parametrize a possibly sparse precision matrix instead of the Gaussian covariance C, or use other efficient representations such as FastFood kernels (Le et al., 2013; Yang et al., 2015). Going beyond Gaussian samples, one can take inspiration from the scalar output $p = 1$ case. Here, sampling Ξ from a Student's t distribution leads to a limiting Matérn kernel; this follows from the fact that such densities are Fourier transform pairs. Similar results hold for Cauchy and Laplace distributions. In the $p > 1$ case, one can still sample Ξ from such non-Gaussian distributions on the reshaped $\mathbb{R}^{dp}$ space. The main hyperparameters that characterize these distributions are generalized lengthscale matrices $C \in \mathbb{R}^{dp\times dp}$, just as in the Gaussian case. Going beyond the Fourier setting, one approach is to work instead with random neural networks, where the feature map is a hidden neuron of the form $x \mapsto \varrho(\Xi x + B)$ for some activation function $\varrho(\cdot)$. These directions are left to future work.

3.5 Current Limitations of the Methodology

We now detail additional practical considerations of the proposed stochastic inversion framework as well as some of its limitations. First, we highlight that the law of $\mathcal{G}(u)$ and the

prior distributions are assumed to be Gaussian in our formulation. This assumption is not strictly necessary, but it enables straightforward application of ensemble Kalman methods.

Though RFR has admirable scaling, the objective function (34) scales naïvely as $(Np)^2$. This significantly slows down workflows involving high-dimensional spaces or large data sets. Therefore, controlling the computational cost necessitates further approximations at implementation time. One example, already discussed in Section 3.3, is to evaluate the objective on a validation partition of cardinality less than or equal to the total sample size N. Another approximation is to avoid computing $u \mapsto \Gamma(u)$ in (44) at all u (requiring the covariance of (41) under the random features) by using an offline constant approximation $\Gamma(u) \approx \Gamma(u^\dagger)$ at a fixed $u^\dagger$. In the following numerical results, $u^\dagger$ is taken as the mean of the hyperparameter prior. The effect of improved approximation of $\Gamma(u)$ is not considered in the current paper but remains an interesting direction for future work.

Finding local optima naturally has dependence on the initial condition. For EKI, this arises from the prior used to create the initial ensemble. There is much debate on prior selection (Gelman et al., 2017), particularly in high-dimensional and nonphysical systems. The heuristic adopted in the present work is to choose weakly informative priors over the hyperparameters. We use the same prior structure across all of the forthcoming numerical experiments. Appendix A.1 contains the details. Moreover, it is tempting to directly optimize the likelihood, i.e., define instead $\mathcal{G}(u)$ to be equal to the scalar value $-\log P(z \mid u)$ and take $z = 0$ as the "observable". This results in a one-dimensional optimization problem. Unfortunately, we report that this approach incurs large approximation errors in numerical experiments with input or output dimensions of even moderate size. The more costly vector-valued formulation in Subsection 3.3 is preferred.

4. Applications

To illustrate the role of the hyperparameter tuning, this section presents different applications that require the machine learning emulator to represent nonlocal features away from the training data. Global sensitivity analysis requires the posterior mean function to be accurate at quasi-random samples from the d-dimensional cube (Subsection 4.1); to integrate a Lorenz 63 system, the posterior mean function must progress along, and attract towards, the Lorenz attractor (Subsection 4.2); for accelerated uncertainty quantification, the emulator mean and covariance predictions are used within a sampling method to explore the support of a five-dimensional posterior distribution (Subsection 4.3). In all three applications, the data consist of a moderate number of observations that are perturbed by additive Gaussian noise with known covariance matrix.

For comparison to the approach developed in the present paper, two other emulators are presented, an untuned RF model using hyperparameters from the mean of the EKI initialization and a tuned GP from a robust software package. To emulate smooth functions $\mathbb{R}^d \to \mathbb{R}$ (i.e., $p = 1$) with GPs, we use a radial basis function kernel with scaling parameter, nugget parameter, and learnable lengthscales in each of the d input dimensions. To handle $p > 1$ output dimensions, we use a singular value decomposition (SVD) in the output space to decorrelate the coordinates, and then fit and tune p decorrelated GP regressors in each dimension. This corresponds to a single vector-valued GP regressor with a diagonal matrix-valued kernel. Therefore, a modest $d(p + 2)$ parameters are found with gradient-based

optimizers built into the chosen software packages. In our experiments we report results from `SciKitLearn.jl v0.7`. We also tried `GaussianProcesses.jl v0.12.5`, but found hyperparameter optimization to be too brittle to provide consistent comparisons.

Design Choices. To support many of the heuristics in the upcoming experiments, we provide supplementary investigations in Appendix A for each application. These include details about the EKI configuration, objective function configuration, and convergence plots over different random initializations. Some other specific studies we include are run times and accuracy with respect to input dimension size (Appendix A.3) and numbers of features (Appendix A.3.1), and accuracy over different feature distribution ranks and covariance structures in the multi-output setting (Appendix A.4.1 and A.5).

4.1 Global Sensitivity Analysis

Global sensitivity analysis (GSA) seeks to attribute the variance of model output to its input variables over a given domain (Saltelli, 2002). Practitioners are interested in either ranking variables by their direct contribution to the variance, computing $\mathsf{V}_i = \mathbb{V}(\mathbb{E}(f(x)\,|\,x_i))/\mathbb{V}(f(x))$, or residuals of the total variance when conditioning on all other variables, $\mathsf{TV}_i = (\mathbb{V}(f(x)) - \mathbb{V}(\mathbb{E}(f(x)\,|\,x_{j\neq i})))/\mathbb{V}(f(x))$. Here $x_{j\neq i}$ indicates all variables except x_i and $\mathbb{V}$ represents the variance operator.

Two standard functions from the sensitivity analysis literature are used in the demonstration. These are chosen for their analytic representation of V_i and TV_i. Furthermore, accurate empirical estimates of V_i and TV_i can be calculated on the cube using quasi-Monte Carlo methods. Functions are evaluated over points from a low discrepancy Sobol sequence (Sobol and Levitan, 1999) and are referred to as *empirical estimates* in our results.

4.1.1 Ishigami Function

First, the Ishigami function (Ishigami and Homma, 1990; Sobol and Levitan, 1999)

$$x \mapsto f(x; a, b) \coloneqq (1 + bx_3^4)\sin(x_1) + a\sin(x_2) \tag{49}$$

on $[-\pi, \pi]^3$ with $a = 7$ and $b = 0.1$ is considered. Data is taken as 300 evaluations of the Ishigami function from a length 16000 Sobol sequence, and Gaussian noise is added with variance 0.1^2. After their hyperparameters are tuned, the emulators are fit to the data and used to predict values over the entire Sobol sequence. This is how V_i and TV_i are calculated empirically.

Figure 1 shows the results from a single trial. Row 1 shows slices of the true Ishigami function against each variable evaluated on the Sobol sequence. Rows 2–4 show the noisy data (red) used for learning and the emulator predictions over the Sobol sequence (blue). Row 2 corresponds to the untuned RF, Row 3 is the prediction from a tuned GP, and Row 4 shows results of the EKI-tuned RF with 500 features. The successful trials show that the tuned models fit the true Ishigami function well.

To quantify the accuracy of GSA, Table 2 displays the calculated Sobol indices of the tuned emulators. The tuning was repeated over 20 different initial samples to gauge the robustness of randomized hyperparameter learning algorithms. Table 2 shows the mean and standard deviation of these trials. For both V_i and TV_i, both GP and RF produce close

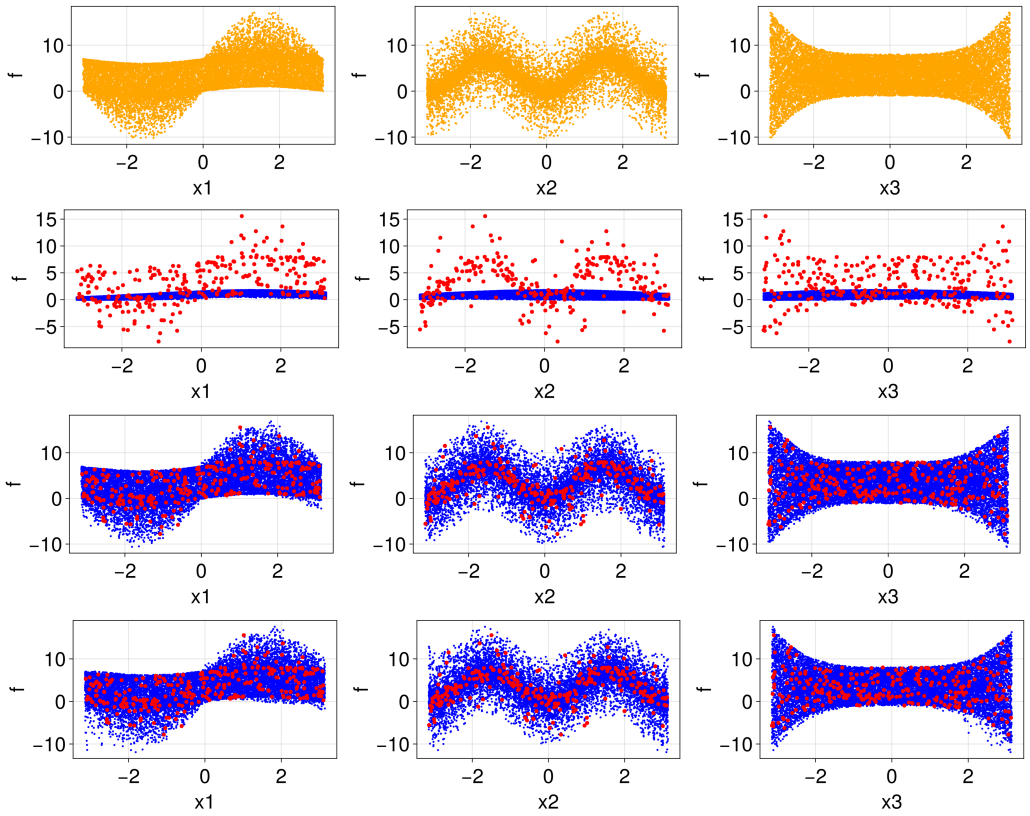

Figure 1: Learning the $\mathbb{R}^3 \to \mathbb{R}$ Ishigami function from 300 noisy samples. In orange, Sobol samples of the true function; in red, 300 noisy observations; in blue, Sobol samples of the emulated functions. Row 2 is sampled from an RF emulator without hyperparameter tuning, Row 3 is sampled from a tuned GP emulator, and Row 4 is sampled from the RF approximation (using a nonseparable feature distribution with full rank covariance) using our hyperparameter optimizer.

values. In particular, RF has variations of at most 5% of the true sensitivity and the true values always lie within two standard deviations of the RF emulator variability.

Further details about the implementation of this experiment relating to EKI configuration and convergence are provided in Appendix A.3. The algorithm is robust to the data seeding, the number of features used for optimization and prediction, and to various modifiers of the optimization algorithm.

4.1.2 Sobol G Function

The Sobol G function (Archer et al., 1997; Saltelli et al., 2010), defined by

$$x \mapsto G(x; a) = \prod_{i=1}^{d} \frac{|4x_i - 2| + a_i}{1 + x_i}, \tag{50}$$

is another GSA test function that additionally allows for varying the input dimension d and for tuneable interaction between different inputs via the choice of coefficients $\{a_i\}_{i=1}^{d}$. Here

Table 2: First order global sensitivity analysis of the Ishigami function. Displayed are the analytic values, the empirical approximation from evaluating the true function at 16000 Sobol indices, and the empirical approximations obtained from evaluating the tuned GP and RF emulators at the Sobol indices. The final column estimates a mean and standard deviation (in parentheses) of the emulators from repeating the stochastic tuning procedure 20 times on the same data.

Sobol Index	Analytic	Empirical	Tuned GP	Tuned RF	
V_1	0.314	0.313	0.309	0.308	(0.024)
V_2	0.442	0.442	0.451	0.453	(0.023)
V_3	0	−0.006	−0.007	−0.007	(0.005)
TV_1	0.557	0.562	0.550	0.537	(0.026)
TV_2	0.442	0.442	0.459	0.498	(0.038)
TV_3	0.244	0.245	0.235	0.223	(0.022)

we take $a_i \coloneqq (i-1)/2 \geq 0$, where small i (and thus small a_i) imply larger first-order effects; interactions are primarily present between these variables.

For this experiment, the results are only presented with a tuned RF. We remark that a tuned GP was accurate in offline tests for dimensions $d = 3, 6$, and 10, but timed out at $d = 20$ dimensions. The untuned RF performed very poorly and was removed for plot readability. Over input dimensions $d = 3, 6, 10$, and 20, approximately $2000 \cdot d$ Sobol points are used to produce empirical estimates of first-order Sobol indices. The exact number is determined by the GSA package `GlobalSensitivityAnalysis.jl v1.2.0`. Of these points, $250 \cdot d$ (approximately 10% to 15%) were used for tuning the emulator. The posterior mean evaluation at all Sobol points was then used to calculate empirical Sobol indices.

Figure 2 shows the indices V_i and TV_i for $i = 1, \dots d$ with one panel for each experiment. The emulation was repeated 30 times to produce the spread. Qualitatively, the RF is always able to capture the decaying sensitivity as i increases. Quantitatively, for $d = 3, 6$, and 10, the RF spread well captures the analytic indices, while for $d = 20$, only the V_i index is captured well. Most noticeably for the TV_i indices, there is a dimensional effect: as more insensitive dimensions are added, the RF emulation begins to overpredict the TV_i of the least sensitive dimensions and correspondingly overpredicts TV_i in the most sensitive dimensions. Figure 3 provide views in the three most sensitive dimensions for just one of the RF realizations. The emulator has more of a challenge to capture the sharp transition at $x_i = 0.5$ in the most sensitive dimensions as d increases. Appendix A.3 contains additional results pertaining to emulator performance as the input dimension d and the number of features M are varied.

4.2 Lorenz 63 Integrator

The next task is to learn an integrator, or $\mathbb{R}^3 \to \mathbb{R}^3$ next-step map, of the Lorenz 63 dynamical system (Lorenz, 1963). The data is produced from observations of a trajectory

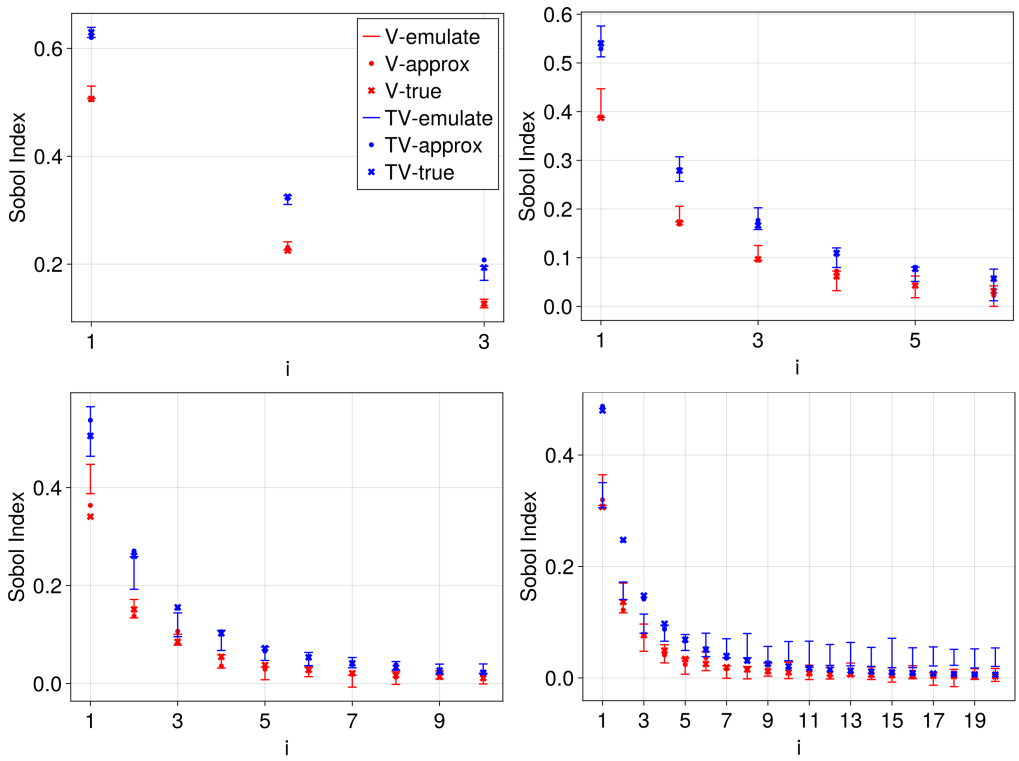

Figure 2: First order global sensitivity analysis of the Sobol G function with input dimensions $d = 3, 6, 10, 20$. The vertical axis represents the dimensionless values and the horizontal axis represents the index. Crosses ($\times$) denote the analytic Sobol indices, circles ($\circ$) denote the empirical indices calculated at $1600 \cdot d$ points, error bars denote the $(0.05, 0.95)$ percentile range of 30 random feature trials with $250 \cdot d$ training data points.

of the full state that is perturbed with additive Gaussian noise. There is growing use of the Lorenz system (without noise) in the recurrent network literature to validate the learning of deep or recurrent structure from full-state observations (Shahi et al., 2022; Dudul, 2005; Fablet et al., 2018; Scher and Messori, 2019); here, learning is demonstrated efficiently without deep networks. Since the Lorenz 63 system gives rise to the archetypal chaotic dynamics, the measure of a successful integrator is an accurate representation of the chaotic attractor over long integration times rather than pointwise accurate fits to a test trajectory.

The classical parameter choices $(\sigma, \rho, \beta) := (10, 28, 8/3)$ are chosen for the Lorenz system

$$\frac{dx}{dt} = \sigma(y - x)\,, \tag{51a}$$

$$\frac{dy}{dt} = x(\rho - z) - y\,, \tag{51b}$$

$$\frac{dz}{dt} = xy - \beta z \tag{51c}$$

to produce chaotic dynamics. The state is denoted by $(x(t), y(t), z(t))$ at each time t. The true dynamics were integrated with an Euler method and time step of 10^{-2}. For the sample

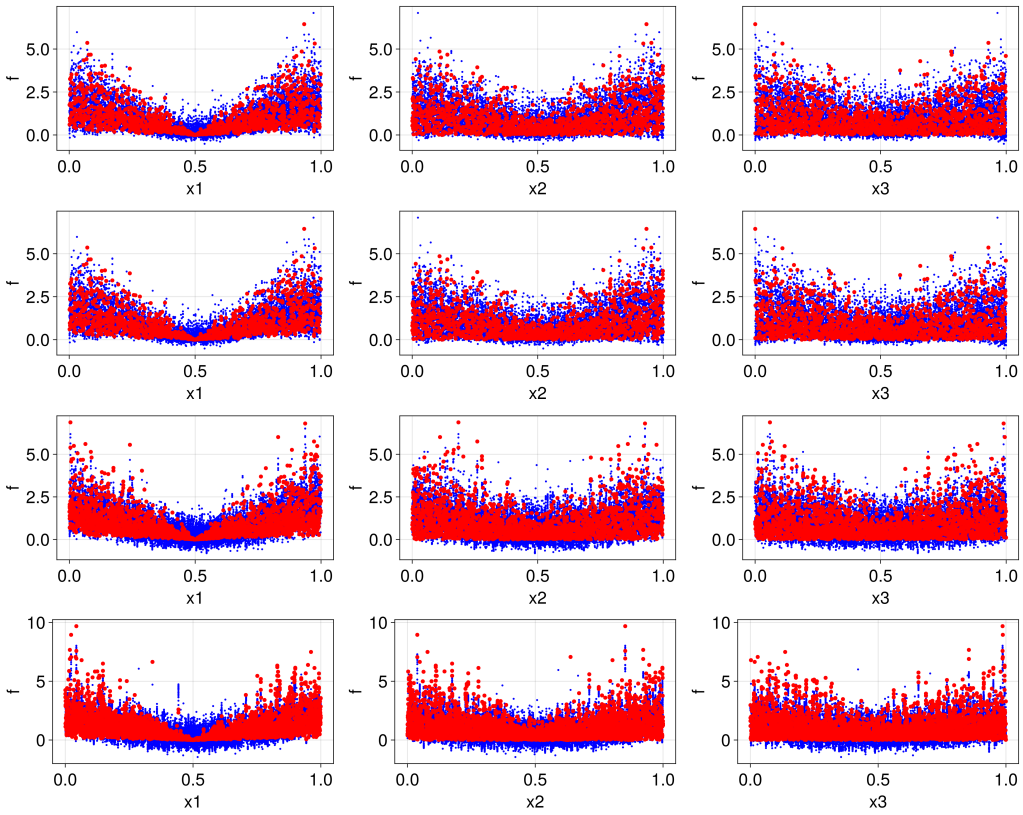

Figure 3: Views of the Sobol G function plotted against the first three variables for $d = 3$, 6, 10, and 20 (from top to bottom row). The red points denote the noisy observed data, and the blue represents the tuned RF emulator prediction at all other Sobol points.

size, 500 training data pairs were sourced randomly[9] from $[0, 20]$. The data pairs therefore comprise inputs $\{(x(t_n), y(t_n), z(t_n)\}$ and (noisy) outputs $\{(x(t_{n+1}), y(t_{n+1}), z(t_{n+1})) + \eta\}$, where $\eta \sim \mathsf{N}(0, \Sigma)$ and $\{t_n\}$ is the Euler time discretization. We emulate these maps with both scalar GPs (requiring decorrelation of the output space) and with vector-valued RFs (native output space).

For validation, we recursively apply the emulated integrators on time interval $[20, 1020]$, plotting the trace and emulated attractor over $[20, 40]$ and the marginal probability density functions (PDFs) of the individual state variables. The results for $\Sigma = 10^{-4}I$ are shown in Figure 4, with the true integrated system in orange and the emulated system in blue. Row 1 shows the fit of an untuned RF emulator with $M = 600$ features. Row 2 shows the fit of the tuned GP emulator, and Row 3 shows the fit of the tuned vector-valued RF emulation. Without hyperparameter learning, the integrator failed to reproduce the attractor, while both the tuned vector-valued RF and GP methods demonstrate stable integration for long times. The trace plot illustrates a trajectory forecast skill until time $T = 4$ for RF and for GP, and after these time horizons the trajectories diverge as expected. Importantly, the

9. In experiments, sourcing data sequentially from a trajectory gives similar performance and introducing a spin-up time (e.g., learn with data from time interval $[10, 20]$) decreases performance slightly.

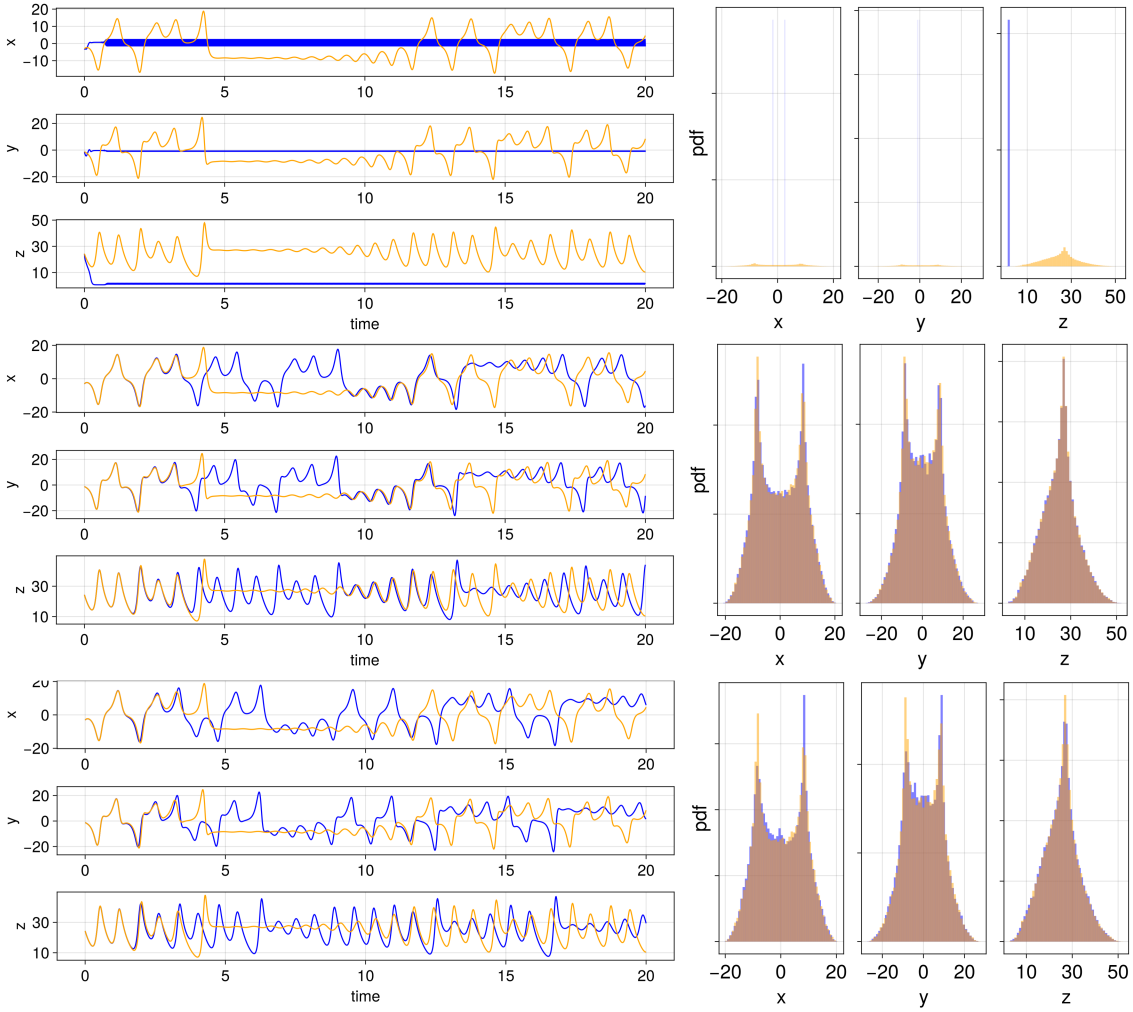

Figure 4: Learning a Lorenz 63 integrator from noisy data: emulated (blue) vs. truth (orange). The first column displays the time evolution of the three state variables $x(t)$, $y(t)$, and $z(t)$ (vertical axis) as a function of time t (horizontal axis) on initial conditions not seen during training. The second column visualizes the marginal probability density functions for each state variable. Row 1 shows integration with the untuned RF without hyperparameter learning. Row 2 shows integration with the GP emulator with 12 total hyperparameters learned, and Row 3 shows integration with RF using a rank-4 nonseparable feature distribution with 31 total hyperparameters learned. The 500 data pairs were subjected to observational noise with covariance $\Sigma = 10^{-4}I$.

emulated integrators continue to remain on the attractor, fitting the long-time PDFs closely. The approach is relatively robust to noise in the observations, as seen in the supplementary Figure 9, where the noise of the data is increased to $\Sigma = 10^{-2}I$. Here the short term forecasts are still relatively accurate and the dynamics still explore the chaotic attractor. However, the long-time PDFs lose accuracy. Further increases to the noise eventually lead

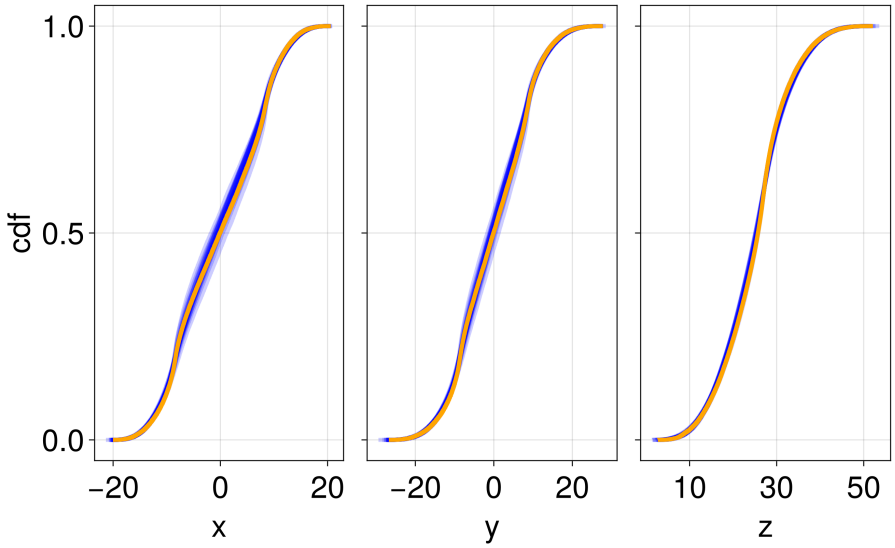

Figure 5: Comparison of true marginal empirical CDF (orange) with the CDF of the RF emulator corresponding to 30 re-tunings of the hyperparameters (blue) when performing the experiment displayed in Figure 4. The noise covariance is $\Sigma = 10^{-4}I$.

to a catastrophic failure, where emulators were observed to collapse to periodic orbits or fixed points.

To inspect the robustness of the optimization methods, Figure 5 shows the difference between the true cumulative density function (CDF) and emulated CDF corresponding to the tuned RF emulator over 30 independent runs of the stochastic optimization algorithm. Parameter sweeps also show that the algorithm is robust to the data seeding, to the number of features used for optimization and prediction, and to various modifiers of the optimization algorithm. Our results are consistent with efficient integrator learning approaches for this chaotic system. These approaches include recurrent networks, reservoir computers, or vector auto-regression models (Shahi et al., 2022) that are often tuned and fit to 10^3 to 10^4 noiseless points. In contrast, feedforward or deep neural networks are frequently trained on more than 10^4 noiseless points (Dudul, 2005; Fablet et al., 2018; Scher and Messori, 2019). Further details about the implementation of this experiment relating to EKI configuration and its convergence are provided in Appendix A.4. This appendix also includes supplementary results that explore the effect of feature distribution structure—such as rank and separability—on performance.

4.3 Uncertainty Quantification of a Cloud Model

The final task is to investigate a parametric uncertainty quantification problem. Here, one has access to a black-box computational simulator G—specifically a parameter-to-data map—that is dependent on several parameters θ, e.g., unknown physical constants. Given a noisy observation y, the task is to estimate the posterior parameter distribution that explains this measured data. Common challenges in the scientific realization of such problems are

that (i) the simulator can only be run a relatively few times (e.g., fewer than 10^3 times), (ii) the parameter-to-data map is likely non-differentiable or noisy, or both, and (iii) the parametrizations are not directly observable. One can address these challenges by posing the underlying task as a classical Bayesian inverse problem and using gold-standard sampling methods such as MCMC (Metropolis et al., 1953; Sherlock et al., 2010). Under a Gaussian noise model with covariance matrix Γ,[10] MCMC algorithms draw samples from a density over θ proportional to $\exp(-L(\theta, y))$, where the negative log-posterior is proportional to

$$L(\theta, y) \coloneqq \frac{1}{2}\left\|\Gamma^{-1/2}\big(y - G(\theta)\big)\right\|^2 + \frac{1}{2}\log\det\Gamma - \log P(\theta). \tag{52}$$

In (52), $P(\theta)$ denotes the probability density function of the prior on θ. Unfortunately, MCMC sampling algorithms may require $\mathcal{O}(10^3)$ times more model evaluations than those available. Furthermore, non-differentiable and noisy forward maps G may cause rough landscapes that are challenging to sample the posterior from.

One approach to address the challenges of using MCMC is to replace the parameter-to-data forward map with a smooth and fast-to-evaluate surrogate (e.g., a statistical machine learning emulator) and sample using the surrogate instead. For example, when the forward map is random, previous work (Cleary et al., 2021; Dunbar et al., 2021) approximated the forward map by a tuned GP posterior $G \sim \mathsf{GP}(G_{\mathrm{ML}}, \Gamma_{\mathrm{ML}})$. Then, (52) can be replaced with

$$L(\theta, y) \coloneqq \frac{1}{2}\left\|\big(\Gamma_{\mathrm{ML}}(\theta)\big)^{-1/2}\big(y - G_{\mathrm{ML}}(\theta)\big)\right\|^2 + \frac{1}{2}\log\det\big(\Gamma_{\mathrm{ML}}(\theta)\big) - \log P(\theta). \tag{53}$$

This approximation and the efficient selection of training points around the regions of high posterior mass are discussed at length in these studies; the algorithm is known as Calibrate-Emulate-Sample (CES) (Cleary et al., 2021). The statistical emulator of the forward map provides a natural smoothing property as it can directly predict the mean of the random process. In the present paper, the goal is to compare the resulting approximate posterior distributions for a scientific inverse problem using GP emulators (GP-CES) and RF emulators (RF-CES).

The example in this section is taken from the geophysics literature (Lopez-Gomez et al., 2022). At its core, the emulation task involves representing a map $G\colon \mathbb{R}^5 \to \mathbb{R}^{50}$. To handle the multi-output problem with GPs, the output space is whitened and 50 independent $\mathbb{R}^5 \to \mathbb{R}$ maps are tuned and fitted. The 50 GPs can be viewed as a single vector-valued GP with a corresponding diagonal matrix-valued kernel. This approach depends entirely on the existence of the whitening approximation—which assumes a statistical structure on the inputs and outputs—and the training data sample size, while the new proposed approach using vector-valued RFs does not.

In this scientific application, the forward map is a one-dimensional vertical column model that represents turbulence and moist convection processes in the atmosphere. It can also represent cloud formation. The model is a parametrization of three-dimensional flow and is a type of eddy-diffusivity mass-flux (EDMF) model. It depends on several parameters governing eddy dissipation and diffusion due to turbulence, stability under convection, and the transport of tracers in (entrainment) and out (detrainment) of cloud updrafts. Both the

10. This matrix Γ is inherent to the physical inverse problem and should not be confused with the noise model for the EKI algorithm appearing in (45), which includes a matrix denoted by the same symbol.

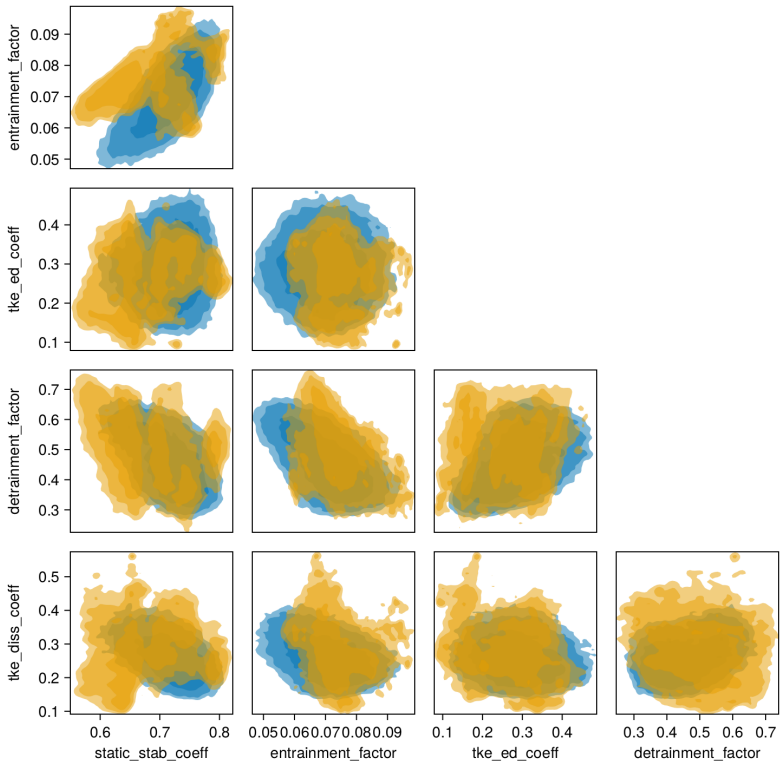

Figure 6: Application of CES to the EDMF problem. Displayed are corner-plots showing slices through the joint distribution of five physically meaningful parameters calibrated to data. Orange (blue) contours: the one, two, and three standard deviation density of the posterior distribution obtained using GP-CES (RF-CES) after hyperparameter tuning.

model and experiment setup is detailed at length in the work of Lopez-Gomez et al. (2022). The data is taken to be time-averaged liquid water path (a scalar value measuring the amount of cloud water in the column) from a suite of 50 three-dimensional cloud-resolving Large Eddy Simulations (LES); each LES simulation represents a different forcing scenario that mimics a range of sites and climates. The observational noise comes from the LES and uses artificial noise inflation to reasonably capture the structural model error. EKI was used to find optimal parameter values. In this experiment, we use an emulator tuned with input-output samples generated by the iterates of the EKI algorithm within CES to approximate the full posterior distribution.

Figure 6 displays pairwise slices through the five-dimensional posterior distribution for the geophysical problem. Visualized are the final 10^5 steps of MCMC using the likelihood calculated from (53). In orange contours, the GP-CES algorithm is used. In blue contours, the RF-CES algorithm is used. Reasonable qualitative agreement is shown across the support of the approximate posterior obtained from either approach. However, the GP-CES distribution contain some fine-scale artifacts, while the RF-CES distribution appears smoother overall. Such artifacts appear to be due to learning small lengthscales in the GP hyperparameter tuning algorithm rather than physically meaningful features in the poste-

rior distribution. The GP-CES posterior was also slower to converge; four times as many samples of burn-in were taken for GP-CES compared to those taken for RF-CES. Likewise, pairwise correlation between several variables such as entrainment (`entrainment_factor`) and the stability coefficient (`static_stab_coeff`) are more clearly seen in the vector-valued RF setting.

Further details of the implementation of this experiment relating to EKI configuration and convergence are provided in Appendix A.5. In that appendix, we also show numerical results using an untuned RF emulator and a tuned RF emulator with a separable feature distribution. One caveat pertaining to the results of the present section is the finding that the structure of the feature distribution plays a significant role in the shape of the posterior distribution (as does the choice of covariance kernel for GP). The design choices (e.g., covariance structure and rank) we selected were based on L^2 errors of the tuned predictors evaluated on a test set, as reported in Table 6. However, these forward map errors were not as reliable at predicting the skill in sampling the posterior distribution in this experiment, as suggested by the supplementary Figure 12.

5. Conclusion

Ensemble Kalman inversion (EKI) and its variants have proven to be easily implementable and versatile for efficiently solving optimization problems with stochastic objective functions. Such problems arise when tuning hyperparameters in randomized algorithms, and specifically in the model selection problem found in randomized machine learning approaches. This paper explored a concrete implementation applied to random feature regression for function emulation. The main developments in this work were threefold. First, by viewing random feature emulators as Gaussian processes, the paper used empirical Bayesian arguments to construct an objective function for hyperparameter estimation. Second, this objective function was adapted to an inverse problem framework compatible with EKI. Third, the work provided structured choices of feature distributions which, when tuned with EKI, produced emulators that performed comparably to Gaussian processes and displayed robustness to overfitting and feature selection.

5.1 Discussion

Random features can offer satisfactory approximation with fewer structural assumptions and computational requirements than Gaussian processes. As illustrated in Section 4.3, vector-valued random feature emulators were able to produce qualitatively similar posterior distributions without relying on diagonalization in the output space—though the framework can take advantage of diagonal structure when known. As with Gaussian processes, the choice of feature map and distribution is important. In this paper, the random Fourier features with feature distributions parametrized by low-rank perturbations (Subsection 3.4) were effective across all test cases for a suitably large rank. Further investigation of sensible feature distribution selection is left to future work.

Performance of random feature regression depends on balancing expressiveness with efficiency in several areas. One must select an appropriate feature distribution family that is both flexible and parameter-efficient for large input and output dimensions. One must propose an optimizer and objective function in order to tune the free parameters of this

family efficiently and robustly. The proposed feature distribution families in Subsection 3.4 were shown to be sufficiently flexible for a wide range of applications, but required a rank that had to be selected offline. The objective function (45) contains a data misfit that scales with Np and the noise factor Γ with $(Np)^2$, where N is the size of the training data set and p the output dimension. The proposed adjustment in this paper is to use a cross-validation-inspired partitioning and an offline estimation of a constant Γ to reduce these costs. Such changes afforded a feasible online cost of the objective function in our experiments, but we expect that a better balance of accuracy with efficiency can be gained through more principled approximation of Γ. The proposed EKI optimizer relies on selection of an ensemble size and a prior distribution. A success of EKI is that the ensemble size and parameter dimension are in fact only weakly coupled and newer variants use sampling error correction methods to decouple them further (Vishny et al., 2024). Such scalability properties are inherited from the ensemble Kalman filtering literature, where assimilators must routinely estimate states of size $\mathcal{O}(10^9)$ with an ensemble size of $\mathcal{O}(10)$. Sensible choice of the prior distribution, here taken to be weakly informative independent distributions in each dimension, is still an open question. Since the proposed framework learns the hyperparameters of the feature distribution, the number of random feature samples M is a free parameter. In practice, it is often advantageous to take M smaller for hyperparameter learning (where accuracy is less important) and take M larger for prediction tasks (where accuracy is the goal). To explore the practical significance of such benefits, comparison with other fixed-feature approaches is an interesting direction for further work, as is elucidating the relationship to large scale hyperparameter transfer methods (Yang and Hu, 2020; Yang et al., 2022)

Along related lines, a fruitful area for future development is the design of alternative objective functions for hyperparameter tuning that further improve optimizer performance and accelerate convergence. Objective functions not motivated from Bayesian arguments exist for kernel methods and are based on cross-validation (Owhadi and Yoo, 2019; Chen et al., 2021; Naslidnyk et al., 2023). Such objective functions have proven successful in derivative-based settings for deterministic kernels and could also be recast for random objective functions such as those required in the present derivative-free framework. The development of theoretical guarantees, like those in Chen et al. (2021), is also of great importance. Furthermore, since EKI solves an inverse problem, misfits appearing in the objective function need not be based on directly observed input-output pairs. This opens the door to tuning hyperparameters from *indirect* data. Learning from indirect data is useful in many applications where one cannot directly observe the outputs of the model. For example, embedding a machine learning model into the right hand side of a dynamical system will often require online training for stability and performance, yet the input and output of the model is not observable (Pahlavan et al., 2024; Wu et al., 2024).

5.2 Outlook

Broadly, the present paper intends to act as a case study within a more general framework for learning distributions that are sampled in randomized algorithms. Besides the random features studied here, other machine learning tools that follow a randomization approach (Scardapane and Wang, 2017) include Nyström methods (Williams and Seeger,

2000; Sun et al., 2015; Meanti et al., 2022) and reservoir computers (Gauthier, 2018; Gauthier et al., 2021; Griffith et al., 2019). Common patterns to tune the hyperparameters of such methods often follow the existing literature on random features: employing manual calibration, fitting to fixed realizations of random variables, and restricting to a narrow choice of algorithm- and problem-specific hyperparameters. The proposed framework is a principled and automated alternate strategy. Moreover, these fields have witnessed other Bayesian optimization methods learn low-dimensional (e.g., fewer than 10) hyperparameters effectively (Griffith et al., 2019; Racca and Magri, 2021; Antonik et al., 2023). Since EKI is highly scalable when compared with such Gaussian process-based optimizers, it may enable the learning of much higher-dimensional hyperparameters. We remark that recent work also suggests Bayesian optimization methods may be scaled to high input and output spaces (Hvarfner et al., 2024; Xu et al., 2024; Maddox et al., 2021).

A parallel literature uses derivative-free evolutionary algorithms to search for deterministic network architectures in a process termed neural evolution and claims superiority over grid search and random search (Bergstra and Bengio, 2012) over discrete domains (Real et al., 2017; Stanley et al., 2019; Xiao et al., 2020). Such algorithms are also applied at times to continuum hyperparameter domains in low dimensions. All such methods are susceptible to the curse of dimensionality, however. Future work that systematically compares the effectiveness of EKI against Bayesian optimizers, evolutionary algorithms, and random search would be of great value to the field and more decisively determine whether EKI is better suited for higher-dimensional hyperparameters. Altogether, the methodological framework that the present paper develops lays the groundwork for novel automated approaches to hyperparameter calibration on different scales than the existing literature, while simultaneously providing robust and user-friendly software.

Code Availability

The code and examples for this work are available in three open-source software packages written in the Julia programming language. `RandomFeatures.jl v0.3.4` contains a range of different random feature families, `EnsembleKalmanProcesses.jl v2.0.1` contains a range of EKI variants and utilities for creating and solving inverse problems, and `CalibrateEmulateSample.jl v0.6` contains the code for the application problems presented in this paper.

Acknowledgments and Disclosure of Funding

ORAD is supported by Schmidt Sciences, LLC, the National Science Foundation (NSF) under award number AGS-1835860, and the Office of Naval Research (ONR) under award number N00014-23-1-2654. NHN acknowledges support from NSF award number DMS-2402036, the NSF Graduate Research Fellowship Program under award number DGE-1745301, the Amazon/Caltech AI4Science Fellowship, the Air Force Office of Scientific Research under MURI award number FA9550-20-1-0358 (Machine Learning and Physics-Based Modeling and Simulation), and the Department of Defense Vannevar Bush Faculty Fellowship held by Andrew M. Stuart under ONR award number N00014-22-1-2790. The computations pre-

sented in this paper were partially conducted on the Resnick High Performance Computing Center, a facility supported by the Resnick Sustainability Institute at the California Institute of Technology. The authors are grateful to Daniel J. Gauthier for helpful comments on a previous version of the paper and to the two anonymous referees for their valuable feedback.

Appendix A. Additional Numerical Experiment Details

In this appendix, we add some additional notes about the experimental configurations to facilitate reproducibility. Appendix A.1 discusses hyperparameter priors for the chosen feature distribution parametrizations, while the settings for the EKI algorithm are collected in Appendix A.2. The applications in Section 4 were chosen because they require satisfactory performance off of training data. In this appendix, we supplement these scientific applications with auxiliary investigations into the heuristics of running the proposed hyperparameter tuning algorithm. Summarizing the additional results found in Appendices A.3, A.4, and A.5, we find that few features—fewer than training data set size—are sufficient to obtain high-performing tuned hyperparameter, and the structure and rank of the feature distributions are important but can be selected with using a held-out validation data set.

Finally, we note that all numerical experiments are reproducible at

`https://github.com/CliMA/CalibrateEmulateSample.jl`.

Users may interact with this code base as a playground to understand our EKI-based algorithm and how its various setup parameters influence performance in the context of emulator training. Documentation can be found here:

`https://clima.github.io/CalibrateEmulateSample.jl/dev/`.

Readers should navigate to `Examples` → `Emulator testing`.

A.1 Hyperparameter Priors

To apply EKI, prior probability distributions must be placed on learnable parameters. From a practical perspective, priors serve as a method of introducing information, such as enforcing bounds and setting the initial ensemble span and correlation structure. In particular, for linear problems, the standard EKI update will be confined to the span of the initial ensemble. Thus, prior specification is a crucial component part of the framework.

We choose to use a Gaussian prior distribution due to its relationship with a squared-exponential kernel (Rahimi and Recht, 2007) and find that performance of the algorithm depends strongly on the choice of the prior's covariance structure. Desiring a feature distribution that is flexible yet scales better than the naïve $\mathcal{O}((d+p)^2)$ parameters of a Cholesky factored covariance, the low-rank perturbation structure from Section 3.4 was used in all applications:

$$\begin{aligned}\varphi(x;(\varsigma,U,S)) &= \sqrt{\varsigma}\cos(\Xi x + B), \quad \text{where}\\ (\Xi, B) &\sim \mathsf{N}\big(0, (I+USU^\top)(I+USU^\top)^\top\big) \otimes \mathsf{Unif}([0,2\pi]^p).\end{aligned}$$

We also compare this class of nonseparable feature distributions with separable feature distributions in multi-output experiments. The class of separable distributions we chose factors both the input and output space, leading to

$$\begin{aligned}\varphi(x;(\varsigma,V_1,T_1,V_2,T_2)) &= \sqrt{\varsigma}\cos(\Xi x + B), \quad \text{where}\\ (\Xi,B) &\sim \mathrm{MatrixNormal}\big(0,C_{\mathrm{in}},C_{\mathrm{out}}\big)\otimes \mathsf{Unif}([0,2\pi]^p)\,,\\ C_{\mathrm{in}} &\coloneqq (I+V_1T_1V_1^\top)(I+V_1T_1V_1^\top)^\top\,, \quad \text{and}\\ C_{\mathrm{out}} &\coloneqq (I+V_2T_2V_2^\top)(I+V_2T_2V_2^\top)^\top.\end{aligned}$$

This separable kernel places a covariance structure on inputs (subscript 1) and outputs (subscript 2) independently and is more restrictive than the nonseparable distribution. However, the separable case has fewer hyperparameters. In the preceding displays, the matrices S, T_1, and T_2 are all diagonal.

To enforce positivity of ς and the entries of the diagonal matrices, we endow them with independent log-normal prior distributions with 99% of prior support covering the interval $(10^{-3}, 10^3)$. For the entries U_{ij}, we took independent Gaussian prior distributions with 99% of the prior support covering $(-300, 300)$. In the different experiments, the dependence on these prior ranges was quite weak. They were observed however to influence the speed of algorithm convergence when the prior span was overly concentrated or overly spread by several orders of magnitude.

A.2 Common EKI Configuration

Besides the prior, we also detail the common EKI configuration parameters used across all three application experiments. In the experiments, we chose to use the adaptive learning rate scheduler from the work of Iglesias and Yang (2021). The adaptive value of step size Δt_{n+1} provides a large step when bounding Jeffrey's divergence between the distributions represented by the ensemble at times t_n and t_{n+1}. This prevents accumulation of errors due to ill-conditioned updates with large Δt. A small amount of additive white noise (i.e., inflation) was added to the particles at each iteration to prevent degeneracy in the ensemble exploration. Additionally, the optimizer was more robust under multiple validation partitions Ω^{c} as described in Subsection 3.3. We chose to stack two partitions of the data during training and took $\mathsf{K}/N = 0.2$ for all experiments except in Subsection 4.3, where we set $\mathsf{K}/N = 0.1$. The input and output data was normalized before training. In particular, a whitening matrix was applied so that the input samples had empirical mean zero and unit empirical variance. The $\Gamma = \Gamma(u)$ in Subsection 3.3 was always estimated at a fixed value of $u = u^\dagger$ taken as the mean of the prior. The output data was also whitened so that the variability matrix Γ had identity covariance over the output data. An optimal linear shrinkage estimator was used to ensure a well-conditioned matrix (Ledoit and Wolf, 2004).

A.3 Details for Section 4.1: Global Sensitivity Analysis

To tune the hyperparameters in the Ishigami experiment, the ensemble size was taken to be $J = 30$ for the 13 parameters $\#\{\varsigma, \{U_{ij}\}_{ij}, \{D_{ii}\}_i\} = (1, 9, 3)$. The number of features used while optimizing the hyperparameters was taken to be 150. In Figure 7, the convergence

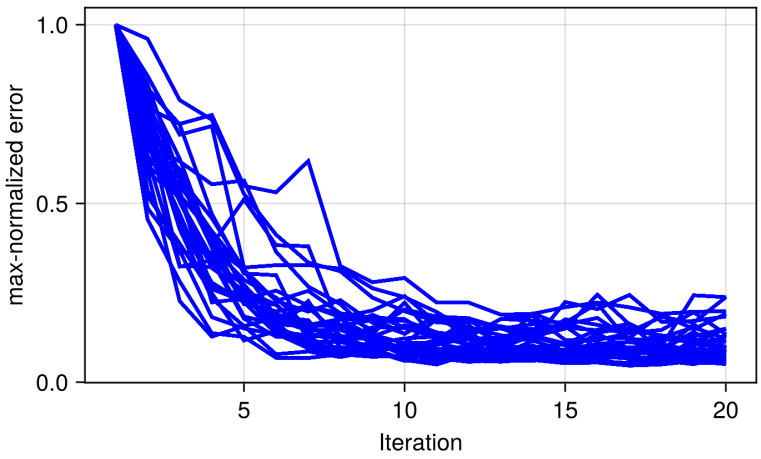

Figure 7: Convergence of the error $n_{\text{iter}}^{-1} \sum_{n=1}^{n_{\text{iter}}} \|\Gamma^{-1/2}(z_n - \mathcal{G}(u_n))\|^2$ for 20 different tuning experiments with scalar-valued random features used to learn the Ishigami function for global sensitivity analysis. The initial objective value is normalized to one in the plot. For this experiment, convergence stagnates at around eight ensemble iterations.

is plotted over 20 iterations and under 20 restarts of the algorithm with different random samplings from the prior.

To tune the hyperparameters in the G-function experiment of dimension d, the ensemble size was taken to be 200 across all dimensional experiments. For the parameters of the rank-r feature distribution, $\#\{\varsigma, \{U_{ij}\}_{ij}, \{S_{ii}\}_i\} = (1, 20r, r)$, we took rank $r = \min(10, d)$. The number of features used while optimizing the hyperparameters was taken to be $M = 300$. The optimization was run for 20 iterations with a fixed ensemble size of 200 over all experiments. We do not plot the convergence for all experiments for brevity, but report that the normalized objective value was reduced to under 10^{-1} over the iterations.

Table 3 displays timings of the tuning and fitting process for the Sobol G-Function against different input dimensions. Both GP and RF methods are compared. Since the algorithms are not reaching the same convergence criteria and performance may depend heavily on implementation details of numerical linear algebra subroutines under-the-hood, such timings are qualitative. However, these results provide evidence of the poor scaling of GP hyperparameter tuning even in moderate dimensions and even with a fixed number of input data points. On the other hand, EKI scales linearly and is also able to take advantage of multithreading—though some overheads are seen here.

A.3.1 Performance Dependence on Number of Features

A useful perk that the proposed hyperparameter tuning framework for RFs can utilize is a decoupling of the number of features M required during tuning from the number of features $M' \neq M$ required during prediction. The timings and performance of the algorithm for the six-dimensional Sobol G-function over varying numbers of features M used in tuning is displayed in Table 4. The error here represents the L^2 difference between the predicted RF

Table 3: Mean time to train (tune and fit) the machine learning algorithms, averaged over 30 repeated experiments for learning the Sobol G-function for global sensitivity analysis. Note this comparison runs the default convergence criteria for L-BFGS-B from `SciKitLearn.jl` and runs EKI for 20 iterations using 200 ensemble members. ⋆The final GP training was canceled after 10 hours. The memory for both tools in serial was 2GB, while EKI in parallel used approximately 6GB.

d	GP (L-BFGS-B)	RF (EKI) – Serial	RF (EKI) – 8 Threads
3	16s	46s	12s
6	309s	65s	23s
10	2653s	134s	38s
20	>10h⋆	270s	67s

Table 4: The tuned performance for the six-dimensional Sobol G-function emulators as different numbers of features M are taken during tuning. The training and test errors are the average L^2 difference between the tuned RF mean (created always with $M' = 1000$ features) and noiseless data at training and test points. The final row corresponds to the GP emulator. The time taken for the tuning process is also shown.

M (When Tuning)	Optimizer Time (Serial)	Training Error	Test Error
25	7s	0.014420	0.004687
50	13s	0.011280	0.004069
100	30s	0.007872	0.003643
200	72s	0.007720	0.003797
400	204s	0.006297	0.003535
800	700s	0.005756	0.003660
∞ (GP Limit)	320s	0.000358	0.002827

mean (with $M' = 1000$ features) and the noiseless data, averaged either over the fit inputs (training error) or the other sampling points (test error). In either case, it is observed that increasing the number of features leads to a scaling in time between linear and quadratic, and a reduction of both the training and test error. However, there is clear stagnation in performance with respect to both training and test error when the algorithm exceeds $M = 100$ features. This is potentially due to sample error domination or optimization effects.

A.4 Details for Section 4.2: Lorenz 63 Integrator

To tune the hyperparameters in the chaotic dynamics experiment, the EKI ensemble size was taken to be 42 for the 41 parameters $\#\{\varsigma, \{U_{ij}\}_{ij}, \{S_{ii}\}_i\} = (1, 36, 4)$ in the rank-4 nonseparable feature distribution. The number of features used while optimizing the hyperparameters was taken to be $M = 200$. The convergence is plotted over 20 iterations and under 20 restarts of the algorithm with different random samplings from the prior in

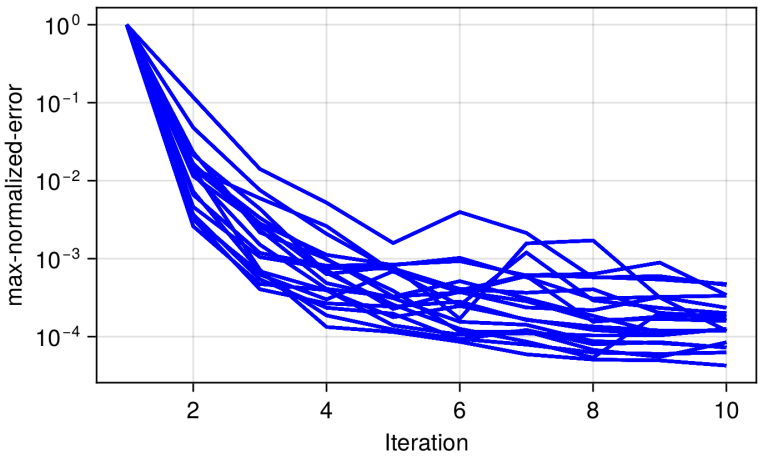

Figure 8: Convergence of the error $n_{\text{iter}}^{-1} \sum_{n=1}^{n_{\text{iter}}} \|\Gamma^{-1/2}(z_n - \mathcal{G}(u_n))\|^2$ for 20 independent experiments on the Lorenz 63 system, plotted on a log-scale. The initial objective value is normalized to one. For this experiment, convergence stagnates at around 8 ensemble iterations. Iterations for this plot contain 42 ensemble members.

Figure 8. An experiment run with a larger noise level in the training data is shown in Figure 9; this figure should be compared with Figure 4.

A.4.1 Performance Dependence on Feature Distribution Structure

To illustrate the dependence on the rank under optimization, the Lorenz 63 experiment of Section 4.2 is repeated across all ranks $1, \ldots, 9$ for the nonseparable feature distribution and all ranks $(1,1), \ldots, (3,3)$ for the separable feature distribution, taking all other factors (such as the number of features or optimizer options) fixed. The optimization took an mean time of 114 seconds largely independent of the rank or separability structure. In Figure 10, the resulting L^2-error of the predicted mean at each time step with both training and test data (and normalized by trajectory length) are plotted against the rank. The plot was obtained from averages over 10 re-initializations of the optimizer. We include the performance of the GP for reference. For the nonseparable feature distribution structure, the increase in rank shows large benefits to performance on the training data until a plateau occurs at higher ranks. This motivates our choice of taking the rank equal to four in the experiments because increasing the rank quadratically increases the number of tunable parameters. The separable structure is able to achieve similar performance when the input is full rank and poor performance otherwise. Both are able to obtain performance similar to that of the GP reference, as observed in an application to forecast the attractor. We observe that in both experiments, qualitative behavior exhibited on the training trajectory is reflected in the test trajectory.

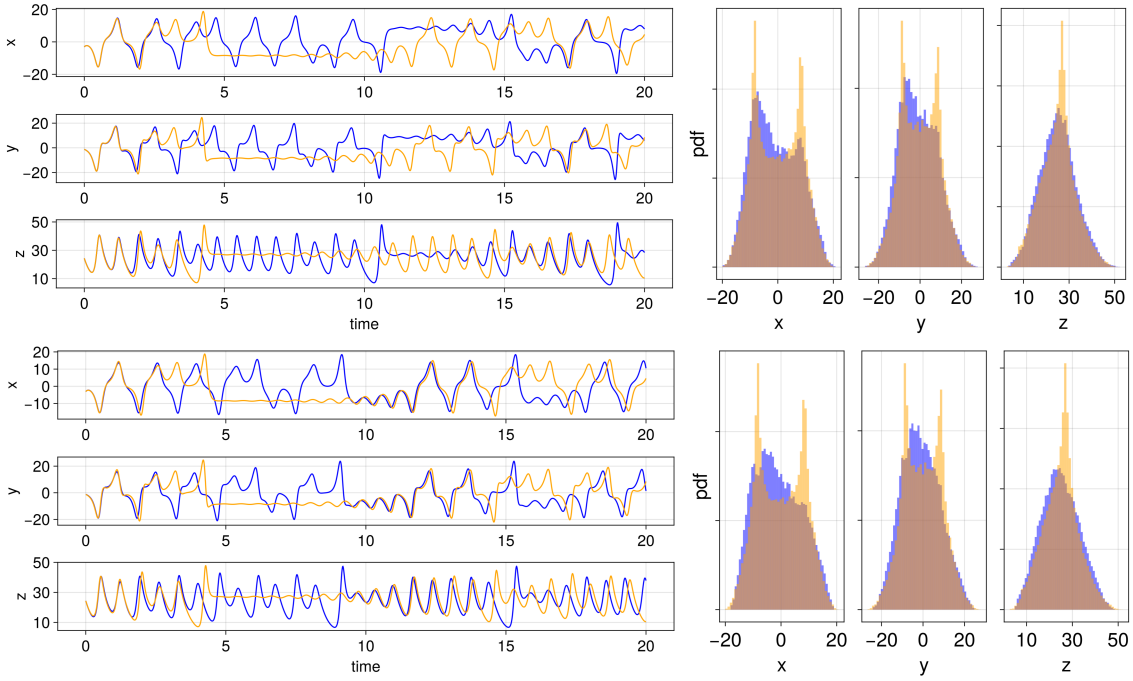

Figure 9: Row one GP and row two RF experiments for the Lorenz 63 system as in Figure 4 (cf. rows two and three there), but with increased noise level corresponding to noise covariance matrix $\Sigma = 10^{-2}I$.

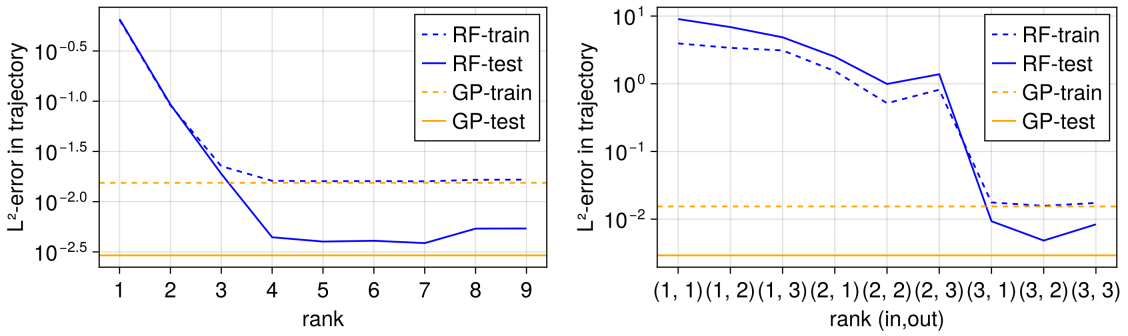

Figure 10: Results on changing the rank $r = 1, \ldots, 9$ of the nonseparable feature distribution (left plot) covariance (48) and of the separable feature distribution covariance (right plot) with input and output ranks $r = (1,1), \ldots, (3,3)$ for the Lorenz 63 system. On each plot, the two curves represent the L^2 error of predicted time steps along the training (dashed) and the test (solid) trajectories. The GP results (independent of rank) are shown for comparison in orange alongside RF results in blue. The vertical axis shows the L^2 error of the tuned emulator against the rank on the horizontal axis.

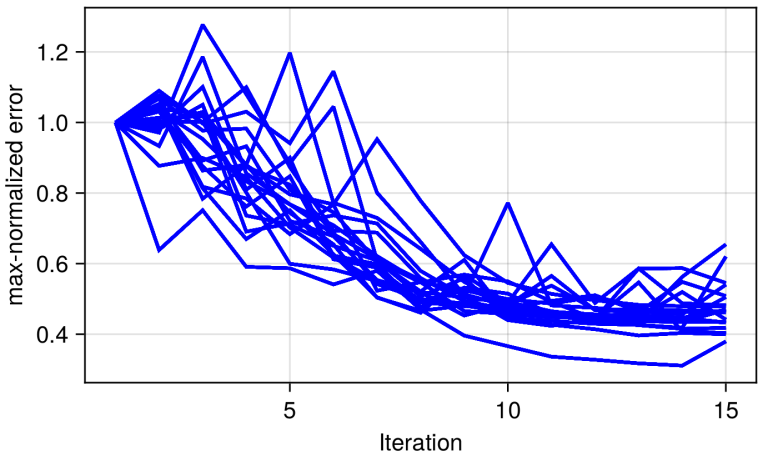

Figure 11: Convergence of the error $n_{\text{iter}}^{-1} \sum_{n=1}^{n_{\text{iter}}} \|\Gamma^{-\frac{1}{2}}(z_n - \mathcal{G}(u_n))\|^2$ for tuning the RF emulator used in uncertainty quantification of a cloud model, normalized to be initially one over 20 experiments. For this experiment, convergence stagnates at around 12 ensemble iterations. Iterations for this plot contain 635 ensemble members.

Table 5: Priors for the parameters for the EDMF calibration experiment. Priors are defined by mapping unit variance normal distributions into the desired bounds with a shifted and scaled logit function.

Parameter	Bounds	Prior Mean
`entrainment_factor`	$[0, 1]$	0.13
`detrainment_factor`	$[0, 1]$	0.51
`tke_ed_coeff`	$[0.01, 1]$	0.14
`tke_diss_coeff`	$[0.01, 1]$	0.22
`static_stab_coeff`	$[0.01, 1]$	0.40

A.5 Details for Section 4.3: Uncertainty Quantification of a Cloud Model

To tune the hyperparameters in the Bayesian inverse problem experiment involving the cloud model, the ensemble size was taken to be $J = 635$ for the 634 parameters given by $\#\{\varsigma, \{U_{ij}\}_{ij}, \{S_{ii}\}_i\} = (1, 630, 3)$. The number of features used while optimizing the hyperparameters was taken to be $M = 200$. The convergence is plotted over 20 iterations of EKI and under 20 restarts of the algorithm with different random samplings from the prior in Figure 11. The priors for the physical parameters in the underlying geophysical inverse problem (not to be confused with the hyperparameter priors for the forward map surrogate) are given in Table 5.

For the MCMC algorithm, we use a simple random walk Metropolis method. The step size was tuned in each experiment to achieve an acceptance rate of approximately 25%. The MCMC method is initialized in a region of high posterior mass and the GP selected a

Table 6: Performance of the different emulators on the accelerated uncertainty quantification test problem, measured as average L^2-error over the training data and over a held-out test set, under 10 restarts of the algorithm. Reported timings are for the tuning procedure, and the average time to perform one MCMC likelihood calculation. The rank of the separable or nonseparable distribution are indicated in parentheses in column one.

Emulator	Tuning Time	Training Error	Test Error	MCMC Step Time
GP (L-BFGS-B)	260s	4.85	15.08	7.8ms
RF (untuned), rank 1	—	15.80	12.61	3.0ms (8 threads)
RF (EKI), rank $(5, 1)$	504s (8-threads)	16.77	14.05	3.8ms (8 threads)
RF (EKI), rank 1	565s (8-threads)	12.81	11.70	3.0ms (8 threads)

smaller step size than the RF did by a factor of approximately three. We ran the algorithm until the posteriors appeared to converge, for a total of 2×10^5 steps for RF and 5×10^5 steps for GP. Table 6 compares the time taken for tuning and prediction and also displays training and test errors with different feature distributions or with GPs. The ranks chosen for the RF method were based on having low test error (as done in Appendix A.4.1). The test set was held out of the final iteration of data from the training point selection algorithm (Cleary et al., 2021); by construction, test accuracy is thus evaluated on a region of high posterior mass that the MCMC samples will explore. Table 6 shows more consistent performance on training and test error with RF, while the GP results show a much lower error on training data than on test data, indicating possible overfitting.

In Figure 12, the panels show the performance of RF-CES posteriors based on a separable feature distribution and on an untuned RF emulator, in addition to the emulator with nonseparable feature distribution seen in Subsection 4.3 and Figure 6 in the main text. As expected, the posterior corresponding to the untuned emulator performs poorly; its samples cover much of the geophysical parameter prior support; see Table 5. However, this is somewhat surprising in light of Table 6 because the untuned RF emulator obtained low test error there. This implies that low emulator test error for the forward map does not automatically translate to an accurate posterior for the inverse problem. The RF emulator with separable feature distribution also produces a posterior with too wide of a support compared to the GP-CES reference posterior. Finally, we observe that although the GP emulator obtained large forward map test error compared to that of our chosen nonseparable RF emulator, the resulting posterior distributions remained similar as seen in Figure 6.

References

S. Ambikasaran, M. O'Neil, and K. R. Singh. Fast symmetric factorization of hierarchical matrices with applications. *preprint arXiv.1405.0223*, 2016. (cited on p. 18)

P. Antonik, N. Marsal, D. Brunner, and D. Rontani. Bayesian optimisation of large-scale photonic reservoir computers. *Cognitive Computation*, pages 1–9, 2023. (cited on p. 31)

G. E. B. Archer, A. Saltelli, and I. M. Sobol. Sensitivity measures, ANOVA-like techniques and the use of bootstrap. *Journal of Statistical Computation and Simulation*, 58(2):

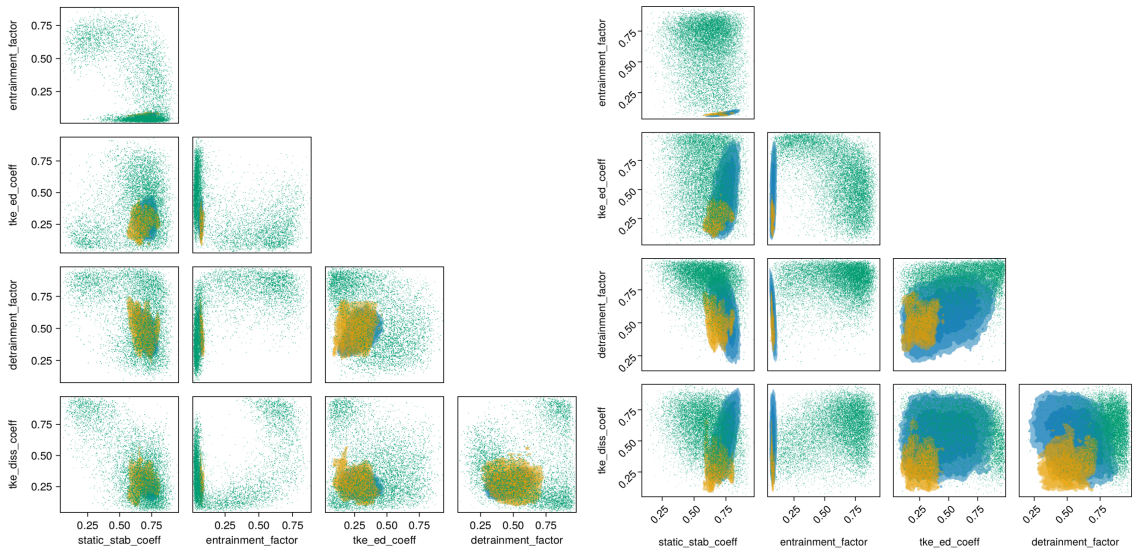

Figure 12: Additional plots for the uncertainty quantification of a cloud model. The corner plots display marginal posterior distributions resulting from emulators using GP-CES (yellow), RF-CES (blue), and RF-CES without tuning (green scatter). Left: rank-1 non-separable feature distribution in RF-CES, as shown in Figure 6. Right: separable feature distribution in RF-CES with rank (5,1).

99–120, 1997. (cited on p. 21)

J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012. (cited on p. 31)

C. M. Bishop and N. M. Nasrabadi. *Pattern recognition and machine learning*. Springer, 2006. (cited on p. 8)

E. Bollt. On explaining the surprising success of reservoir computing forecaster of chaos? The universal machine learning dynamical system with contrast to VAR and DMD. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 31(1):013108, 2021. (cited on p. 2)

L. Böttcher. Gradient-free training of neural ODEs for system identification and control using ensemble Kalman inversion. *preprint arXiv:2307.07882*, 2023. (cited on p. 3)

R. Brault, M. Heinonen, and F. Buc. Random Fourier features for operator-valued kernels. In *Asian Conference on Machine Learning*, pages 110–125. PMLR, 2016. (cited on p. 10)

E. Calvello, S. Reich, and A. M. Stuart. Ensemble Kalman Methods: A Mean Field Perspective. *Acta Numerica*, 2025. (cited on pp. 3, 14, and 15)

A. Chattopadhyay, P. Hassanzadeh, and D. Subramanian. Data-driven predictions of a multiscale Lorenz 96 chaotic system using machine-learning methods: Reservoir computing,

artificial neural network, and long short-term memory network. *Nonlinear Processes in Geophysics*, 27(3):373–389, 2020. (cited on p. 2)

Y. Chen and D. S. Oliver. Ensemble randomized maximum likelihood method as an iterative ensemble smoother. *Mathematical Geosciences*, 44:1–26, 2012. (cited on p. 3)

Y. Chen, H. Owhadi, and A. M. Stuart. Consistency of empirical Bayes and kernel flow for hierarchical parameter estimation. *Mathematics of Computation*, 90(332):2527–2578, 2021. (cited on pp. 12 and 30)

E. Cleary, A. Garbuno-Inigo, S. Lan, T. Schneider, and A. M. Stuart. Calibrate, emulate, sample. *Journal of Computational Physics*, 424:109716, 2021. ISSN 0021-9991. (cited on pp. 3, 10, 27, and 39)

S. V. Dudul. Prediction of a Lorenz chaotic attractor using two-layer perceptron neural network. *Applied Soft Computing*, 5(4):333–355, 2005. ISSN 1568-4946. (cited on pp. 23 and 26)

O. R. Dunbar, A. B. Duncan, A. M. Stuart, and M.-T. Wolfram. Ensemble inference methods for models with noisy and expensive likelihoods. *SIAM Journal on Applied Dynamical Systems*, 21(2):1539–1572, 2022a. (cited on p. 3)

O. R. A. Dunbar, A. Garbuno-Inigo, T. Schneider, and A. M. Stuart. Calibration and Uncertainty Quantification of Convective Parameters in an Idealized GCM. *Journal of Advances in Modeling Earth Systems*, 13(9):e2020MS002454, 2021. (cited on pp. 3 and 27)

O. R. A. Dunbar, I. Lopez-Gomez, A. Garbuno-Iñigo, D. Z. Huang, E. Bach, and J.-l. Wu. EnsembleKalmanProcesses.jl: Derivative-free ensemble-based model calibration. *Journal of Open Source Software*, 7(80):4869, 2022b. (cited on pp. 3, 4, and 15)

O. R. A. Dunbar, M. Bieli, A. Garbuno-Iñigo, M. Howland, A. N. Souza, L. A. Mansfield, G. L. Wagner, and N. Efrat-Henrici. CalibrateEmulateSample.jl: Accelerated Parametric Uncertainty Quantification. *Journal of Open Source Software*, 9(97):6372, 2024. (cited on p. 4)

P. Dutilleul. The MLE algorithm for the matrix normal distribution. *Journal of Statistical Computation and Simulation*, 64(2):105–123, 1999. (cited on p. 18)

R. Fablet, S. Ouala, and C. Herzet. Bilinear residual neural network for the identification and forecasting of geophysical dynamics. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pages 1477–1481, 2018. (cited on pp. 23 and 26)

J. I. T. Falk, C. Cilibert, and M. Pontil. Implicit kernel meta-learning using kernel integral forms. In *Uncertainty in Artificial Intelligence*, pages 652–662. PMLR, 2022. (cited on p. 3)

A. Garbuno-Inigo, F. Hoffmann, W. Li, and A. M. Stuart. Interacting Langevin diffusions: Gradient structure and ensemble Kalman sampler. *SIAM Journal on Applied Dynamical Systems*, 19(1):412–441, 2020a. (cited on pp. 14 and 15)

A. Garbuno-Inigo, N. Nüsken, and S. Reich. Affine invariant interacting Langevin dynamics for Bayesian inference. *SIAM Journal on Applied Dynamical Systems*, 19(3):1633–1658, 2020b. (cited on pp. 14 and 15)

D. J. Gauthier. Reservoir computing: Harnessing a universal dynamical system. *SIAM News*, 51(2):12, 2018. (cited on p. 31)

D. J. Gauthier, E. Bollt, A. Griffith, and W. A. Barbosa. Next generation reservoir computing. *Nature Communications*, 12(1):5564, 2021. (cited on pp. 2 and 31)

A. Gelman, D. Simpson, and M. Betancourt. The prior can often only be understood in the context of the likelihood. *Entropy*, 19(10), 2017. ISSN 1099-4300. (cited on p. 19)

A. Griffith, A. Pomerance, and D. J. Gauthier. Forecasting chaotic systems with very low connectivity reservoir computers. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 29(12):123108, 2019. (cited on p. 31)

R. Hamid, Y. Xiao, A. Gittens, and D. DeCoste. Compact random feature maps. In E. P. Xing and T. Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 19–27, Bejing, China, 22–24 Jun 2014. PMLR. (cited on p. 2)

J. Hensman, A. Matthews, and Z. Ghahramani. Scalable variational Gaussian process classification. In *Artificial Intelligence and Statistics*, pages 351–360. PMLR, 2015. (cited on p. 7)

J. Hensman, N. Durrande, and A. Solin. Variational Fourier features for Gaussian processes. *Journal of Machine Learning Research*, 18(151):1–52, 2018. (cited on p. 7)

D. Z. Huang, J. Huang, S. Reich, and A. M. Stuart. Efficient derivative-free Bayesian inference for large-scale inverse problems. *Inverse Problems*, 38(12):125006, oct 2022. (cited on pp. 3 and 14)

G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1-3):489–501, 2006. (cited on pp. 1 and 2)

C. Hvarfner, E. O. Hellsten, and L. Nardi. Vanilla Bayesian optimization performs great in high dimension. *preprint arXiv:2402.02229*, 2024. (cited on p. 31)

M. Iglesias and Y. Yang. Adaptive regularisation for ensemble Kalman inversion. *Inverse Problems*, 37(2):025008, 2021. (cited on p. 33)

M. A. Iglesias, K. J. Law, and A. M. Stuart. Ensemble kalman methods for inverse problems. *Inverse Problems*, 29(4):045001, 2013. (cited on pp. 3 and 14)

T. Ishigami and T. Homma. An importance quantification technique in uncertainty analysis for computer models. In *First International Symposium on Uncertainty Modeling and Analysis*, pages 398–403. IEEE, 1990. (cited on p. 20)

H. Kadri, E. Duflos, P. Preux, S. Canu, A. Rakotomamonjy, and J. Audiffren. Operator-valued kernels for learning from functional response data. *Journal of Machine Learning Research*, 17(20):1–54, 2016. (cited on p. 10)

A. Kammonen, J. Kiessling, P. Plecháč, M. Sandberg, and A. Szepessy. Adaptive random Fourier features with Metropolis sampling. *Foundations of Data Science*, 2(3):309–332, 2020. (cited on p. 3)

A. Kammonen, J. Kiessling, P. Plecháč, M. Sandberg, A. Szepessy, and R. Tempone. Smaller generalization error derived for a deep residual neural network compared with shallow networks. *IMA Journal of Numerical Analysis*, 43(5):2585–2632, 2023. (cited on p. 3)

M. Kanagawa, P. Hennig, D. Sejdinovic, and B. K. Sriperumbudur. Gaussian processes and kernel methods: A review on connections and equivalences. *preprint arXiv:1807.02582*, 2018. (cited on p. 6)

N. B. Kovachki and A. M. Stuart. Ensemble Kalman inversion: a derivative-free technique for machine learning tasks. *Inverse Problems*, 35(9):095005, 2019. (cited on p. 3)

S. Lanthaler and N. H. Nelsen. Error bounds for learning with vector-valued random features. In A. Oh, T. Neumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 71834–71861. Curran Associates, Inc., 2023. (cited on pp. 2, 5, 9, and 11)

Q. Le, T. Sarlós, A. Smola, et al. Fastfood-approximating kernel expansions in loglinear time. In *Proceedings of the International Conference on Machine Learning*, volume 85, 2013. (cited on pp. 2 and 18)

O. Ledoit and M. Wolf. A well-conditioned estimator for large-dimensional covariance matrices. *Journal of Multivariate Analysis*, 88(2):365–411, 2004. (cited on p. 33)

C.-L. Li, W.-C. Chang, Y. Mroueh, Y. Yang, and B. Poczos. Implicit kernel learning. In K. Chaudhuri and M. Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 2007–2016. PMLR, 16–18 Apr 2019. (cited on p. 3)

F. Liu, X. Huang, Y. Chen, and J. A. K. Suykens. Random features for kernel approximation: A survey on algorithms, theory, and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):7128–7148, 2022. (cited on p. 2)

I. Lopez-Gomez, C. Christopoulos, H. L. Langeland Ervik, O. R. A. Dunbar, Y. Cohen, and T. Schneider. Training Physics-Based Machine-Learning Parameterizations With Gradient-Free Ensemble Kalman Methods. *Journal of Advances in Modeling Earth Systems*, 14(8):e2022MS003105, 2022. (cited on pp. 27 and 28)

E. N. Lorenz. Deterministic nonperiodic flow. *Journal of Atmospheric Sciences*, 20(2): 130–141, 1963. (cited on p. 22)

W. J. Maddox, M. Balandat, A. G. Wilson, and E. Bakshy. Bayesian optimization with high-dimensional outputs. In *Advances in Neural Information Processing Systems*, volume 34, pages 19274–19287, 2021. (cited on p. 31)

G. Meanti, L. Carratino, E. De Vito, and L. Rosasco. Efficient hyperparameter tuning for large scale kernel ridge regression. In *International Conference on Artificial Intelligence and Statistics*, pages 6554–6572. PMLR, 2022. (cited on p. 31)

N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21 (6):1087–1092, 1953. (cited on p. 27)

C. Micchelli and M. Pontil. Kernels for multi–task learning. In *Advances in Neural Information Processing Systems*, volume 17, 2004. (cited on p. 10)

C. A. Micchelli and M. Pontil. On learning vector-valued functions. *Neural Computation*, 17(1):177–204, 2005. (cited on p. 10)

M. Naslidnyk, M. Kanagawa, T. Karvonen, and M. Mahsereci. Comparing Scale Parameter Estimators for Gaussian Process Regression: Cross Validation and Maximum Likelihood. *preprint arXiv:2307.07466*, 2023. (cited on pp. 12 and 30)

N. H. Nelsen and A. M. Stuart. The random feature model for input-output maps between Banach spaces. *SIAM Journal on Scientific Computing*, 43(5):A3212–A3243, 2021. (cited on pp. 5 and 10)

H. Owhadi. Do ideas have shape? Idea registration as the continuous limit of artificial neural networks. *Physica D: Nonlinear Phenomena*, art. 133592, 2022. (cited on p. 10)

H. Owhadi and G. R. Yoo. Kernel flows: From learning kernels from data into the abyss. *Journal of Computational Physics*, 389:22–47, 2019. ISSN 0021-9991. (cited on p. 30)

H. A. Pahlavan, P. Hassanzadeh, and M. J. Alexander. Explainable Offline-Online Training of Neural Networks for Parameterizations: A 1D Gravity Wave-QBO Testbed in the Small-Data Regime. *Geophysical Research Letters*, 51(2):e2023GL106324, 2024. (cited on p. 30)

A. Racca and L. Magri. Robust optimization and validation of echo state networks for learning chaotic dynamics. *Neural Networks*, 142:252–268, 2021. ISSN 0893-6080. (cited on p. 31)

A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, volume 20. Citeseer, 2007. (cited on pp. 2, 7, 18, and 32)

A. Rahimi and B. Recht. Uniform approximation of functions with random bases. In *2008 46th Annual Allerton Conference on Communication, Control, and Computing*, pages 555–561. IEEE, 2008a. (cited on p. 7)

A. Rahimi and B. Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *Advances in Neural Information Processing Systems*, volume 21, 2008b. (cited on p. 7)

E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin. Large-scale evolution of image classifiers. In *International conference on machine learning*, pages 2902–2911. PMLR, 2017. (cited on p. 31)

A. Rudi and L. Rosasco. Generalization properties of learning with random features. *Advances in Neural Information Processing Systems*, 30, 2017. (cited on p. 7)

A. Saltelli. Sensitivity analysis for importance assessment. *Risk Analysis*, 22(3):579–590, 2002. (cited on p. 20)

A. Saltelli, P. Annoni, I. Azzini, F. Campolongo, M. Ratto, and S. Tarantola. Variance based sensitivity analysis of model output: Design and estimator for the total sensitivity index. *Computer Physics Communications*, 181(2):259–270, 2010. ISSN 0010-4655. (cited on p. 21)

D. Sanz-Alonso, A. M. Stuart, and A. Taeb. *Inverse problems and data assimilation*, volume 107. Cambridge University Press, 2023. (cited on p. 15)

S. Scardapane and D. Wang. Randomness in neural networks: an overview. *WIREs Data Mining and Knowledge Discovery*, 7(2):e1200, 2017. (cited on pp. 1 and 30)

S. Scher and G. Messori. Generalization properties of feed-forward neural networks trained on Lorenz systems. *Nonlinear Processes in Geophysics*, 26(4):381–399, 2019. (cited on pp. 23 and 26)

C. Schillings and A. M. Stuart. Analysis of the ensemble Kalman filter for inverse problems. *SIAM Journal on Numerical Analysis*, 55(3):1264–1290, 2017. (cited on pp. 3 and 14)

S. Shahi, F. H. Fenton, and E. M. Cherry. Prediction of chaotic time series using recurrent neural networks and reservoir computing techniques: A comparative study. *Machine Learning with Applications*, 8:100300, 2022. ISSN 2666-8270. (cited on pp. 23 and 26)

C. Sherlock, P. Fearnhead, and G. O. Roberts. The Random Walk Metropolis: Linking Theory and Practice Through a Case Study. *Statistical Science*, 25(2):172–190, 2010. ISSN 08834237. (cited on p. 27)

A. Sinha and J. C. Duchi. Learning kernels with random features. In *Advances in Neural Information Processing Systems*, volume 29, 2016. (cited on p. 2)

I. M. Sobol and Y. L. Levitan. On the use of variance reducing multipliers in Monte Carlo computations of a global sensitivity index. *Computer Physics Communications*, 117(1): 52–61, 1999. (cited on p. 20)

K. O. Stanley, J. Clune, J. Lehman, and R. Miikkulainen. Designing neural networks through neuroevolution. *Nature Machine Intelligence*, 1(1):24–35, 2019. (cited on p. 31)

A. M. Stuart. Inverse problems: a Bayesian perspective. *Acta Numerica*, 19:451–559, 2010. (cited on p. 6)

S. Sun, J. Zhao, and J. Zhu. A review of Nyström methods for large-scale machine learning. *Information Fusion*, 26:36–48, 2015. ISSN 1566-2535. (cited on pp. 2, 7, and 31)

M. Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *Artificial Intelligence and Statistics*, pages 567–574. PMLR, 2009. (cited on p. 7)

D. Vishny, M. Morzfeld, K. Gwirtz, E. Bach, O. R. A. Dunbar, and D. Hodyss. High-dimensional covariance estimation from a small number of samples. *Journal of Advances in Modeling Earth Systems*, 16(9):e2024MS004417, 2024. (cited on p. 30)

C. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems*, volume 13, 2000. (cited on p. 30)

C. K. Williams and C. E. Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT Press, Cambridge, MA, 2006. (cited on pp. 2 and 6)

J.-L. Wu, M. E. Levine, T. Schneider, and A. Stuart. Learning about structural errors in models of complex dynamical systems. *Journal of Computational Physics*, art. 113157, 2024. (cited on p. 30)

H. Xiao, J.-L. Wu, J.-X. Wang, R. Sun, and C. J. Roy. Quantifying and reducing model-form uncertainties in Reynolds-averaged Navier–Stokes simulations: A data-driven, physics-informed Bayesian approach. *Journal of Computational Physics*, 324:115–136, 2016. ISSN 0021-9991. (cited on p. 3)

X. Xiao, M. Yan, S. Basodi, C. Ji, and Y. Pan. Efficient hyperparameter optimization in deep learning using a variable length genetic algorithm. *preprint arXiv:2006.12703*, 2020. (cited on p. 31)

Z. Xu, H. Wang, J. M. Phillips, and S. Zhe. Standard Gaussian process can be excellent for high-dimensional Bayesian optimization. *preprint arXiv:2402.02746*, 2024. (cited on p. 31)

G. Yang and E. J. Hu. Feature learning in infinite-width neural networks. *preprint arXiv:2011.14522*, 2020. (cited on p. 30)

G. Yang, E. J. Hu, I. Babuschkin, S. Sidor, X. Liu, D. Farhi, N. Ryder, J. Pachocki, W. Chen, and J. Gao. Tensor programs V: Tuning large neural networks via zero-shot hyperparameter transfer. *preprint arXiv:2203.03466*, 2022. (cited on p. 30)

Z. Yang, A. Wilson, A. Smola, and L. Song. A la Carte – Learning Fast Kernels. In G. Lebanon and S. V. N. Vishwanathan, editors, *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, pages 1098–1106, San Diego, California, USA, 09–12 May 2015. PMLR. (cited on pp. 2 and 18)