Stable likelihood computation for machine learning of linear differential operators with Gaussian processes

Chatrabgoun, O., Esmaeilbeigi, M., Cheraghi, M. & Daneshkhah, A

DOI 10.1615/Int.J.UncertaintyQuantification.2022038966
ISSN 2152-5080
ESSN 2152-5099

Publisher: © BEGELL HOUSE Inc. 2023

Stable Likelihood Computation for Machine Learning of Linear Differential Operators with Gaussian Processes

O. Chatrabgoun[a,b], **M. Esmaeilbeigi**[a,*], **M. Cheraghi**[a], **A. Daneshkhah**[c]

[a] Faculty of Mathematical Sciences and Statistics, Malayer University, Malayer 65719-95863, Iran

[b] School of Computing, Electronics and Mathematics, Coventry University, Coventry, CV1 5FB, UK

[c] Centre for Computational Science and Mathematical Modelling, Coventry University, Coventry, CV1 5FB, UK

Abstract

In many applied sciences, the main aim is to learn the parameters in the operational equations which best fit the observed data. A framework for solving such problems is to employ Gaussian process (GP) emulators which are well-known as non-parametric Bayesian machine learning techniques. GPs are among a class of methods known as kernel machines which can be used to approximate rather complex problems by tuning their hyperparameters. The maximum likelihood estimation (MLE) has widely been used to estimate the parameters of the operators and kernels. However, the MLE-based and Bayesian inference in the standard form are usually involved in setting up a covariance matrix which is generally ill-conditioned. As a result, constructing and inverting the covariance matrix using the standard form will become unstable to learn the parameters in the operational equations. In this paper, we propose a novel approach to tackle these computational complexities and also resolve the ill-conditioning problem by forming the covariance matrix using alternative bases via the Hilbert-Schmidt SVD (HS-SVD) approach. Applying this approach yields a novel matrix factorization of the block-structured covariance matrix which can be implemented stably by isolating the main source of the ill-conditioning. In contrast to standard matrix decompositions which start with a matrix and produce the

*Corresponding author. E-mail address: m.esmaeilbeigi@malayeru.ac.ir

resulting factors, the HS-SVD is constructed from the Hilbert-Schmidt eigenvalues and eigenvectors without the need to ever form the potentially ill-conditioned matrix. We also provide stable MLE and Bayesian inference to adaptively estimate hyperparameters, and the corresponding operators can then be efficiently predicted at some new points using the proposed HS-SVD bases. The efficiency and stability of the proposed HS-SVD method will be compared with the existing methods by several illustrations of the parametric linear equations, such as ordinary and partial differential equations, integro-differential and fractional order operators.

Keyword: Gaussian processes; Hilbert–Schmidt's theory; Stable computation; Probabilistic machine learning; Uncertainty quantification.

1 Introduction

One of the major fields in applied sciences is to model different phenomena in terms of flexible operational equations [1, 2]. In other words, the researchers usually attempt to find a coherent form of flexible operational equations corresponding to the observed data to the effect that they best describe and govern them [3, 4]. Therefore, the necessity of existence of parametric operational equations is created, that is, the equations that have parameters and they increase the flexibility to cover the observed data. These parametric equations come from real-world mathematical modelling, and their parameters should be determined in terms of the observed data. Since nonlinear operators can be approximated by linear ones in many cases [5], an important category of these parametric operational equations is linear operations equation. Therefore, the major aim is to find their parameter which is known as a linear inverse problem (IP) [6].

To illustrate the key ingredients and focal point of this study, let us start by considering linear operational equations of the form

$$\mathcal{L}_{\boldsymbol{x}}^{\boldsymbol{\zeta}} u(\boldsymbol{x}) = f(\boldsymbol{x}), \tag{1}$$

which models the relationship between $u(\boldsymbol{x})$ and $f(\boldsymbol{x})$ functions. Here, $f(\boldsymbol{x})$ is considered as a black-box forcing term, $\mathcal{L}_{\boldsymbol{x}}^{\boldsymbol{\zeta}}$ is a linear operator equipped with parameter $\boldsymbol{\zeta}$ and $u(\boldsymbol{x})$ denotes the latent solution. Take, for instance, the classical problem of heat conduction in a medium with unknown conductivity properties, albeit with an unknown thermal diffusivity coefficient $\boldsymbol{\zeta}$. The main objective for solving the linear parametric problems (1) is to optimally and

stably find the parameters $\boldsymbol{\zeta}$ from the observed data using an implicit method. The linear operational equations given by (1), have enormous practical benefits which are discussed in [7, 8].

Raissi et al. [6] provide an innovative method for resolving such problems by employing and adopting GPs [9, 10] within a Bayesian framework. Likewise, the Bayesian procedure adopted here uses the GP as a flexible prior distribution over functions thereby providing analytical tractability. Once combined with the observed data, it will supply a fully probabilistic approach to approximate the functions. Moreover, GPs are among a class of methods known as kernel machines (as discussed in [5, 11, 12]) and have close similarities with regularisation approaches [13, 14].

GP as a kernel-based non-parametric method relies on an appropriate selection of kernel k. It is common to select a parametrized family of kernels. These kernels are parametrized by one or more hyperparameters $\boldsymbol{\theta}$, which then need to be estimated on the basis of the observed data. Indeed, the selection of the kernel k and the estimation of its hyperparameters have a profound impact on the performance of the GP through the covariance matrix $\boldsymbol{K}$ which is constructed based on the selected kernel.

By applying the GP as a kernel-based method which includes the hyperparameters $\boldsymbol{\theta}$, the Eq. (1) can be considered as a surrogate model where the proposed GP serves as the probabilistic approximation of the underlying problem. Given the observed data, the aim is then to learn the hyperparameters $\boldsymbol{\theta}$ and parameters $\boldsymbol{\zeta}$ using a stable method. The learned GP using the mentioned method can then be used to probabilistically solve the governing linear operational equation given in (1) and predict the behaviour of this system in the future. However, the goal pursued in many other studies addressing the similar inverse problem is that the model parameters and hyperparameters are learned directly from the observed data using various optimization techniques including Power Function method, Cross Validation method, Trial and Error method, and the Contour-Pade algorithm [15, 16, 18]. Raissi et al. [6] applied commonly done estimating the model parameters and hyperparameters directly from the data by minimizing the negative log likelihood of the probabilistic model in GP regression (See [9]), and then they used the usual optimization to fit a GP model to the parametric operational problem given in Eq. (1). The main advantage of the method proposed in [6] in comparison with the approaches mentioned above is that the optimal model parameters and hyperparameters are all learned directly from the data by maximizing the joint marginal

log-likelihood of the probabilistic model. Therefore, minimizing the negative log likelihood function estimates the (hyper)parameters that will most likely be used to produce the data required for the linear operational problem. In constructing the likelihood function, the inverse of the covariance matrix appeared in the quadratic term is used to learn from the training data, while the log-determinant of the covariance matrix penalizes the model complexity. If the covariance matrix is ill-conditioned, computing both the inverse and log-determinant of the covariance matrix with standard methods (e.g., Cholesky factorization) will probably be inaccurate. This would restrict us to use the maximum likelihood estimation (MLE) to evaluate the validity of parameters and hyperparameters estimation and their posterior mean accuracy.

This paper deals with stabilizing the likelihood computation of Gaussian process regression for data of linear operational equations and subsequent optimization of (hyper-) parameters. The main innovation is based on the novel matrix factorization of the block-structured covariance matrix which is generally unstable and ill-conditioned and needs to be inverted using Hilbert-Schmidt SVD. These block-structured partitioning of the covariance matrix is implemented stably without any major computational burdens by isolating the main source of the ill-conditioning.

Without such isolating the covariance matrix is ill-conditioned such that the likelihood computation and subsequent optimization of (hyper-) parameters is inaccurate. Therefore, the main aim of the present paper is to stabilize ill-conditioning behaviour of the likelihood computation through factorization of the covariance matrix (main source of instability) using the Hilbert-Schmidt (HS) SVD as an alternative base. Whereas the standard and basic methods (e.g. Cholesky [6], RBF-QR [19] and weighted SVD [20]) for tackling this issue is still suffering from the same presented instabilities in the likelihood function. Unlike the previous methods, to stabilize the likelihood computation every positive definite kernel is represented in terms of the (positive) eigenvalues and (normalized) eigenfunctions of an associated compact integral operator without needing to decompose the covariance matrix in the same unstable way [21]. Finally, using the properties such as orthogonality of eigenfunctions, the rapid decay of eigenvalues for highly smooth kernels, and isolation of swiftly decaying eigenvalues (as the main source of ill-conditioning in the covariance matrix) likelihood computation is implemented in stabilized and stable manner. In fact, the principal key for computing likelihood function is that it is not necessary to directly deal with the kernel in its closed form [16, 22]

because the composition of the covariance matrix can be obtained by the associate eigenvalues and eigenfunctions based on the Mercer's expansion. Furthermore, due to the nature of the linear operators such as fractional-order operators and the inherent ill-conditioning of kernel-based approximation methods, we have included the fractional derivatives of the associate eigenfunctions of the kernel instead of the kernel directly [23, 24, 26]. Therefore, the source of the ill-conditioning in the posterior prediction process is analytically removed using the HS-SVD method proposed in this paper.

At the end, the efficiency and stability of the proposed HS-SVD method have been compared with the existing methods by several illustrations of the parametric linear equations, such as ordinary and partial differential equations, integro-differential and fractional order operators.

This paper is organized as follows. In Section 2 the challenges of the GPs as a data-driven algorithm for learning general parametric linear equations are presented. In Section 3, we introduce the new stable bases using the HS-SVD method and discuss how this method can tackle the computational challenges of the likelihood function. We illustrate several numerical examples in Section 4. Finally, some conclusions are presented in Section 5.

2 Computational challenges in learning linear operators using GPs

First, we introduce the machine learning of the linear differential operators using GPs and its challenges in the computation of likelihood function and posterior mean. Without loss of generality, we assume the observed data was generated by a zero-mean GP, i.e., $\mu = 0$, although a nonzero mean can also be considered. In the following, the proposed data-driven algorithm presented by Raissi et al. [6] for learning general parametric linear equations of the form (1) corresponding to the differential operators is presented. The algorithm starts by assuming that $u(\boldsymbol{x})$ is GP with mean 0 and covariance function $k_{uu}(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta})$, i.e.,

$$u(\boldsymbol{x}) \sim \mathcal{GP}(0, k_{uu}(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta})), \tag{2}$$

where $\boldsymbol{\theta}$ denotes the hyperparameters of the kernel k_{uu}. The key observation to make is that any linear transformation of a GP such as differentiation and integration is still a GP.

Consequently,

$$\mathcal{L}_{\boldsymbol{x}}^{\boldsymbol{\zeta}} u(\boldsymbol{x}) = f(\boldsymbol{x}) \sim \mathcal{GP}(0, k_{ff}(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta}, \boldsymbol{\zeta})), \tag{3}$$

with the following relationship between the kernels k_{uu} and k_{ff},

$$k_{ff}(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta}, \boldsymbol{\zeta}) = \mathcal{L}_{\boldsymbol{x}}^{\boldsymbol{\zeta}} \mathcal{L}_{\boldsymbol{x}'}^{\boldsymbol{\zeta}} k_{uu}(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta}). \tag{4}$$

Furthermore, the covariance between $u(\boldsymbol{x})$ and $f(\boldsymbol{x}')$, and also the one between $f(\boldsymbol{x})$ and $u(\boldsymbol{x}')$ are determined by $k_{uf}(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta}, \boldsymbol{\zeta}) = \mathcal{L}_{\boldsymbol{x}'}^{\boldsymbol{\zeta}} k_{uu}(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta})$, and $k_{fu}(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta}, \boldsymbol{\zeta}) = \mathcal{L}_{\boldsymbol{x}}^{\boldsymbol{\zeta}} k_{uu}(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta})$, respectively. It goes without saying that the main purpose is to estimate the parameters $\boldsymbol{\zeta}$ of the operator $\mathcal{L}_{\boldsymbol{x}}^{\boldsymbol{\zeta}}$ and the hyperparameters of the kernels k_{ff}, k_{uf}, and k_{fu}.

There are some situations where it is reasonable to assume that the observations are noise-free, for example in computer simulations. Many scientific phenomena are investigated by complex computer models or codes. A feature of many computer experiments is that the output is deterministic i.e., rerunning the code with the same inputs gives identical observations[25]. Due to the fact that our data in the present paper are taken from computer simulations, it is assumed $\boldsymbol{y} = \begin{bmatrix} \boldsymbol{y}_u \\ \boldsymbol{y}_f \end{bmatrix}$, such that $\boldsymbol{y}_u = u(\boldsymbol{X}_u), \boldsymbol{y}_f = f(\boldsymbol{X}_f)$. Based on the aforementioned properties of the MLE, as pointed out by Raissi et al. in [6], "the hyperparameters $\boldsymbol{\theta}$ and more importantly the parameters $\boldsymbol{\zeta}$ of the linear operator $\mathcal{L}_x^{\boldsymbol{\zeta}}$ can be trained by minimizing the negative log marginal likelihood (NLML)

$$\text{NLML}(\boldsymbol{\zeta}, \boldsymbol{\theta}) = -\log p(\boldsymbol{y}|\boldsymbol{\zeta}, \boldsymbol{\theta}), \tag{5}$$

where $p(\boldsymbol{y}|\boldsymbol{\zeta}, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{K})$, however (5) can be rewritten as

$$\text{NLML}(\boldsymbol{\zeta}, \boldsymbol{\theta}) = \frac{1}{2}\log(|\boldsymbol{K}|) + \frac{1}{2}\boldsymbol{y}^{\mathsf{T}}\boldsymbol{K}^{-1}\boldsymbol{y} + \frac{N}{2}\log 2\pi, \tag{6}$$

and $\boldsymbol{K}$ is given by

$$\boldsymbol{K} = \begin{bmatrix} k_{uu}(\boldsymbol{X}_u, \boldsymbol{X}_u; \boldsymbol{\theta}) & k_{uf}(\boldsymbol{X}_u, \boldsymbol{X}_f; \boldsymbol{\theta}, \boldsymbol{\zeta}) \\ k_{fu}(\boldsymbol{X}_f, \boldsymbol{X}_u; \boldsymbol{\theta}, \boldsymbol{\zeta}) & k_{ff}(\boldsymbol{X}_f, \boldsymbol{X}_f; \boldsymbol{\theta}, \boldsymbol{\zeta}) \end{bmatrix}." \tag{7}$$

As pointed out by Raissi et al. in [6], "the marginal likelihood does not simply favor the models that fit the training data best. In fact, it induces an automatic trade-off between data-fit and model complexity. The likelihood function includes the $\boldsymbol{K}^{-1}$ term in a quadratic framework which targets to fit the training data, while the log-determinant term $\log|\boldsymbol{K}|$ penalizes the

model complexity". The most computationally intensive part of the training is associated with inverting dense covariance matrix $\boldsymbol{K}$, and computing determinant $\boldsymbol{K}$. This scales cubically with the number of observed data. This scaling is a well-known limitation of the GP, and even if $\boldsymbol{K}$ is invertible it may still be numerically ill-conditioned [27]. These challenges make the results unreliable and reduce the validity of the method. Actually, when $\boldsymbol{K}$ becomes ill-conditioned, it can lead to an ill-conditioned problem and computing with standard methods (e.g., Cholesky factorization) is probably inaccurate, leaving us unable to use the MLE to judge the validity of the method.

After training the model and parameter estimation in the previous step, we predict the values $u(\boldsymbol{x})$ and $\mathcal{L}_{\boldsymbol{x}}^{\boldsymbol{\zeta}} u(\boldsymbol{x}) = f(\boldsymbol{x})$ at a new test point $\boldsymbol{x}$ by writing the posterior distributions

$$\begin{aligned} p(u(\boldsymbol{x})|\boldsymbol{y}) &= N\left(\overline{u}(\boldsymbol{x}), v_u^2(\boldsymbol{x})\right), \\ p(f(\boldsymbol{x})|\boldsymbol{y}) &= N\left(\overline{f}(\boldsymbol{x}), v_f^2(\boldsymbol{x})\right), \end{aligned} \tag{8}$$

such that

$$\begin{aligned} \bar{u}(\boldsymbol{x}) = \boldsymbol{k}_u^{\mathsf{T}}(\boldsymbol{x})\boldsymbol{K}^{-1}\boldsymbol{y}, \quad v_u^2(\boldsymbol{x}) = k_{uu}(\boldsymbol{x},\boldsymbol{x}) - \boldsymbol{k}_u^{\mathsf{T}}(\boldsymbol{x})\boldsymbol{K}^{-1}\boldsymbol{k}_u(\boldsymbol{x}), \\ \overline{f}(\boldsymbol{x}) = \boldsymbol{k}_f^{\mathsf{T}}(\boldsymbol{x})\boldsymbol{K}^{-1}\boldsymbol{y}, \quad v_f^2(\boldsymbol{x}) = k_{ff}(\boldsymbol{x},\boldsymbol{x}) - \boldsymbol{k}_f^{\mathsf{T}}(\boldsymbol{x})\boldsymbol{K}^{-1}\boldsymbol{k}_f(\boldsymbol{x}), \end{aligned} \tag{9}$$

and

$$\begin{aligned} \boldsymbol{k}_u^{\mathsf{T}}(\boldsymbol{x}) = \boldsymbol{k}^{\mathsf{T}}(\boldsymbol{x}) = \begin{bmatrix} k_{uu}^{\mathsf{T}}(\boldsymbol{x}, \boldsymbol{X}_u) & k_{uf}^{\mathsf{T}}(\boldsymbol{x}, \boldsymbol{X}_f) \end{bmatrix}, \\ \boldsymbol{k}_f^{\mathsf{T}}(\boldsymbol{x}) = \begin{bmatrix} k_{fu}^{\mathsf{T}}(\boldsymbol{x}, \boldsymbol{X}_u) & k_{ff}^{\mathsf{T}}(\boldsymbol{x}, \boldsymbol{X}_f) \end{bmatrix}, \end{aligned} \tag{10}$$

where for notational convenience the dependence of the kernels on hyperparameters and parameters is dropped. Apart from $\bar{u}(\boldsymbol{x})$ and $\overline{f}(\boldsymbol{x})$, the posterior variances $v_u^2(\boldsymbol{x})$ and $v_f^2(\boldsymbol{x})$ can be used as good indicators of how confident one could be about the estimated parameters $\boldsymbol{\zeta}$ of the linear operator $\mathcal{L}_{\boldsymbol{x}}^{\boldsymbol{\zeta}}$ and posterior predictions made based on these parameters. However, both of these representations can lead to severe numerical instability. To overcome these challenges, we address them in Section 3. Since the GP inherits the properties of its kernel, a brief review of the kernels and their hyperparameters is required.

Since everything hinges upon our selection of the kernel k_{uu}-though this kernel is not usually known-it is common to consider a parametrized family of kernels. As mentioned before, the selection of these hyperparameters has a significant effect on the performance of the

Table 1: Some well-known kernels k_{uu} with their hyperparameters $\boldsymbol{\theta} = (\varepsilon, \beta, a, b)$.

Name	Definition
Squared Exponential (SE)	$\exp\Big(-\frac{1}{2}\sum_{i=1}^{d}\varepsilon_i(x_i - x_i')^2\Big)$
Multiquadrics (MQ)	$\sqrt{1+(\varepsilon r)^2}$
Periodic spline	$\sum_{n=1}^{\infty}\frac{2}{(4n^2\pi^2)^{\beta}}2\cos(2n\pi(x-x'))$
Generalized periodic spline	$\sum_{n=1}^{\infty}\frac{2}{(4n^2\pi^2+\varepsilon^2)^{\beta}}2\cos(2n\pi(x-x')), \quad x, x' \in [0,1], \beta \in \mathbb{N}$
Chebyshev	$\Big(2a(1-b)\frac{b(1-b^2)-2b(x^2+x'^2)+(1+3b^2)xx'}{(1-b^2)^2+4b(b(x^2+x'^2)-(1+b^2)xx')}\Big)$
	$+(1-a), \quad a \in (0,1], b \in (0,1)$

GP. Some common kernel families are included in Table 1 [9]. In Multiquadrics (MQ) kernel $r = \| \boldsymbol{x} - \boldsymbol{x'} \|$ such that $\| \, . \, \|$ is a norm on $\mathbb{R}^d$ and usually the Euclidean norm [15]. The hyperparameter b in Chebyshev kernel acts like a shape parameter, where $b \to 1$ yields more peakier kernels with increased locality (and thus reduced interactions between kernels) and $b \to 0$ yields a flatter kernel with increasingly global behavior which is less concentrated. Also, the hyperparameter a is not that significant as long as $a \in (0,1)$ as it just shifts and scales the kernel vertically. However, setting $a = 1$ eliminates the vertical shift and therefore makes it markedly more difficult to fit data with a nonzero mean. The existence of the hyperparameter ε as shape parameter in squared exponential and multiquadrics kernel (or other hyperparameters such as β in periodic spline) allows for flexibility to select a kernel supported by the data without having to explore the endless selection of all positive definite kernels. Unfortunately, this flexibility is often accompanied by the danger of severe ill-conditioning for small ε because of the increasing linear dependence of the vectors in the covariance matrix. Therefore, computing the standard form of the covariance matrix is not a good idea [28, 29, 30]. The ill-conditioning can be corrected using alternative bases and matrix factorization.

3 Overcoming computational challenges using HS-SVD bases

3.1 Stable computation in the posterior prediction

We now develop the HS-SVD method to create an alternative basis, to eliminate the ill-conditioning of the covariance matrix $\boldsymbol{K}$. In the standard approach, we work with a basis

generated by the kernel "shifts" which are known as standard bases. The new basis derived from the eigenfunction expansion produces a linear system that is devoid of the ill-conditioning of the standard form of $\boldsymbol{K}$. The HS-SVD method has two advantages in practical application: (i) it is not necessary to form the covariance matrix, and (ii) it is not necessary to know the kernel and its derivatives in closed form [16, 22]. Therefore, we apply it to the covariance matrix $\boldsymbol{K}$. We define the truncated Hilbert–Schmidt expansion with M terms (or truncated Mercer series expansion) of our kernel k_{uu} in (2) as

$$k_{uu}(\boldsymbol{x},\boldsymbol{x'};\boldsymbol{\theta}) = \sum_{n=1}^{M} \lambda_n \varphi_n(\boldsymbol{x})\varphi_n(\boldsymbol{x'}) = \begin{bmatrix}\varphi_1(\boldsymbol{x}) & \dots & \varphi_M(\boldsymbol{x})\end{bmatrix} \begin{bmatrix}\lambda_1 & & \\ & \ddots & \\ & & \lambda_M\end{bmatrix} \begin{bmatrix}\varphi_1(\boldsymbol{x'}) \\ \vdots \\ \varphi_M(\boldsymbol{x'})\end{bmatrix}, \quad (11)$$

where the truncated Mercer series provides the best M-term approximation of the kernel in the mean-square error [16].

It should be noted that, (λ_n, φ_n) are orthonormal eigenpairs of a Hilbert–Schmidt integral operator $T_{k_{uu}} : L_2(\Omega,\rho) \to L_2(\Omega,\rho)$ defined as

$$(T_{k_{uu}} f)(\boldsymbol{x}) = \int_{\Omega} k_{uu}(\boldsymbol{x},\boldsymbol{x'};\boldsymbol{\theta}) f(\boldsymbol{x'}) \rho(\boldsymbol{x'}) d\boldsymbol{x'},$$

where $\Omega \subseteq \mathbb{R}^d$, ρ is a weight function and $\| k_{uu} \|_{L_2(\Omega\times\Omega,\rho\times\rho)} < 0$ and also, M is chosen as the smallest value that satisfies $\lambda_M < \epsilon_{\text{mach}}\ \lambda_{n_u+n_f}$ and we will always assume that $M > n_u + n_f$ such that n_u and n_f are the number of training points chosen from $u(\boldsymbol{x})$ and $f(\boldsymbol{x})$ functions, respectively. Here ϵ_{mach} is machine precision (assumed to be 10^{-16}). Therefore, the quadratic form (11) can be replaced by

$$k_{uu}(\boldsymbol{x},\boldsymbol{x'};\boldsymbol{\theta}) = \boldsymbol{\phi}(\boldsymbol{x})^{\mathsf{T}} \Lambda \boldsymbol{\phi}(\boldsymbol{x'}), \quad (12)$$

where

$$\boldsymbol{\phi}(\boldsymbol{x}) = \begin{bmatrix}\varphi_1(\boldsymbol{x}) \\ \vdots \\ \varphi_M(\boldsymbol{x})\end{bmatrix}, \qquad \Lambda = \begin{bmatrix}\lambda_1 & & \\ & \ddots & \\ & & \lambda_M\end{bmatrix}.$$

The eigen-decomposition (12) provides an accurate approximation of the kernel k_{uu} without ever forming it. According to (12), the Hilbert–Schmidt eigen-decomposition of the vector $k_{uu}(\boldsymbol{x},\boldsymbol{X}_u;\boldsymbol{\theta})^{\mathsf{T}} = [k_{uu}(\boldsymbol{x},\boldsymbol{x}_1;\boldsymbol{\theta}) \dots k_{uu}(\boldsymbol{x},\boldsymbol{x}_{n_u};\boldsymbol{\theta})]$ and also the matrix $k_{uu}(\boldsymbol{X}_u,\boldsymbol{X}_u;\boldsymbol{\theta})$ in (7) are as

$$k_{uu}(\boldsymbol{x},\boldsymbol{X}_u;\boldsymbol{\theta})^{\mathsf{T}} = \boldsymbol{\phi}(\boldsymbol{x})^{\mathsf{T}} \Lambda \Phi^{\mathsf{T}} \Rightarrow k_{uu}(\boldsymbol{X}_u,\boldsymbol{X}_u;\boldsymbol{\theta}) = \Phi \Lambda \Phi^{\mathsf{T}}, \quad (13)$$

where

$$\Phi = \begin{bmatrix} \boldsymbol{\phi}(\boldsymbol{x}_1)^\mathsf{T} \\ \vdots \\ \boldsymbol{\phi}(\boldsymbol{x}_{n_u})^\mathsf{T} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\varphi}_1(\boldsymbol{x}_1) & \dots & \boldsymbol{\varphi}_M(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ \boldsymbol{\varphi}_1(\boldsymbol{x}_{n_u}) & \dots & \boldsymbol{\varphi}_M(\boldsymbol{x}_{n_u}) \end{bmatrix}.$$

However, it is not recommended to directly use the decomposition (13) either because all of the ill-conditioning associated with matrix $k_{uu}(\boldsymbol{X}_u, \boldsymbol{X}_u; \boldsymbol{\theta})$ is still present in the matrix Λ. We now use mostly standard numerical linear algebra to isolate some of the ill-conditioning and develop the HS-SVD. The key step in removing the ill-conditioning is to write the component matrices that appear in the eigen-decomposition of the matrix $k_{uu}(\boldsymbol{X}_u, \boldsymbol{X}_u; \boldsymbol{\theta})$ in (13) using blocks $\Phi = (\Phi_1 \ \Phi_2)$ and $\Lambda = \begin{bmatrix} \Lambda_1 & \\ & \Lambda_2 \end{bmatrix}$ where

$$\Phi_1 = \begin{bmatrix} \boldsymbol{\varphi}_1(\boldsymbol{x}_1) & \dots & \boldsymbol{\varphi}_{n_u}(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ \boldsymbol{\varphi}_1(\boldsymbol{x}_{n_u}) & \dots & \boldsymbol{\varphi}_{n_u}(\boldsymbol{x}_{n_u}) \end{bmatrix}, \Phi_2 = \begin{bmatrix} \boldsymbol{\varphi}_{n_u+1}(\boldsymbol{x}_1) & \dots & \boldsymbol{\varphi}_M(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ \boldsymbol{\varphi}_{n_u+1}(\boldsymbol{x}_{n_u}) & \dots & \boldsymbol{\varphi}_M(\boldsymbol{x}_{n_u}) \end{bmatrix},$$

and $\Phi_1, \Lambda_1 \in \mathbb{R}^{n_u \times n_u}$, $\Phi_2 \in \mathbb{R}^{n_u \times (M-n_u)}, \Lambda_2 \in \mathbb{R}^{(M-n_u)\times(M-n_u)}$. Now, we can write the eigen-decomposition of the vector $k_{uu}(\boldsymbol{x}, \boldsymbol{X}_u; \boldsymbol{\theta})^\mathsf{T}$ in (13) as

$$k_{uu}(\boldsymbol{x}, \boldsymbol{X}_u; \boldsymbol{\theta})^\mathsf{T} = \underbrace{\boldsymbol{\phi}(\boldsymbol{x})^\mathsf{T} \begin{bmatrix} I_{n_u} \\ \Lambda_2 {\Phi_2}^\mathsf{T} {\Phi_1}^{-\mathsf{T}} \Lambda_1^{-1} \end{bmatrix}}_{\boldsymbol{\psi}(\boldsymbol{x})^\mathsf{T}} \Lambda_1 {\Phi_1}^\mathsf{T} = \boldsymbol{\psi}(\boldsymbol{x})^\mathsf{T} \Lambda_1 {\Phi_1}^\mathsf{T},$$

and also, we can proceed to construct block matrix $k_{uu}(\boldsymbol{X}_u, \boldsymbol{X}_u; \boldsymbol{\theta})$ as following

$$k_{uu}(\boldsymbol{X}_u, \boldsymbol{X}_u; \boldsymbol{\theta}) = \Phi \Lambda \Phi^\mathsf{T} = \underbrace{\Phi \begin{bmatrix} I_{n_u} \\ \Lambda_2 \Phi_2^\mathsf{T} \Phi_1^{-\mathsf{T}} \Lambda_1^{-1} \end{bmatrix}}_{\Psi} \Lambda_1 \Phi_1^\mathsf{T} = \Psi \Lambda_1 \Phi_1^\mathsf{T}. \tag{14}$$

The rate of decay of the Hilbert-Schmidt eigenvalues determines the smoothness of the kernel k_{uu}: the faster the eigenvalues decay, the smoother the kernel (and vice versa). More specifically, if the eigenvalues decay at an algebraic rate of $\mathcal{O}^{-\beta+1+\tau}$ with $\beta \in \mathbb{N}_0$ and arbitrarily small $\tau > 0$, then the kernel will be a finite smooth kernel in C^β, and if the eigenvalues decay geometrically, then the kernel will be infinitely smooth, even analytic. Moreover, the smoothness of the kernel determines the rate of convergence of the kernel-based approximation

method. In this case the general rule of thumb is: the smoother the kernel, the faster the rate of convergence of the approximation method (for more information, see [16]).

The Mercer's theorem guarantees the uniform convergence of the series (11) provided that $T_{k_{uu}}$ is a positive operator. Also, the Mercer series provides the best approximation for the selected kernel in terms of various metrics, in particular, the mean-square error [16]. Generally, our kernel k_{uu} is positive definite and thus $T_{k_{uu}}$ is a positive operator. It would be plausible to assume the linear operators $\mathcal{L}^{\boldsymbol{\zeta}}$ are continuous (bounded) and due to the uniform convergence of the series (11), we can proceed to construct an approximation for $k_{uf}(\boldsymbol{x},\boldsymbol{x}';\boldsymbol{\theta},\boldsymbol{\zeta}) = \mathcal{L}^{\boldsymbol{\zeta}}_{\boldsymbol{x}'} k_{uu}(\boldsymbol{x},\boldsymbol{x}';\boldsymbol{\theta})$, $k_{fu}(\boldsymbol{x},\boldsymbol{x}';\boldsymbol{\theta},\boldsymbol{\zeta}) = \mathcal{L}^{\boldsymbol{\zeta}}_{\boldsymbol{x}} k_{uu}(\boldsymbol{x},\boldsymbol{x}';\boldsymbol{\theta})$ and $k_{ff}(\boldsymbol{x},\boldsymbol{x}';\boldsymbol{\theta},\boldsymbol{\zeta}) = \mathcal{L}^{\boldsymbol{\zeta}}_{\boldsymbol{x}}\mathcal{L}^{\boldsymbol{\zeta}}_{\boldsymbol{x}'} k_{uu}(\boldsymbol{x},\boldsymbol{x}';\boldsymbol{\theta})$ using Mercer series and construct block matrices of $\boldsymbol{K}$ as following

$$k_{uf}(\boldsymbol{X}_u,\boldsymbol{X}_f;\boldsymbol{\theta},\boldsymbol{\zeta}) = \underbrace{\Phi \begin{bmatrix} I_{n_f} \\ \Lambda_2 {\Phi_2}^{\mathsf{T}}_{,\mathcal{L}^{\boldsymbol{x}'}} {\Phi_1}_{,\mathcal{L}^{\boldsymbol{x}'}}{}^{-\mathsf{T}} \Lambda_1^{-1} \end{bmatrix}}_{\Psi_{\mathcal{L}^{\boldsymbol{x}'}}} \Lambda_1 {\Phi_1}^{\mathsf{T}}_{,\mathcal{L}^{\boldsymbol{x}'}} = \Psi_{\mathcal{L}^{\boldsymbol{x}'}} \Lambda_1 {\Phi_1}^{\mathsf{T}}_{,\mathcal{L}^{\boldsymbol{x}'}},$$

$$k_{ff}(\boldsymbol{X}_f,\boldsymbol{X}_f;\boldsymbol{\theta},\boldsymbol{\zeta}) = \underbrace{\Phi_{\mathcal{L}^{\boldsymbol{x}}} \begin{bmatrix} I_{n_f} \\ \Lambda_2 {\Phi_2}^{\mathsf{T}}_{,\mathcal{L}^{\boldsymbol{x}'}} {\Phi_1}^{-\mathsf{T}}_{,\mathcal{L}^{\boldsymbol{x}'}} \Lambda_1^{-1} \end{bmatrix}}_{\Psi_{\mathcal{L}^{\boldsymbol{x}}\mathcal{L}^{\boldsymbol{x}'}}} \Lambda_1 {\Phi_1}^{\mathsf{T}}_{,\mathcal{L}^{\boldsymbol{x}'}} = \Psi_{\mathcal{L}^{\boldsymbol{x}}\mathcal{L}^{\boldsymbol{x}'}} \Lambda_1 {\Phi_1}_{,\mathcal{L}^{\boldsymbol{x}'}}{}^{\mathsf{T}},$$

$$k_{fu}(\boldsymbol{X}_f,\boldsymbol{X}_u;\boldsymbol{\theta},\boldsymbol{\zeta}) = \underbrace{\Phi_{\mathcal{L}^{\boldsymbol{x}}} \begin{bmatrix} I_{n_u} \\ \Lambda_2 \Phi_2^{\mathsf{T}} \Phi_1^{-\mathsf{T}} \Lambda_1^{-1} \end{bmatrix}}_{\Psi_{\mathcal{L}^{\boldsymbol{x}}}} \Lambda_1 \Phi_1^{\mathsf{T}} = \Psi_{\mathcal{L}^{\boldsymbol{x}}} \Lambda_1 {\Phi_1}^{\mathsf{T}}, \tag{15}$$

where

$$\Phi_{\mathcal{L}^{\boldsymbol{x}}} = \begin{bmatrix} \mathcal{L}_1^{\boldsymbol{x}} \boldsymbol{\phi}(\boldsymbol{x})^{\mathsf{T}} \\ \vdots \\ \mathcal{L}_{n_f}^{\boldsymbol{x}} \boldsymbol{\phi}(\boldsymbol{x})^{\mathsf{T}} \end{bmatrix} = \begin{pmatrix} \Phi_{1,\mathcal{L}^{\boldsymbol{x}}} & \Phi_{2,\mathcal{L}^{\boldsymbol{x}}} \end{pmatrix}.$$

Indeed, to achieve more stability, the QR decomposition is used as

$$\Phi_{\mathcal{L}^{\boldsymbol{x}}} = \begin{pmatrix} \Phi_{1,\mathcal{L}^{\boldsymbol{x}}} & \Phi_{2,\mathcal{L}^{\boldsymbol{x}}} \end{pmatrix} = \mathsf{Q} \begin{pmatrix} \mathsf{R}_{1,\mathcal{L}^{\boldsymbol{x}}} & \mathsf{R}_{2,\mathcal{L}^{\boldsymbol{x}}} \end{pmatrix} \Rightarrow {\Phi_2}_{,\mathcal{L}^{\boldsymbol{x}}}{}^{\mathsf{T}} {\Phi_1}_{,\mathcal{L}^{\boldsymbol{x}}}{}^{-\mathsf{T}} = \mathsf{R}^{\mathsf{T}}_{2,\mathcal{L}^{\boldsymbol{x}}} \mathsf{Q}^{\mathsf{T}} \mathsf{Q} \mathsf{R}^{-\mathsf{T}}_{1,\mathcal{L}^{\boldsymbol{x}}} = \mathsf{R}^{\mathsf{T}}_{2,\mathcal{L}^{\boldsymbol{x}}} \mathsf{R}^{-\mathsf{T}}_{1,\mathcal{L}^{\boldsymbol{x}}}. \tag{16}$$

For example, using the relation above, it can safely be concluded that the correction matrix $[\Lambda_2 {\Phi_2}_{,\mathcal{L}^{\boldsymbol{x}}}{}^{\mathsf{T}} {\Phi_1}_{,\mathcal{L}^{\boldsymbol{x}}}{}^{-\mathsf{T}} \Lambda_1^{-1}]$ is as

$$[\Lambda_2 {\Phi_2}_{,\mathcal{L}^{\boldsymbol{x}}}{}^{\mathsf{T}} {\Phi_1}_{,\mathcal{L}^{\boldsymbol{x}}}{}^{-\mathsf{T}} \Lambda_1^{-1}] = [\Lambda_2 \mathsf{R}^{T}_{2,\mathcal{L}^{\boldsymbol{x}}} \mathsf{R}^{-\mathsf{T}}_{1,\mathcal{L}^{\boldsymbol{x}}} \Lambda_1^{-1}].$$

Now, using the relations (14–15), the covariance matrix $\boldsymbol{K}$ can be decomposed as follows

$$\boldsymbol{K} = \begin{bmatrix} \Psi\Lambda_1\Phi_1^\mathsf{T} & \Psi_{\boldsymbol{\mathcal{L}}^{\boldsymbol{x}'}}\Lambda_1{\Phi_{1,\boldsymbol{\mathcal{L}}^{\boldsymbol{x}'}}}^\mathsf{T} \\ \Psi_{\boldsymbol{\mathcal{L}}^{\boldsymbol{x}}}\Lambda_1{\Phi_1}^\mathsf{T} & \Psi_{\boldsymbol{\mathcal{L}}^{\boldsymbol{x}},\boldsymbol{\mathcal{L}}^{\boldsymbol{x}'}}\Lambda_1{\Phi_{1,\boldsymbol{\mathcal{L}}^{\boldsymbol{x}'}}}^\mathsf{T} \end{bmatrix},$$

$$= \underbrace{\begin{bmatrix} \Psi & \Psi_{\boldsymbol{\mathcal{L}}^{\boldsymbol{x}'}} \\ \Psi_{\boldsymbol{\mathcal{L}}^{\boldsymbol{x}}} & \Psi_{\boldsymbol{\mathcal{L}}^{\boldsymbol{x}},\boldsymbol{\mathcal{L}}^{\boldsymbol{x}'}} \end{bmatrix}}_{\boldsymbol{\Psi}} \underbrace{\begin{bmatrix} \Lambda_1 & 0 \\ 0 & \Lambda_1 \end{bmatrix}}_{\boldsymbol{\Lambda}_1} \underbrace{\begin{bmatrix} \Phi_1^\mathsf{T} & 0 \\ 0 & \Phi_{1,\boldsymbol{\mathcal{L}}^{\boldsymbol{x}'}}^\mathsf{T} \end{bmatrix}}_{\boldsymbol{\Phi}_1^\mathsf{T}} = \boldsymbol{\Psi}\boldsymbol{\Lambda}_1{\boldsymbol{\Phi}_1}^\mathsf{T}. \tag{17}$$

To demonstrate the usefulness of the HS-SVD, we write the posterior mean (9) as

$$\begin{aligned} \bar{u}(\boldsymbol{x}) &= \boldsymbol{k}_u^\mathsf{T}(\boldsymbol{x})\boldsymbol{K}^{-1}\boldsymbol{y} = \boldsymbol{\psi}_u^\mathsf{T}(\boldsymbol{x})\boldsymbol{b}, \\ \overline{f}(\boldsymbol{x}) &= \boldsymbol{k}_f(\boldsymbol{x})^\mathsf{T}\boldsymbol{K}^{-1}\boldsymbol{y} = \boldsymbol{\psi}_f^\mathsf{T}(\boldsymbol{x})\boldsymbol{b}, \end{aligned} \tag{18}$$

where $\boldsymbol{b} = \boldsymbol{\Psi}^{-1}\boldsymbol{y}$, $\boldsymbol{\psi}_u^\mathsf{T}(\boldsymbol{x}) = [\boldsymbol{\psi}(\boldsymbol{x})^\mathsf{T}\ \boldsymbol{\psi}_{\boldsymbol{\mathcal{L}}^{\boldsymbol{x}'}}(\boldsymbol{x})^\mathsf{T}]$ and $\boldsymbol{\psi}_f^\mathsf{T}(\boldsymbol{x}) = [\boldsymbol{\psi}_{\boldsymbol{\mathcal{L}}^{\boldsymbol{x}}}(\boldsymbol{x})^\mathsf{T}\ \boldsymbol{\psi}_{\boldsymbol{\mathcal{L}}^{\boldsymbol{x}}\boldsymbol{\mathcal{L}}^{\boldsymbol{x}'}}(\boldsymbol{x})^\mathsf{T}]$ and also,

$$\begin{aligned} \boldsymbol{\psi}_{\boldsymbol{\mathcal{L}}^{\boldsymbol{x}}}(\boldsymbol{x})^\mathsf{T} &= \boldsymbol{\phi}_{\boldsymbol{\mathcal{L}}^{\boldsymbol{x}}}(\boldsymbol{x})^\mathsf{T}\begin{bmatrix} I_{n_u} \\ \Lambda_2{\Phi_2}^\mathsf{T}{\Phi_1}^{-\mathsf{T}}\Lambda_1^{-1} \end{bmatrix}, \boldsymbol{\phi}_{\boldsymbol{\mathcal{L}}^{\boldsymbol{x}}}(\boldsymbol{x})^\mathsf{T} = \boldsymbol{\mathcal{L}}^{\boldsymbol{x}}\boldsymbol{\phi}(\boldsymbol{x})^\mathsf{T}, \\ \boldsymbol{\psi}_{\boldsymbol{\mathcal{L}}^{\boldsymbol{x}'}}(\boldsymbol{x})^\mathsf{T} &= \boldsymbol{\phi}(\boldsymbol{x})^\mathsf{T}\begin{bmatrix} I_{n_u} \\ \Lambda_2\Phi_{2,\boldsymbol{\mathcal{L}}^{\boldsymbol{x}'}}^\mathsf{T}{\Phi_{1,\boldsymbol{\mathcal{L}}^{\boldsymbol{x}'}}}^{-\mathsf{T}}\Lambda_1^{-1} \end{bmatrix}, \\ \boldsymbol{\psi}_{\boldsymbol{\mathcal{L}}^{\boldsymbol{x}}\boldsymbol{\mathcal{L}}^{\boldsymbol{x}'}}(\boldsymbol{x})^\mathsf{T} &= \boldsymbol{\phi}_{\boldsymbol{\mathcal{L}}^{\boldsymbol{x}}}(\boldsymbol{x})^\mathsf{T}\begin{bmatrix} I_{n_u} \\ \Lambda_2\Phi_{2,\boldsymbol{\mathcal{L}}^{\boldsymbol{x}'}}^\mathsf{T}{\Phi_{1,\boldsymbol{\mathcal{L}}^{\boldsymbol{x}'}}}^{-\mathsf{T}}\Lambda_1^{-1} \end{bmatrix}. \end{aligned}$$

Now, the ill-conditioning due to the dangerous $\boldsymbol{\Lambda}_1^{-1}$ term, introduced by applying $\boldsymbol{K}^{-1}$, is removed analytically through the $\boldsymbol{\Lambda}_1$ term present in $\boldsymbol{k}_u^\mathsf{T}(\boldsymbol{x}) = \boldsymbol{\psi}_u^\mathsf{T}(\boldsymbol{x})\boldsymbol{\Lambda}_1\boldsymbol{\Phi}_1^\mathsf{T}$ or $\boldsymbol{k}_f^\mathsf{T}(\boldsymbol{x}) = \boldsymbol{\psi}_f^\mathsf{T}(\boldsymbol{x})\boldsymbol{\Lambda}_1\boldsymbol{\Phi}_1^\mathsf{T}$. Also, it should be noted that computation of the posterior variances is subject to the same ill-conditioning as any expression involving $\boldsymbol{K}^{-1}$; this ill-conditioning can be similarly resolved with

$$\begin{aligned} v_u^2(\boldsymbol{x}_0) &= k_{uu}(\boldsymbol{x}_0,\boldsymbol{x}_0) - \boldsymbol{k}_u^\mathsf{T}(\boldsymbol{x}_0)\boldsymbol{K}^{-1}\boldsymbol{k}_u(\boldsymbol{x}_0) = k(\boldsymbol{x}_0,\boldsymbol{x}_0) - \boldsymbol{\psi}_u^\mathsf{T}(\boldsymbol{x}_0)\boldsymbol{\Psi}^{-1}\boldsymbol{k}_u(\boldsymbol{x}_0), \\ v_f^2(\boldsymbol{x}_0) &= k_{ff}(\boldsymbol{x}_0,\boldsymbol{x}_0) - \boldsymbol{k}_f^\mathsf{T}(\boldsymbol{x}_0)\boldsymbol{K}^{-1}\boldsymbol{k}_f(\boldsymbol{x}_0) = \mathcal{L}_{\boldsymbol{x}_0}^{\boldsymbol{\zeta}}\mathcal{L}_{\boldsymbol{x}_0}^{\boldsymbol{\zeta}}k(\boldsymbol{x},\boldsymbol{x}) - \boldsymbol{\psi}_f^\mathsf{T}(\boldsymbol{x}_0)\boldsymbol{\Psi}^{-1}\boldsymbol{k}_f(\boldsymbol{x}_0). \end{aligned} \tag{19}$$

As a result, the main portion of the ill-conditioning can be resolved in the posterior variances $v_u^2(\boldsymbol{x})$ and $v_f^2(\boldsymbol{x})$ using (19).

3.2 Stable likelihood function computation

For the likelihood function defined in (6), when $\boldsymbol{K}$ becomes ill-conditioned (e.g., for small ε, since the use of "flatter" kernels will lead to more and more similar entries in the system matrix $\boldsymbol{K}$, and therefore to potential numerical problems due to the ill-conditioning of $\boldsymbol{K}$ [16]), computing $\boldsymbol{y}^{\mathsf{T}}\boldsymbol{K}^{-1}\boldsymbol{y}$ and $\det \boldsymbol{K}$ by standard methods (such as Cholesky factorization) may be inaccurate, leaving us unable to use the MLE to judge the validity of the small ε for our estimation, despite the fact that (18) would allow us to make posterior predictions accurately. Using the HS-SVD decomposition of the covariance matrix $\mathbf{K}$ in (17), we can follow a similar strategy as in Subsection 3.1 to the stable computation of log likelihood function (6). Computing $\log\det \boldsymbol{K}$ is relatively straightforward using $\boldsymbol{K} = \boldsymbol{\Psi}\boldsymbol{\Lambda_1}\boldsymbol{\Phi}_1^{\mathsf{T}}$, as

$$\log|\boldsymbol{K}| = \log|\boldsymbol{\Psi}| + \log|\boldsymbol{\Lambda}_1| + \log|\boldsymbol{\Phi}_1^{\mathsf{T}}|. \tag{20}$$

It should be noted that $\boldsymbol{\Lambda_1}$ is diagonal, and therefore the very small eigenvalues can be handled by taking their logarithms. A similar strategy will allow us to compute $\boldsymbol{y}^{\mathsf{T}}\boldsymbol{K}^{-1}\boldsymbol{y}$. Applying $\boldsymbol{\Psi b} = \boldsymbol{y}$ and the Hilbert–Schmidt SVD (17) to $\boldsymbol{y}^{\mathsf{T}}\boldsymbol{K}^{-1}\boldsymbol{y}$ gives

$$\boldsymbol{y}^{\mathsf{T}}\boldsymbol{K}^{-1}\boldsymbol{y} = (\boldsymbol{\Psi b})^{\mathsf{T}}(\boldsymbol{\Psi}\boldsymbol{\Lambda_1}\boldsymbol{\Phi}_1^{\mathsf{T}})^{-1}\boldsymbol{\Psi b} = \boldsymbol{b}^{\mathsf{T}}\boldsymbol{\Psi}^{\mathsf{T}}\boldsymbol{\Phi}_1^{-\mathsf{T}}{\boldsymbol{\Lambda}_1}^{-1}\boldsymbol{b}. \tag{21}$$

Now, we are in a situation to find the Hilbert–Schmidt decomposition of the negative log marginal likelihood in (6) using (20) and (21) as follows

$$\begin{aligned}\mathrm{NLML}_{\mathrm{HS}}(\boldsymbol{\zeta},\boldsymbol{\theta}) = \frac{1}{2}\boldsymbol{b}^{\mathsf{T}}\boldsymbol{\Psi}^{\mathsf{T}}\boldsymbol{\Phi}_{\mathbf{1}}^{-\mathsf{T}}\boldsymbol{\Lambda}_1^{-1}\boldsymbol{b} \\ + \frac{1}{2}(\log|\boldsymbol{\Psi}| + \log|\boldsymbol{\Lambda}_1| + \log|\boldsymbol{\Phi}_1^{\mathsf{T}}|) + \frac{N}{2}\log 2\pi.\end{aligned} \tag{22}$$

The (hyper)parameters $\boldsymbol{\theta}$ and $\boldsymbol{\zeta}$ can be trained by employing a Quasi-Newton optimizer L-BFGS to minimize the negative log marginal likelihood [6]. To set the hyperparameters by minimizing the negative log marginal likelihood, we seek the partial derivatives of the marginal likelihood with respect to the (hyper)parameters. The partial derivatives of the marginal likelihood with respect to the (hyper)parameters can be calculated by the relation

$$\frac{\partial}{\partial\theta_j}\mathrm{NLML}(\boldsymbol{\zeta},\boldsymbol{\theta}) = -\frac{\partial}{\partial\theta_j}\log p(\boldsymbol{y}|\boldsymbol{\zeta},\boldsymbol{\theta}) = -\frac{1}{2}tr\big((\boldsymbol{\alpha}\boldsymbol{\alpha}^{\mathsf{T}} - \boldsymbol{K}^{-1})\frac{\partial\boldsymbol{K}}{\partial\theta_j}\big),$$

where $\boldsymbol{\alpha} = \boldsymbol{K}^{-1}\boldsymbol{y}$ [9]. Equivalent relation for the partial derivatives of Hilbert–Schmidt decomposition of the negative log marginal likelihood in (22) is as

$$\frac{\partial}{\partial\theta_j}\mathrm{NLML}_{\mathrm{HS}}(\boldsymbol{\zeta},\boldsymbol{\theta}) = -\frac{1}{2}tr\Big(\boldsymbol{\beta}\big(\boldsymbol{b}\boldsymbol{b}^{\mathsf{T}}\boldsymbol{\beta}^{\mathsf{T}} - \boldsymbol{\Psi}^{-1}\big)\frac{\partial(\boldsymbol{\Psi}\boldsymbol{\Lambda_1}{\boldsymbol{\Phi}_1}^{\mathsf{T}})}{\partial\theta_j}\Big),$$

where $\boldsymbol{\beta} = (\boldsymbol{\Lambda_1}\boldsymbol{\Phi}_1{}^{\mathsf{T}})^{-1}$.

In the following, we explain how the HS-SVD alternative bases can reduce the instabilities in (hyper)parameter estimation and posterior prediction using the GPs. The first step in creating a stable basis is to study the structure of the system in (13) and asking the question Why is this system ill-conditioned?

It would seem that because the eigenfunctions φ_n are orthogonal the matrix Φ in relation (13) should be relatively well behaved. Therefore, the ill-conditioning appears to instead originate in the diagonal matrix Λ which contains block matrices Λ_1 and Λ_2, whose values are the eigenvalues of $\boldsymbol{K}$ and its 2-norm condition number is as

$$\text{cond}(\Lambda) = \frac{\lambda_1}{\lambda_M}. \tag{23}$$

This condition number is not directly relevant, since the entire Λ matrix is never inverted, but it serves to give an idea of the delicate nature of the $\boldsymbol{K}$ matrix.

In fact, the rate of decay of the Hilbert-Schmidt eigenvalues determines the smoothness of the kernel k_{uu}. Therefore, in the course of using more smooth kernels, the problem of instability will be raised more seriously according to relation the (23). This connection between the ill-conditioning of the system and the smoothness of the kernel has been studied in [16]. It would appear then that the presence of the Λ matrix is the main source of ill-conditioning. The key step in removing the ill-conditioning is to write the component matrices that appear in the eigen-decomposition of $\boldsymbol{K}$ as (17). Therefore, instead of solving the standard systems (6) and (9) which have the potential of yielding inaccurate and unreliable results, we now solve the transformed systems (18), (19) and (22), which are more numerically stable. In addition, we always make sure to simultaneously apply the matrices Λ_1 and Λ_2 required in the computation of the corrector matrices $[\Lambda_2\Phi_2^{\mathsf{T}}\Phi_1^{-\mathsf{T}}\Lambda_1^{-1}]$ and $[\Lambda_2\Phi_{2,\boldsymbol{\mathcal{L}}^{\boldsymbol{x}}}^{\mathsf{T}}\Phi_{1,\boldsymbol{\mathcal{L}}^{\boldsymbol{x}}}^{-\mathsf{T}}\Lambda_1^{-1}]$. For example, in MATLAB software, $[\Lambda_2\Phi_2^{\mathsf{T}}\Phi_1^{-\mathsf{T}}\Lambda_1^{-1}]$ is computed as follows

$$\mathsf{bsxfun}(@\mathsf{rdivide}, \Lambda_2, \Lambda_1). * (\Phi_2/\Phi_1').$$

This minimizes the chances of producing an overflow or underflow error caused, respectively, by dividing the small eigenvalues at the end of Λ_1 or multiplying by the even tinier ones in Λ_2. On the other hand, using the QR decomposition (16) is sometimes preferable to directly invert $\Phi_{1,\boldsymbol{\mathcal{L}}^{\boldsymbol{x}\prime}}^{\mathsf{T}}$ because it may be a more stable computation than LU factorization, which may be preferable depending on the scale of the various eigenfunctions [31]. Therefore, the use

of alternative HS-SVD bases (with or without the optional QR step) allows us to isolate the ill-conditioning primarily in the Λ_1 factor and resolve the corrector matrices safely and recommended for the stable computation of optimal parameters and operators posterior predictions, especially in the kernel flat-limit.

4 Numerical experiments

The main purpose of the numerical results is to point out the efficiency, validity and stability of the method presented in this paper to estimate (hyper)parameters and operators posterior prediction in a numerically stable way as presented here. In the following, we present a series of numerical experiments that demonstrate the effectiveness of our approach. We have implemented the process of (hyper)parameters estimation by employing an L-BFGS optimization method and posterior prediction of various differential and integral linear operators in new points as well. Using various figures, the numerical stability of calculation of MLE in the HS-SVD method in comparison with the direct method is shown. Also, to show the accuracy and numerical stability of posterior mean of the HS-SVD method in comparison with the standard method, the maximum absolute error (absolute error) graphs of the posterior means and the condition number of the covariance matrices for both methods are presented. The absolute error between the exact function $u(\boldsymbol{x})$ and the predicted mean $\bar{u}(\boldsymbol{x})$ is described as below:

$$\max_{1\leq i\leq N} |u(\boldsymbol{x}_i) - \bar{u}(\boldsymbol{x}_i)|.$$

It should also be noted that in the following examples, we have ignored the constant term $\frac{N}{2}\log 2\pi$ in the calculation of the likelihood function (6). In fact, in calculating of NLML, we have used the relations $\text{Dmle} = \text{NLML}(\boldsymbol{\zeta}, \boldsymbol{\theta}) - \frac{N}{2}\log 2\pi$ for standard (direct) computation (labeled in the figures with Direct Likelihood) and $\text{HSmle} = \text{NLML}_{\text{HS}}(\boldsymbol{\zeta}, \boldsymbol{\theta}) - \frac{N}{2}\log 2\pi$ for HS-SVD computation (labeled in the figures with HS-SVD Likelihood) as the likelihood criterion. Note that in all figures and tables to compare "Direct Likelihood" and "HS-SVD likelihood" the NLML (Negative Log Marginal Likelihood) quantity is used such that a higher direct likelihood means a lower NLML.

Example 1 Consider the one dimensional fractional equation

$$\mathcal{L}_x^\alpha u(x) = {}_{-\infty}^{RL}D_x^\alpha u(x) - u(x) = f(x),$$

where $\alpha \in \mathbb{R}$ and ${}^{RL}_{-\infty}D^{\alpha}_{x}$ is defined in the Riemann-Liouville sense [32]. As pointed out by Raissi et al. in [6], "Fractional operators often arise in modeling anomalous diffusion processes. Their non-local behavior poses serious computational challenges as it involves costly convolution operations for resolving the underlying non-Markovian dynamics". It should be noted that ${}^{RL}_{-\infty}D^{\alpha}_{x}k_{uu}$, ${}^{RL}_{-\infty}D^{\alpha}_{y}[{}^{RL}_{-\infty}D^{\alpha}_{x}k_{uu}]$ and ${}^{RL}_{-\infty}D^{\alpha}_{x}\varphi(x)$ were obtained using generalized Gauss-Laguerre quadrature method, involving a weight function of the form $x^{\alpha_{gGL}}e^{-x}$ for $\alpha_{gGL} > -1$ as

$$\int_0^{\infty} f(x)dx = \int_0^{\infty} x^{\alpha_{gGL}}e^{-x}[e^{x}x^{-\alpha_{gGL}}f(x)]dx \simeq \sum_{i}^{n} w_i e^{x_i}x_i^{-\alpha_{gGL}}f(x_i). \tag{24}$$

We use the Golub-Welsch algorithm to find the nodes, but we compute the weights by evaluating the generalized Gauss-Laguerre polynomial at these nodes for higher relative accuracy. In practice, it is essential for α_{gGL} to match the fractional part of power of the monomial in the integrand f, as the remainder yields a smooth function (for more information, see [23]). However, the presented stable machine learning method using the GP overcomes these computational challenges, and we can seamlessly handle all such linear cases without any modifications. In this example, we have used a generalized periodic spline kernel on a set of one-dimensional data in the interval $[0, 1]$. This kernel has eigenvalues and eigenfunctions

$$\lambda_n = \begin{cases} ((2j\pi)^2 + \varepsilon^2)^{(-\beta)} & n = 2j-1, \\ ((2j\pi)^2 + \varepsilon^2)^{(-\beta)} & n = 2j, \end{cases}$$

$$\varphi_n(x) = \begin{cases} \sqrt{2}\sin(2j\pi x) & n = 2j-1, \\ \sqrt{2}\cos(2j\pi x) & n = 2j, \end{cases}$$

for $j = 1, 2, \ldots$. It should also be emphasized that we have used the roots of the squared exponential kernel eigenfunctions as training points labelled as "Roots". We create data values $\{x_u, y_u\}, \{x_f, y_f\}$ by sampling the test function

$$u(x) = \frac{1}{2}e^{-2\pi ix}\left(\frac{(2\pi + i)e^{4i\pi x}}{-1 + (2i\pi)^{\sqrt{2}}} + \frac{2\pi - i}{-1 + (-2i\pi)^{\sqrt{2}}}\right)$$

and $f(x) = 2\pi\cos(2\pi x) - \sin(2\pi x)$, for $\alpha = \sqrt{2}$ which the training points chosen in the interval $[0, 1]$. It should be noted that to illustrate the instability and computational challenges of the direct approach and the efficiency of the HS-SVD approach, various number of Roots data

points are chosen in the interval $[0, 1]$. In Table 2, the absolute error, condition number, the optimal values of α and likelihood criterion are presented for different values of (n_u, n_f) of Roots data points. It should be made clear here that in Table 2, the optimal values of (hyper)parameters are obtained and then the absolute error for the posterior mean of the fractional operators in new points using the optimal (hyper)parameters are reported. The likelihood criterion is evaluated for α spaced uniformly in $[.1, 5]$, ε spaced logarithmically in $[.1, 10]$ and $\beta = 3$ using both the direct approach (labeled Direct Likelihood) based on Cholesky decomposition and HS-SVD method (labeled HS-SVD Likelihood) in Fig. 1 and and specifically 2 which provides more accurate and more stable results. A similar pattern is observed for other values of β. In addition, as the number of data points increases, the instability of the direct method increases. This issue is not brought to the forefront here because it seems to be redundant. In fact, we have demonstrated the training process to learn the optimal α and ε parameters simultaneously using both direct and HS-SVD techniques, in Figs. 1 and 2. we have also made posterior predictions at `Neval`=100 evenly spaced points in the domain to calculate the error value logarithmically for different values of α and ε with $\beta = 3$ to show the validity of the HS-SVD method. Due to the error figures, we have found out that the HS-SVD method correctly determines a region for an "optimal" (hyper) parameter estimate α and ε, while we have noticed instability in the direct approach. It is clear that by increasing the number of training points, the direct approach in parameters estimation and operators posterior prediction loses accuracy and suffers a complete breakdown because $\boldsymbol{K}^{-1}$ is too ill-conditioned. Absolute error between the true fractional order α and the estimated one (top) and also between the exact function $u(x)$ and the predicted mean $\bar{u}(x)$ (middle) in the logarithmic scale as a function of the total number of training points for $u(x)$ and $f(x)$ denoted by n_u and n_f and condition number of covariance matrix $\boldsymbol{K}$ and matrix Ψ (bottom) using both approaches demonstrated in Fig. 3. As Figs. 1,2 and 3 show, in contradiction with the notion that increasing the number of data points leads to an increase in the accuracy of calculations, we see that this does not happen in direct method because of ill-conditioning covariance matrix $\boldsymbol{K}^{-1}$. While in HS-SVD method, by increasing the number of data points, in addition to maintaining stability, the accuracy of the results also increases. It is apparent that the stably computed likelihood parametrization criterion, HSmle, identifies a region for an "optimal" (hyper)parameters estimate that matches the region of smallest error. According to the Figs. 1, 2 and 3, we can see the HS-SVD algorithm learns the parameter α and ε to have

"optimal" values, while the direct maximum likelihood estimator does not precisely locate it. That's because the MLE direct computation loses accuracy and suffers a complete breakdown because $\boldsymbol{K}^{-1}$ is too ill-conditioned. Furthermore, that indicates the satisfactory performance of the method presented in this paper for different trainings.

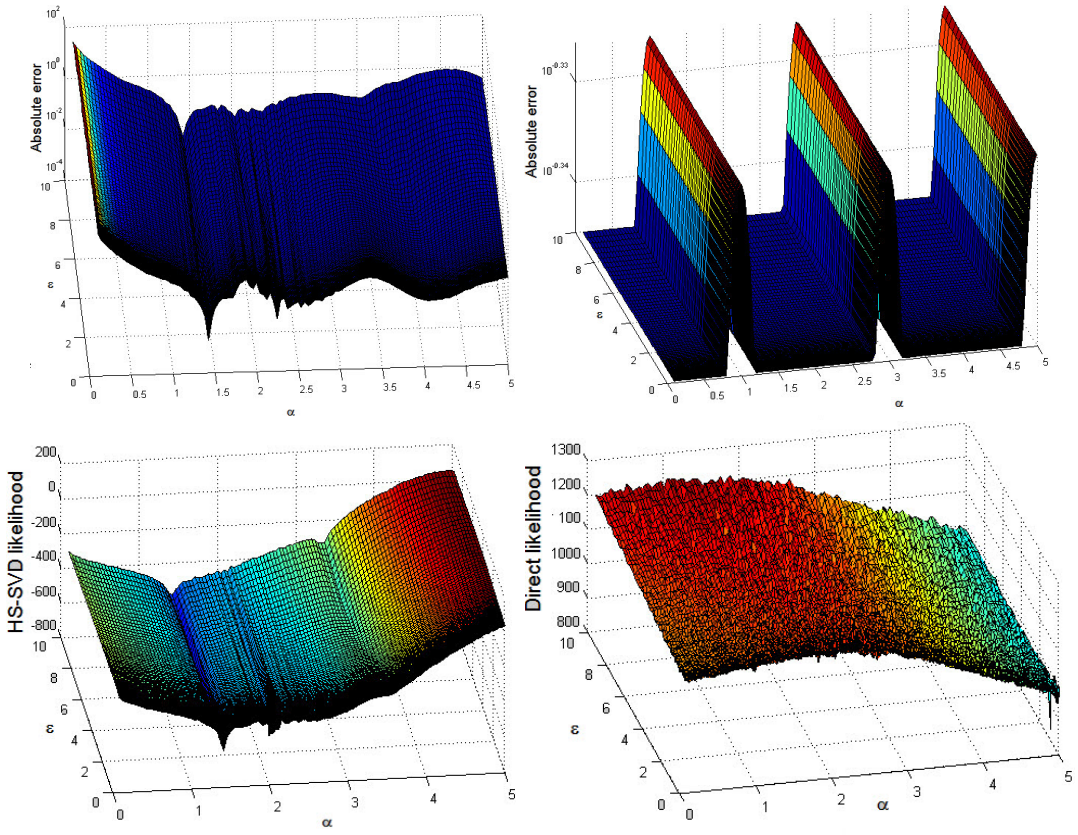

Figure 1: Comparison of the Negative Log Marginal likelihood (NLML) criterion and the error of the posterior mean computed by the direct method (right) and HS-SVD method (left) for $\beta = 3$ and different values ε and α. The top row shows the error of the posterior mean based on $n_u = n_f = 10$ Roots data points using generalized periodic spline kernel displayed. The bottom row displays the corresponding likelihood estimates for Example 1.

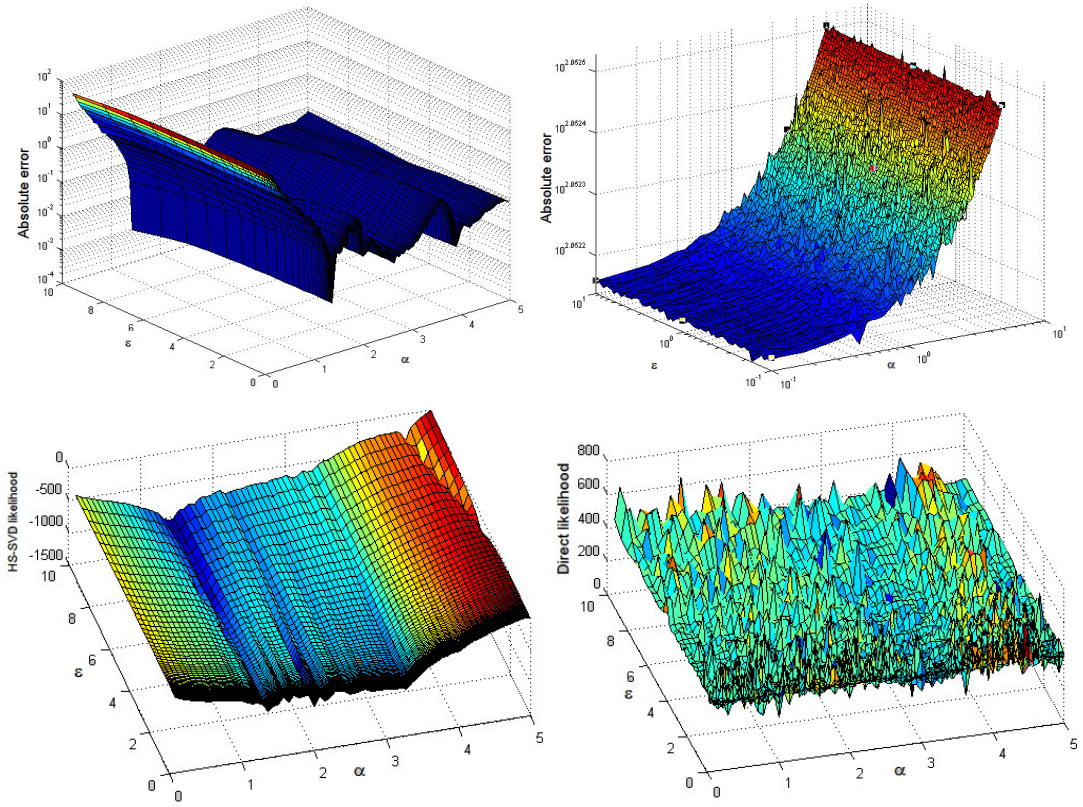

Figure 2: Comparison of the Negative Log Marginal likelihood (NLML) criterion and the error of the posterior mean computed by the direct method (right) and HS-SVD method (left) for $\beta = 3, n_u = n_f = 20$ and different values ε and α for Example 1.

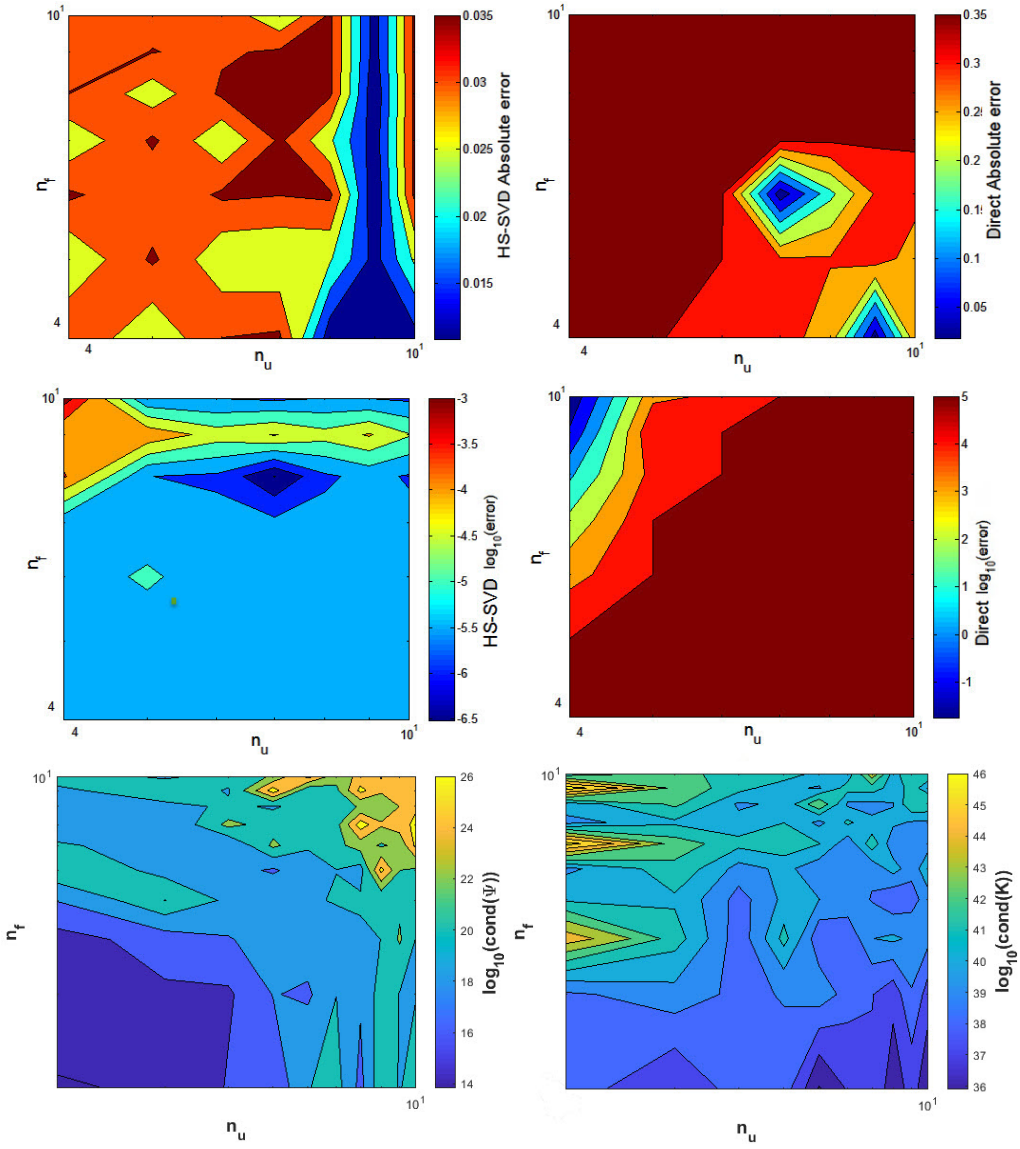

Figure 3: Absolute error between the true fractional order α and the estimated one (top), the exact function $u(x)$ and the predicted mean $\bar{u}(x)$ in the logarithmic scale as a function of the total number of training points for $u(x)$ and $f(x)$ (middle), denoted by n_u and n_f is shown with both methods. Condition number of covariance matrix $\boldsymbol{K}$ defined in (17) and matrix Ψ (bottom) with $\beta = 3$ in the logarithmic scale is demonstrated using Roots data points for Example 1.

Table 2: Parameter estimation α and absolute error (error) of fractional operators posterior prediction for optimal (hyper)parameters obtained with direct and HS-SVD methods using different number of Roots data points for Example 1.

	HS-SVD method				Direct method			
(n_u, n_f)	α	HSmle	cond(Ψ)	error	α	Dmle	cond($\boldsymbol{K}$)	error
$(5,5)$	4.8651	-166.3783	5.5679e+16	0.4518	5.000	568.6083	2.7907e+41	4.7185e-01
$(10,10)$	2.0353	-577.3515	2.3678e+18	0.0938	5.0000	939.1081	1.4683e+44	4.7185e-01
$(20,20)$	1.4043	-1.0783e+03	1.4259e+22	0.0057	3.8715	195.8455	7.2055e+45	1.1283e+02

Example 2 Consider the one dimensional fractional equation

$$\mathcal{L}_x^{\alpha} u(x) = {}_0^C D_x^{.75} u(x) + \alpha u(x) = f(x),$$

where $\alpha \in \mathbb{R}$ and ${}_0^C D_x^{\alpha}$ are defined in the Caputo sense [32]. In this example, we have used periodic spline kernel on a set of one-dimensional data. Also, the fractional derivatives of the kernel and ${}_0^C D_x^{.75} k_{uu}$, ${}_0^C D_y^{.75}[{}_0^C D_x^{.75} k_{uu}]$ and ${}_0^C D_x^{.75}\varphi(x)$ are approximated based on power series expansion with Maple software by the command "fracdiff" as "fracdiff($k_{uu}(x,y)$, α, method = series, method-options = [about= a]". The optional parameters for this method to be specified in method-options, are about= a and order = o. The value of a specifies the point on how to expand the series and o specifies the accuracy or order of the series. For convenience, the order o = 20 and the starting point of the interval a = 0 are considered.

The periodic spline kernel has eigenvalues and eigenfunctions

$$\lambda_n = \begin{cases} (2j\pi)^{(-2\beta)} & n = 2j-1, \\ (2j\pi)^{(-2\beta)} & n = 2j, \end{cases}$$

$$\varphi_n(x) = \begin{cases} \sqrt{2}\sin(2j\pi x) & n = 2j-1, \\ \sqrt{2}\cos(2j\pi x) & n = 2j, \end{cases}$$

for $\beta \in \mathbb{N}$, $j = 1, 2, \ldots$. We simulate data values $\{x_u, y_u\}, \{x_f, y_f\}$ by sampling the test function $u(x) = \sin(2\pi x)$ and

$$f(x) = 6.9320\sqrt[4]{x}\ {}_1F_2(1.0; 0.625, 1.125; -9.86960\,x^2) + 2\sin(2\pi x)$$

for $\alpha = 2$ and $n_u = n_f = 10$ Chebyshev data points are chosen in the interval $[0, 1]$ where ${}_1F_2$ is the generalized hypergeometric function. Likelihood criterion is obtained for the values of $\beta \in \{1, 2 \ldots, 10\}$ and α spaced logarithmically in $[10^{-4}, 10]$ using both the direct method (labelled Direct Likelihood in Fig. 4) based on Cholesky decomposition, and the more elaborate for relation (22) which provide more stable results (labelled HS-SVD Likelihood). The data is then used to make posterior predictions at `Neval` =100 evenly spaced points in the domain, and the absolute error compared to $u(x)$ displayed in Fig. 5. As Fig. 4 shows, the posterior mean (top row), as well as the likelihood criterion (bottom row), can be stably and reliably computed with the help of the HS-SVD (left column)-as compared to the direct approach, displayed in the right column, and computed using the standard methods such as the Cholesky decomposition. According to Fig. 4, we can observe that the HS-SVD algorithm correctly learns the parameter α and β to have "optimal" values, while the direct MLE is near the optimal error and it does not exactly locate it. Also, According to Fig. 5, we realize that the HS-SVD method determines a more precise region for "optimal" (hyper) parameters α and β. In addition, in Table 3 the absolute error and likelihood criterion for different values of (n_u, n_f) of Chebyshev data points are presented. In Table 3, the optimal value of the parameter α is presented, and the absolute error for the posterior mean of the fractional operators in new data points using the optimal (hyper) parameters are reported.

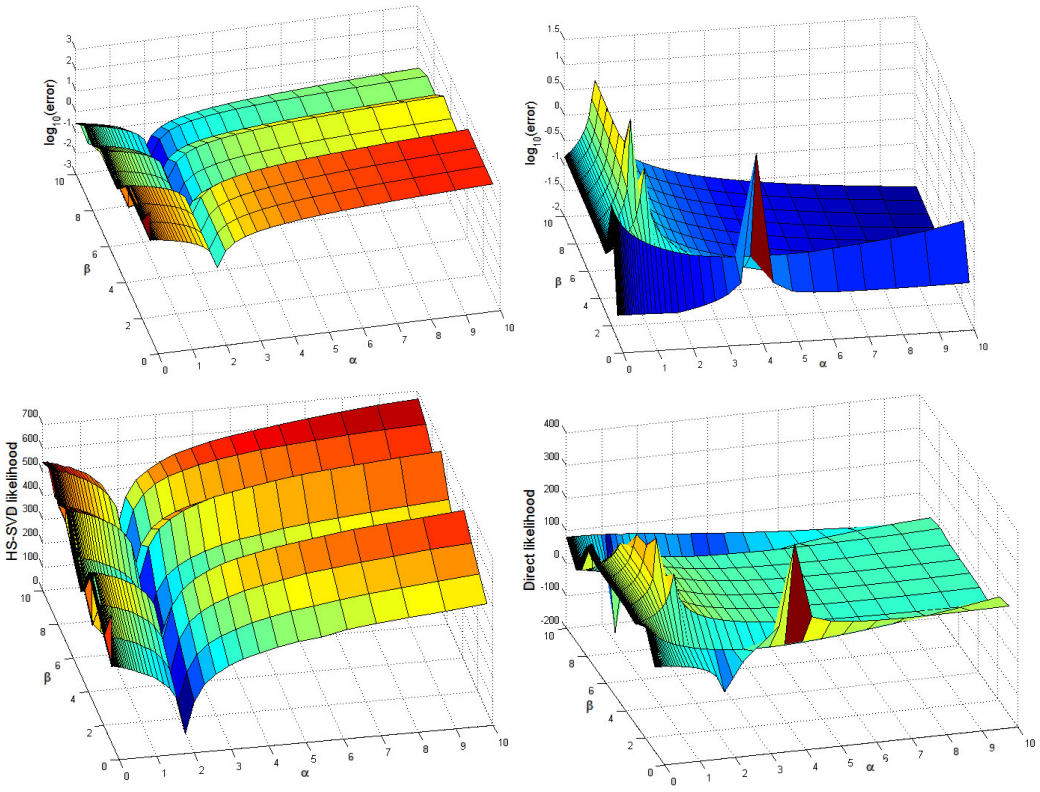

Figure 4: Comparison of Negative Log Marginal likelihood (NLML) criterion computed with both methods. The top row shows the error of the posterior mean based on Chebyshev data points using periodic spline kernel. The bottom row displays the corresponding likelihood estimates.

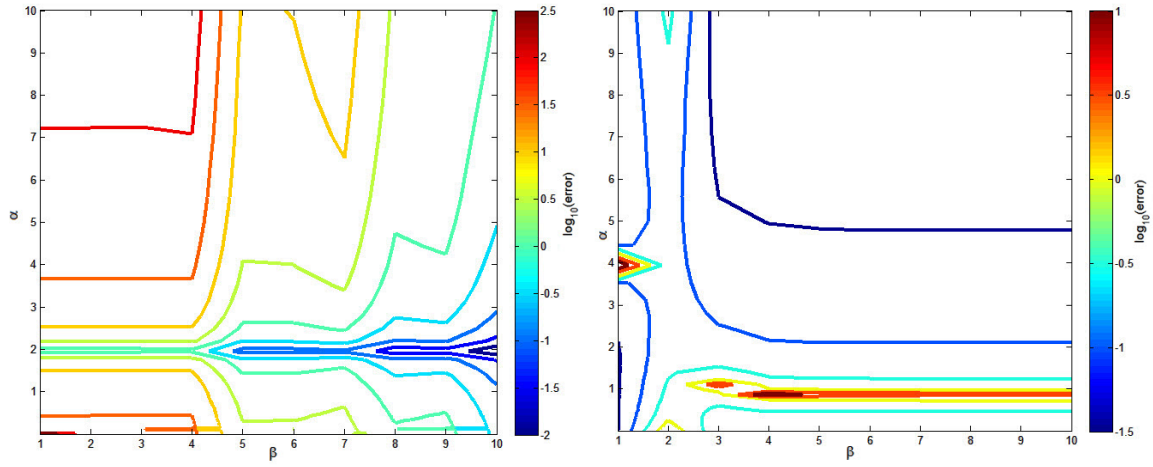

Figure 5: Absolute error for Example 2 using periodic spline kernel computed via the HS-SVD approach (left) and direct approach (right).

Table 3: Parameters estimation α and absolute error of fractional operators posterior prediction for optimal (hyper) parameters obtained with direct and HS-SVD methods by Chebyshev data points for Example 2.

	HS-SVD method				Direct method			
(n_u, n_f)	α	HSmle	cond(Ψ)	error	α	Dmle	cond($\boldsymbol{K}$)	error
$(5,5)$	1.9307	-46.6439	6.1427e+03	0.0152	0.7543	-26.5178	1.8353e+15	11.4753
$(10,10)$	1.9307	-83.5802	2.6829e+05	0.0108	0.0910	-71.1816	7.8813e+16	0.5588
$(20,20)$	1.9698	-125.819	1.9509e+07	0.0015	1.2068	-63.9671	4.0022e+17	0.3404

Example 3 Consider the following differential equation,

$$\mathcal{L}_x^{\alpha} u(x) = \frac{d^2}{dx^2}u(x) + \frac{\alpha x}{x^2+1}\frac{d}{dx}u(x) + u(x) = f(x).$$

Note that the functions $u(x) = -2\cos(4\pi x) + \sin(4\pi x)$ and

$$f(x) = 32\pi^2 cos(4\pi x) - 16\pi^2 \sin(4\pi x) + \frac{\alpha x(8\pi \sin(4\pi x))}{x^2+1}$$
$$\frac{\alpha x(4\pi \cos(4\pi x))}{x^2+1} - 2\cos(4\pi x) + \sin(4\pi x)$$

satisfy the equation. We create data values $\{x_u, y_u\}, \{x_f, y_f\}$ by sampling the test function $u(x)$ and $f(x)$ with $n_u = n_f = 40$ Chebyshev data points are chosen in the interval $[-1, 1]$ and $\alpha = 6$. In this example we have used Chebyshev kernel with eigenvalues and eigenfunctions

$$\lambda_n = \begin{cases} 1 - a & n = 0, \\ \frac{a(1-b)b^n}{b} & n = 1, 2, \ldots \end{cases}$$

$$\varphi_n(x) = \sqrt{2 - \delta_{n0}} T_n(x),$$

where T_n are Chebyshev polynomials of degree n. Likelihood criterion is obtained for values of $a = .5$ and b spaced uniformly in $[10^{-4}, .9]$ using both methods in Fig. 6. The data is then used to make predictions at `Neval` =100 evenly spaced points in the domain, and the absolute errors compared to $u(x)$ are displayed in Figs. 6 and 7. As Fig. 6 shows, the posterior mean and the likelihood criterion can stably and reliably be computed with the help of the HS-SVD -as compared to the direct approach, computed with the standard methods. It is apparent that the stably computed likelihood parametrization criterion identifies a region for an "optimal" (hyper) parameters estimate α and b that matches the region of the smallest error in Figs. 6 and 7. Figure 8 indicates Maximum error between exact function and predicted mean $\bar{u}(x)$, exact function and predicted mean $\bar{f}(x)$ in the logarithmic scale as a function of the total number of training points $u(x)$ and $f(x)$, denoted by n_u and n_f, with both methods and also, the condition number of covariance matrix $\boldsymbol{K}$ and matrix Ψ based on Chebyshev data points. It also reveals that by increasing the number of data points the accuracy of calculations increases. Needless to say, this does not hold true in the direct method. As far as the HS-SVD method is concerned, an increase in the number of data points leads to an increase in the accuracy of the results and also the maintenance of the stability. Of course, whenever $\boldsymbol{K}$ is not severely ill-conditioned (usually this is true for kernels with a low level of smoothness such as Matern kernels or compactly supported Wendland kernels) it is easier and more convenient to work with its Cholesky factorization $\boldsymbol{K} = \mathsf{L}\mathsf{L}^{\mathsf{T}}$ as a fundamental tool in matrix computations.

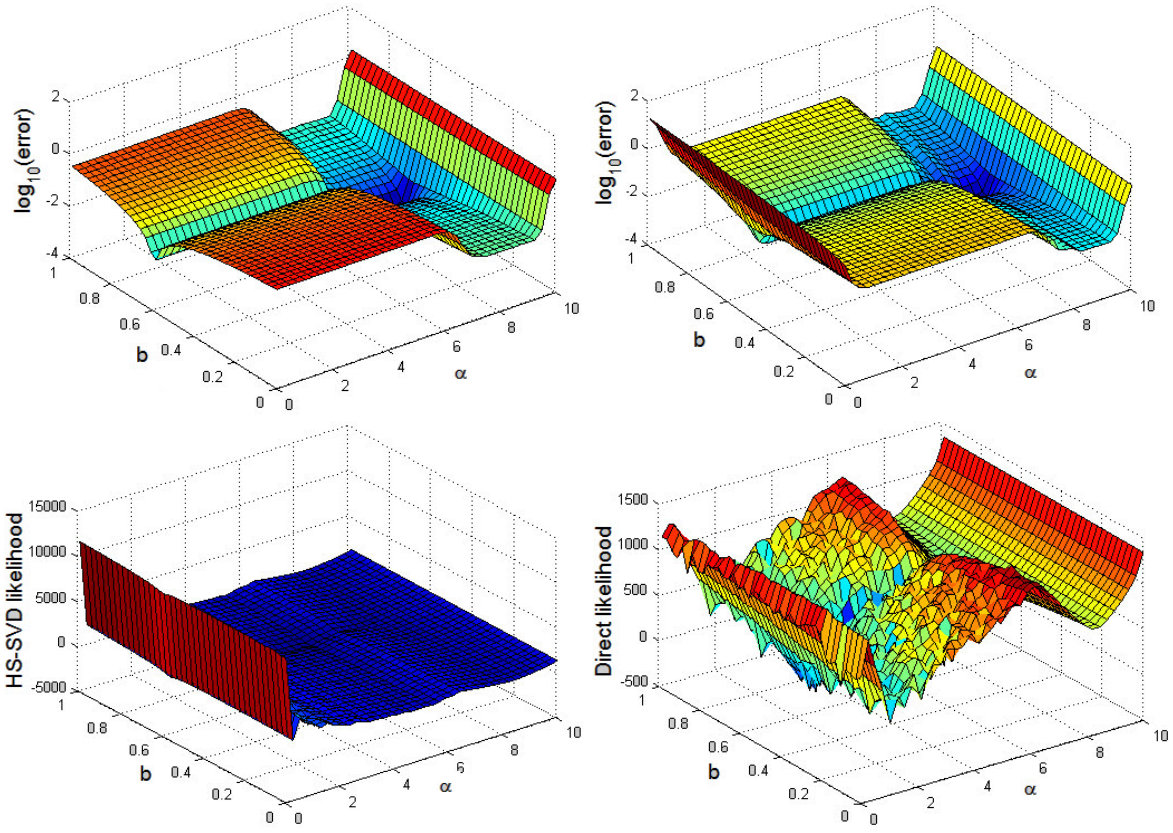

Figure 6: Comparison of Negative Log Marginal likelihood (NLML) criterion computed with both methods. The top row shows the error of the posterior mean based on Chebyshev data points using Chebyshev kernel. The bottom row displays the corresponding likelihood estimates.

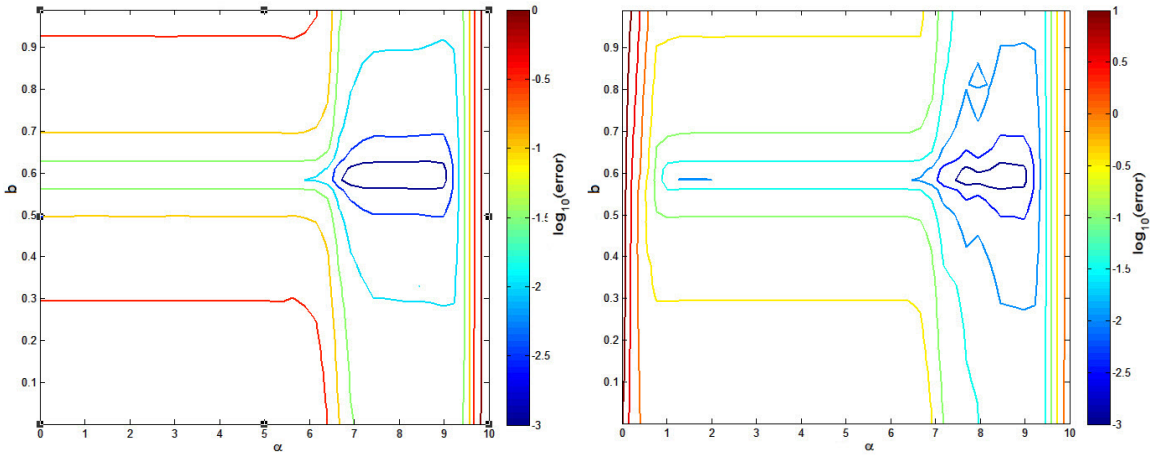

Figure 7: Absolute error for Example 3 using Chebyshev kernel computed via the HS-SVD approach (left) and direct approach (right).

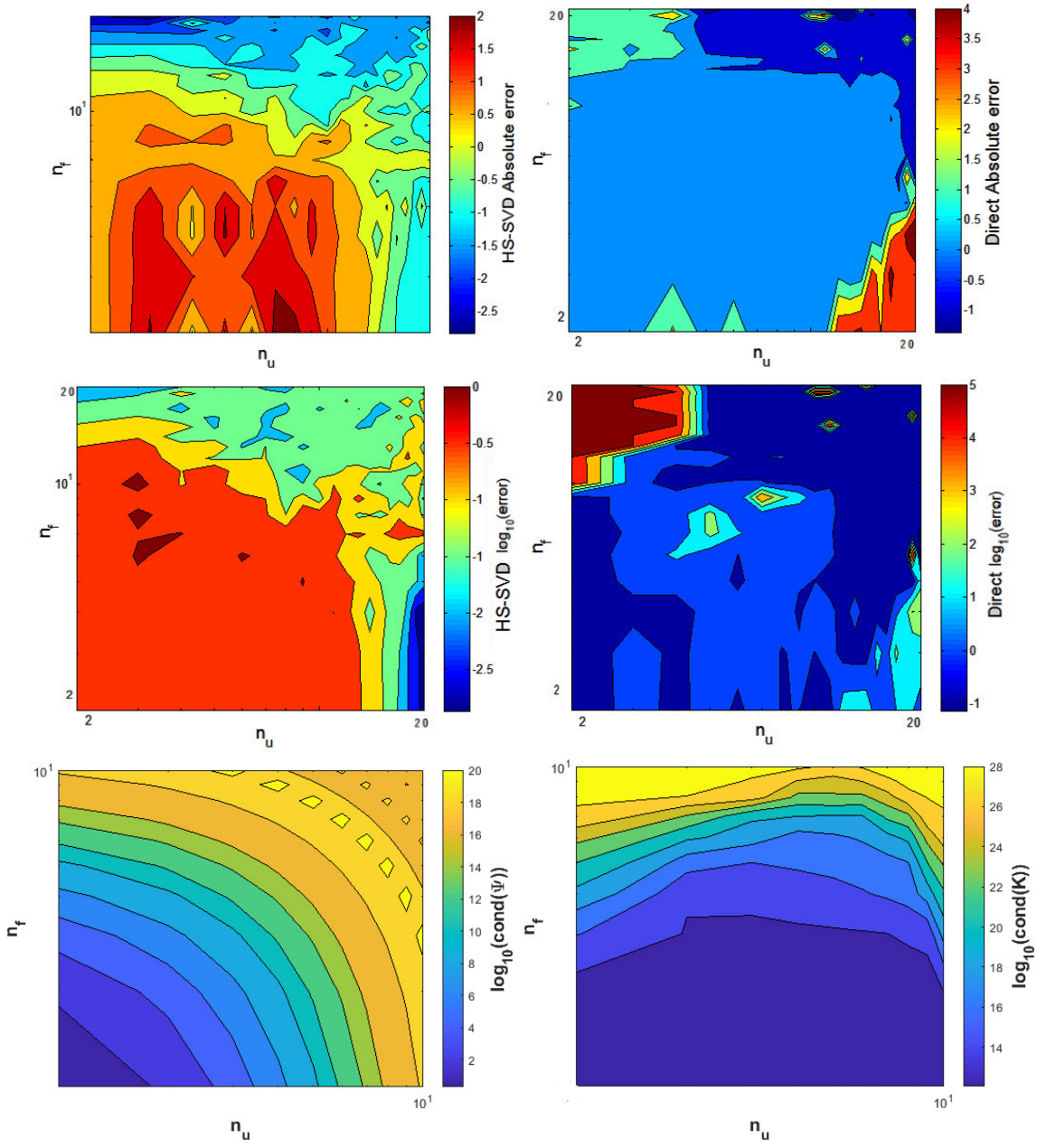

Figure 8: Maximum error between exact function $u(x)$ and predicted mean $\bar{u}(x)$ (top), exact function $f(x)$ and predicted mean $\bar{f}(x)$ (middle) in the logarithmic scale as a function of the total number of training points $u(x)$ and $f(x)$, denoted by n_u and n_f, are demonstrated with HS-SVD and direct methods. The condition number of covariance matrix $\boldsymbol{K}$ and matrix Ψ (bottom) in the logarithmic scale are demonstrated using Chebyshev data points for Example 3.

Example 4 Consider the following integro-differential equation,

$$\mathcal{L}_x^{\alpha} u(x) = \frac{d}{dx}u(x) + 2u(x) + \alpha \int_0^x u(t)dt = f(x).$$

The functions $u(x) = sin(2\pi x)$ and

$$f(x) = 2\pi \cos(2\pi x) + 2\sin(2\pi x) - \alpha \frac{\cos(2\pi x) - 1}{2\pi}$$

satisfy the equation. We have used the data $\{x_u, y_u\}, \{x_f, y_f\}$ generated from $u(x)$ and $f(x)$ with $n_u = n_f = 20$ Chebyshev data points chosen in the interval $[0, 1]$ for $\alpha = 3$. We have demonstrated the effectiveness of the HS-SVD method using the generalized periodic spline kernel. Likelihood criterion is evaluated for $\beta = 3$ and α and ε uniformly spaced in $[.01, 10]$. The data is then used to make predictions at `Neval`=100 evenly spaced points in the domain, and the absolute error is displayed in Fig. 9. Also, likelihood criterion is evaluated for the values of ε logarithmically spaced in $[10^{-2}, 10^2]$ in Fig. 10. This data is then used to make predictions for different values β at `Neval`=100 evenly spaced points in the domain, and the absolute errors are displayed in Fig. 10. It is apparent that the HS-SVD method suffers no ill-conditioning. The maximum likelihood estimator is near the "optimal" error, though it does not precisely locate it. It is clear that by increasing the values of β the MLE direct computation loses accuracy and suffers a complete breakdown because $\boldsymbol{K}^{-1}$ is too ill-conditioned. Also, the absolute errors between the true parameter α and the estimated one, between the exact functions $u(x)$ and $f(x)$ and the predicted means $\bar{u}(x)$ and $\bar{f}(x)$ and also the condition number of covariance matrix $\boldsymbol{K}$ and matrix Ψ using Chebyshev data points are demonstrated in Fig. 11. In Table 4, using the optimal (hyper) parameters the absolute error, the condition number, the optimal values of α and the likelihood criterion are presented for different values of (n_u, n_f).

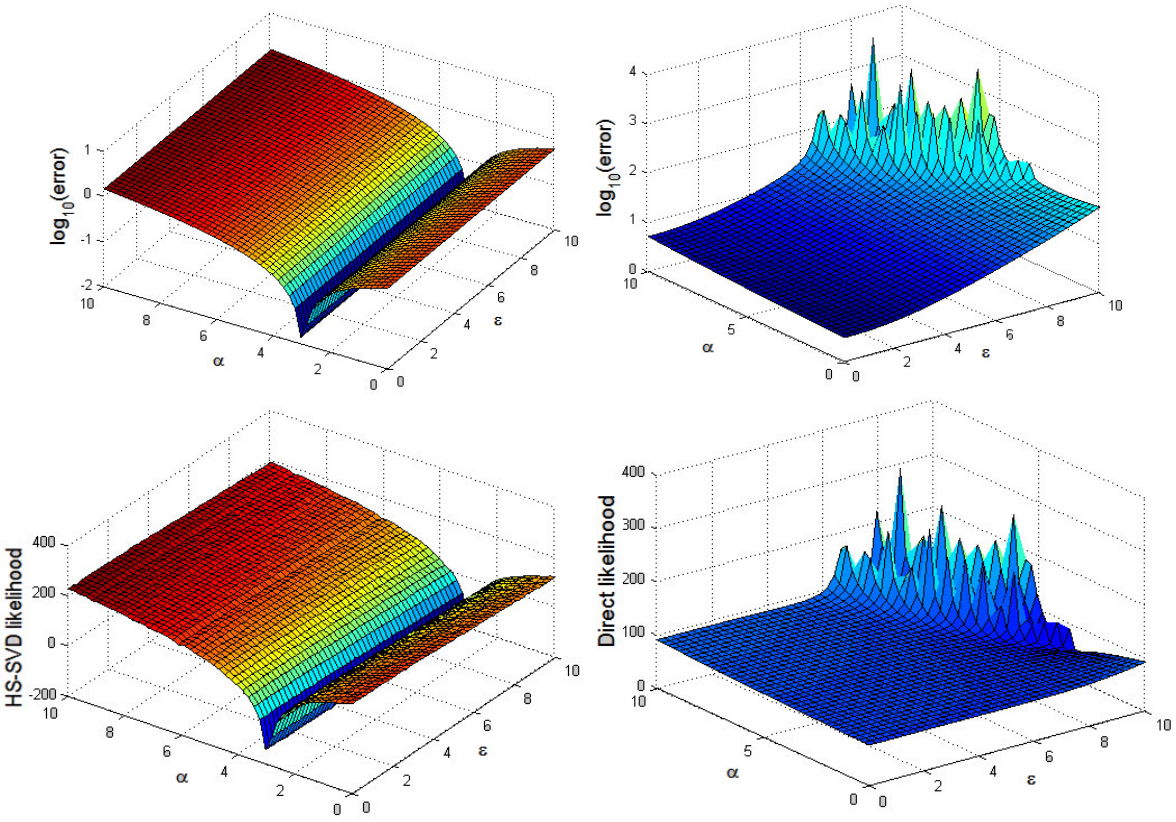

Figure 9: Comparison of Negative Log Marginal likelihood (NLML) criterion computed with both methods for Example 4. The top row shows the error of the posterior mean based on Chebyshev data points using Chebyshev kernel. The bottom row displays the corresponding likelihood estimates.

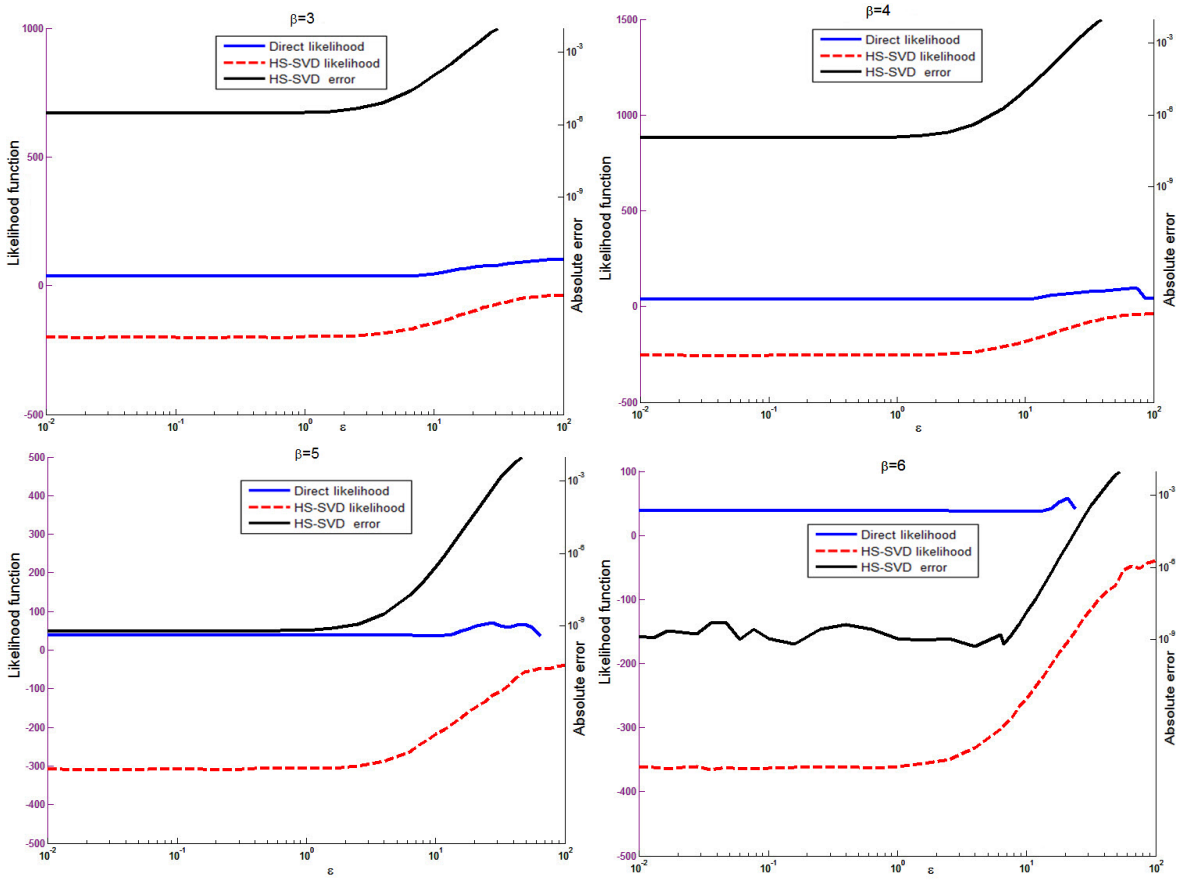

Figure 10: Comparison of the absolute error and MLE estimators of the optimal shape parameter ε for Example 4 using generalized periodic spline kernel computed via both approaches for different values of β.

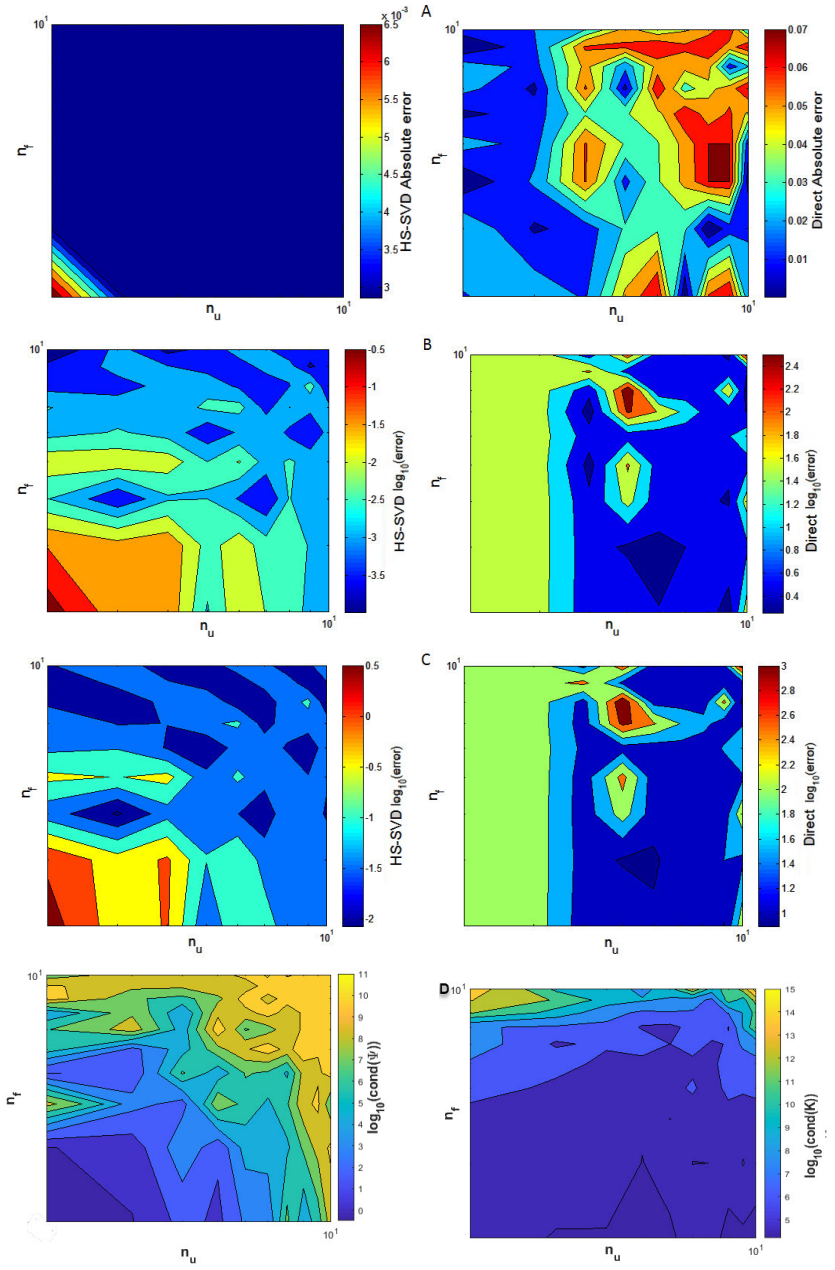

Figure 11: The absolute error between the true parameter α and the estimated one (A), between exact function $u(x)$ and predicted mean $\bar{u}(x)$ in the logarithmic scale (B), between exact function $f(x)$ and predicted mean $\bar{f}(\boldsymbol{x})$ in the logarithmic scale (C) are demonstrated with both methods. The condition number of covariance matrix $\boldsymbol{K}$ and matrix Ψ in the logarithmic scale is demonstrated using Chebyshev data points for Example 4 (D).

Table 4: Parameter estimation α and the absolute error of operator posterior prediction for optimal (hyper)parameters obtained with both methods using Chebyshev data points for Example 4.

			HS-SVD method				Direct method	
(n_u, n_f)	α	HSmle	cond(Ψ)	error	α	Dmle	cond($\boldsymbol{K}$)	error
$(5,5)$	2.98	-225.93	1.5863e+05	0.033	3.0665	-32.7392	5.6795e+09	120.0447
$(10,10)$	3.008	-565.1892	2.0447e+11	0.0085	2.9365	-62.1680	2.4611e+13	590.8914
$(20,20)$	3.0016	-309.2043	5.3467e+16	0.0013	3.8837	23.6461	1.5684e+20	35.3956

436 **Example 5** ***(Transport Equation)*** Consider the following differential equation,

$$\mathcal{L}^{\zeta}_{(x,t)}u(x,t) = \frac{\partial u(x,t)}{\partial t} + \zeta\frac{\partial u(x,t)}{\partial x} = f(x,t).$$

437 The functions $u(x,t) = \exp(-x)\sin(2\pi t)$ and

$$f(x,t) = 2\pi\exp(-x)\cos(2\pi t) - \zeta\exp(-x)\sin(2\pi t)$$

satisfy the equation. We have used the data $\{\boldsymbol{x}_u = (x_u, t_u), \boldsymbol{y}_u\}, \{\boldsymbol{x}_f = (x_f, t_f), \boldsymbol{y}_f\}$ generated by $\boldsymbol{y}_u = u(x_u, t_u)$ and $\boldsymbol{y}_f = f(x_u, t_u)$ with $n_u = n_f = 32$ Halton data points chosen in the interval $[0,1]^2$ for $\zeta = 1$. We have also demonstrated the effectiveness of the HS-SVD method using Squared Exponential kernel with eigenvalues and eigenfunctions

$$\lambda_n = \sqrt{\frac{\alpha^2}{\alpha^2+\delta^2+\varepsilon^2}}\left(\frac{\varepsilon^2}{\alpha^2+\delta^2+\varepsilon^2}\right)^{n-1} \quad n = 1, 2, \ldots$$

$$\varphi_n(x) = \gamma_n e^{-\delta^2 x^2} H_{n-1}(\alpha\beta x),$$

where the H_n are Hermite polynomials of degree n, and

$$\beta = \left(1 + (\frac{2\varepsilon}{\alpha})^2\right)^{\frac{1}{4}},\ \gamma_n = \sqrt{\frac{\beta}{2^{n-1}\Gamma(n)}},\ \delta^2 = \frac{\alpha^2}{2}(\beta^2 - 1),$$

are constants such that they are defined in terms of the shape parameter ε and the parameter α in the weight function $\rho(x) = \frac{\alpha}{\sqrt{\pi}}e^{-\alpha^2 x^2}$ of the Hilbert-Schmidt integral operator. The multivariate case is easily obtained using the tensor product form of Squared Exponential kernel, i.e., for d-variate functions we have

$$\boldsymbol{\lambda}_n = \Pi_{j=1}^2\lambda_{n_j} = \Pi_{j=1}^2\sqrt{\frac{\alpha_j^2}{\alpha_j^2+\delta_j^2+\varepsilon_j^2}}\left(\frac{\varepsilon_j^2}{\alpha_j^2+\delta_j^2+\varepsilon_j^2}\right)^{n_j-1},$$

and

$$\varphi_n(\boldsymbol{x}) = \Pi_{j=1}^2 \varphi_{n_j}(x_j) = \Pi_{j=1}^2 \gamma_{n_j} e^{-\delta_j^2 x_j^2} H_{n_j-1}(\alpha_j \beta_j x_j),$$

where $\boldsymbol{x} = (x_1, x_2) \in \mathbb{R}^2$. For more on tensor product kernels we point the reader to [16, 22].

Note that this formulation allows us to take different shape parameters ε_j and integral weights α_j for different space dimensions (i.e., k_{uu} may be an anisotropic kernel), or we can take them all equal, i.e., $\alpha_j = \alpha$ and $\varepsilon_j = \varepsilon,\ j = 1, 2$ (and then k_{uu} is isotropic or radial) [22]. In the example, we restrict ourselves to using the same α_j and ε_j in all dimensions. Likelihood criterion is evaluated by a fixed value of $\alpha = 3$ and a grid of 625 different values of $[\zeta, \varepsilon]$ with each component uniformly spaced in $[.01, 10]$. The data is then used to make predictions at `Neval`=81 evenly spaced points in the domain. The absolute error and likelihood criterion are displayed in Fig. 12. In Table 5, the absolute error, the condition number, the optimal values of ζ and the likelihood criterion are presented for different values of (n_u, n_f).

Table 5: Parameter estimation ζ and the absolute error of operator posterior prediction for optimal (hyper)parameters obtained with both methods using Halton data points for Example 5.

			HS-SVD method				Direct method	
(n_u, n_f)	ζ	HSmle	cond($\boldsymbol{\Psi}$)	error	ζ	Dmle	cond($\boldsymbol{K}$)	error
$(8, 8)$	0.7	-60.2578	2.3783e+04	0.0577	10	0.2091	6.9541e+12	1.0475
$(16, 16)$	0.8577	- 130.6324	5.6463e+04	0.0205	3.0665	-32.7392	5.6795e+14	0.5317
$(32, 32)$	1.0771	-987.4220	5.3892e+10	.002524	7.4296	-495.7323	2.7838e+17	1.03254

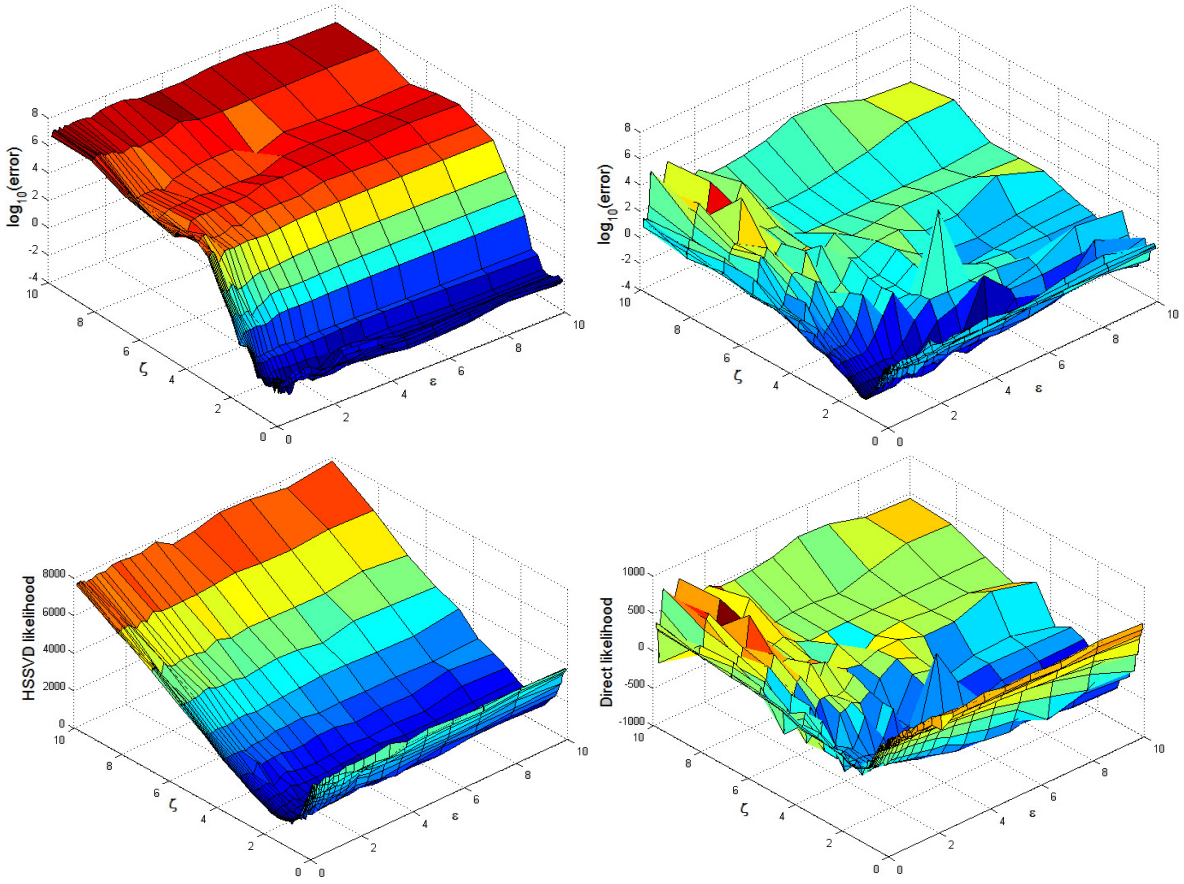

Figure 12: Comparison of Negative Log Marginal likelihood (NLML) criterion computed for $\alpha = 1$ with both methods for Example 5. The top row shows the error of the posterior mean based on Halton data points using Squared Exponential kernel. The bottom row displays the corresponding likelihood estimates.

5 Conclusion

In this paper, we made the unified framework to deal with parametric linear operational equation in (1) which was probabilistically approximated by employing the GPs, and was made computationally more stable and reliable by developing a novel computational strategy for more adaptive parameters and hyperparameters learning leading to more accurately predicting

operators at some unseen operational data points. The standard computational strategies suggested for solving the above linear inverse problems would usually become severely ill-conditioned to estimate the model parameters and hyperparameters, particularly when the number of data points increase, and the flat kernels (e.g., the squared exponential kernel with a small shape parameter, ε) are used. It is evident that by increasing the number of the observed data, the direct approach to estimating parameters and hyperparameters and predicting the operators would become less accurate, and it does not then correctly identify a region for "optimal" (hyper)parameters that match the region of the smallest error. As a result, it will suffer a complete breakdown, because $\boldsymbol{K}^{-1}$ is too ill-conditioned. In this paper, we proposed an alternative computational approach using the HS-SVD at which the computation of the likelihood function becomes more stable, and the determination of the MLEs for optimal posterior predictions is now possible. It is thus apparent that the HS-SVD method correctly identifies a region for "optimal" (hyper)parameters that match the region of the smallest error. The proposed approach was validated by illustrating it in several benchmark problems and various kernels with different attributes. The numerical illustrations confirm the stability of the proposed method, particularly when the number of the observed data points increase, and the flat kernels are used when the standard computational strategies reviewed above would be unable to handle. The absolute error, condition number, the optimal values of (hyper)parameters and also likelihood criterion are presented for different number of data points for both approaches of direct and HS-SVD. We found out that the HS-SVD method correctly determines a region for an "optimal" (hyper) parameters estimate, while instability was witnessed in the direct approach. It was clear that by increasing the number of training points, the direct approach in parameters estimation and operators posterior prediction loses accuracy and suffers a complete breakdown because $\boldsymbol{K}^{-1}$ is too ill-conditioned. Needless to say, the process of reducing the condition number, by the HS-SVD approach, also varies from one problem to another, depending on the type and nature of the model, the number of training points and even the type of the training points (e.g., Chebyshev points, Halton points, random points and and so forth). Therefore, the ill-conditioning improvement in the various examples may be different.

Future researchers are highly recommended to investigate the technique introduced in this paper for the problem of learning nonlinear operational equations as discussed in [5]. The main problem in nonlinear operational equations is that the stated condition "linear transformation

of a GP such as differentiation and integration remains GP" is no longer valid, and a linear approximation of a nonlinear operator must be used. Furthermore, the results gained from this study are limited to those special kernels for which a Hilbert-Schmidt SVD is available (the collection of positive definite kernels and their known Mercer series are presented in [16]). Therefore, understanding how a Mercer series with numerically computed eigenvalues and eigenfunctions affects the quality of these computations can allow this strategy to be applied to a wider range of kernels (which is currently limited by the availability of the Mercer series [17]).

The work in this paper is limited to low dimension in $\boldsymbol{x}$, as a tensor product basis is used in terms of eigenfunctions. This removes an important advantage of kernel methods, which are formally dimension-independent. Despite this limitation, we still have the advantage over using regression in a spline basis that we can see the parameters of the operational equations as kernel hyperparameters and estimate them using a machine learning strategy. At the same time, in our future studies, we are trying to generalize the proposed method in this paper to high dimensional problems without being dependent on the tensor product.

References

[1] Bender, E. A., *An Introduction to Mathematical Modeling*, New York: DOVER PUBLICATIONS INC., Mineola, 1978.

[2] Vries, K. de, Nikishova, A., Czaja, B., Zavodszky, G. and Hoekstra, A. G., Inverse uncertanity quantification of a cell model using a Gaussian process metamodel ,*Int. J. Uncertainty Quantif.*, **10**(4), pp. 333-349, 2020.

[3] Narayan, A., Yan, L. and Zhou, T., Optimal design for kernel interpolation: Applications to uncertainty quantification, *J. Comput. Phys.*, **430**, pp. 1-20, 2021.

[4] Qin, T., Chen, Z., Jakeman, J. D. and Xiu, D., Deep learning of parameterized equations with applications to uncertanity quantification, *Int. J. Uncertainty Quantif.*, **11**(2), pp. 63-82, 2021.

[5] Raissi, M., Perdikaris, P. and Em Karniadakis, G., Numerical Gaussian Processes for Time-Dependent and Nonlinear Partial Differential Equations, *SIAM J. Sci. Comput.*, **40**(1), pp. 172-198, 2018.

[6] Raissi, M., Perdikaris, P. and Em Karniadakis, G., Machine learning of linear differential equations using Gaussian processes, *J. Comput. Phys.*, **348**(1), pp. 683-693, 2017.

[7] Mueller, J. L. and Siltanen, S., *Linear and Nonlinear Inverse Problems with Practical Applications*, Computational Science and Engineering, vol. 10. Philadelphia, PA, USA: SIAM, 2012.

[8] Neto, F. D. M. and Neto, A. J. d. S., *An Introduction to Inverse Problems with Applications*, Springer-Verlag Berlin Heidelberg, 2013.

[9] Williams, C. K. and Rasmussen, C. E., *Gaussian processes for machine learning*, the MIT Press, 2006.

[10] Murphy, K. P., *Machine learning: a probabilistic perspective*, MIT press, 2012.

[11] Vapnik, V., *The nature of statistical learning theory*, Springer Science & Business Media, 2013.

[12] Guo, M. and Hesthaven, J. S., Reduced order modeling for nonlinear structural analysis using Gaussian process regression, *Comput. Meth. Appl. Mech. Eng.*, **341**(1), pp. 807-826, 2018.

[13] Tikhonov, A., Solution of incorrectly formulated problems and the regularization method, *Soviet Math. Dokl.*, **5**, pp. 1035-1038, 1963.

[14] Poggio, T. and Girosi, F., Networks for approximation and learning, *Proceedings of the IEEE*, **78**, pp. 1481-1497, 1990.

[15] Fasshauer, G. E., *Meshfree Approximation Methods with Matlab*, Interdisciplinary Mathematical Sciences, Vol. 6, World Scientific Publishing, Singapore, 2007.

[16] Fasshauer, G. E. and McCourt, M., *Kernel-based Approximation Methods using MATLAB, Interdisciplinary Mathematical Sciences*, World Scientific Publishing, Singapore, 2015.

[17] McCourt, M., and Fasshauer, G. E., *Stable Likelihood Computation for Gaussian Random Fields, Recent Applications of Harmonic Analysis to Function Spaces, Differential Equations, and Data Science. Applied and Numerical Harmonic Analysis*, Birkher, Cham, 2015.

[18] Kohn, R., Ansley, C. F. and Tharm, D., The performance of cross-validation and maximum likelihood estimators of spline smoothing parameters, *J. Am. Stat. Assoc.*, **86**(416), pp. 1042-1050, 1991.

[19] Fornberg, B., Larsson, E. and Flyer, N., Stable Computations with Gaussian Radial Basis Functions, *SIAM J. Sci. Comput.*, **33**(2), pp. 869-892, 2011.

[20] Pazouki, M. and Schaback, R., Bases for kernel-based spaces, *J. Comput. Appl. Math.*, **236**(4), pp. 575-588, 2011.

[21] Quang, M. H., Niyogi, P. and Yao, Y., *Mercer's Theorem, Feature Maps, and Smoothing*, In International Conference on Computational Learning Theory (COLT), 2007.

[22] Fasshauer, G. E. and McCourt, M., Stable evaluation of Gaussian RBF interpolants, *SIAM J. Sci. Comput.*, **34**(2), pp. 737-762, 2012.

[23] Gulian, M., Raissi, M., Perdikaris, P. and Em Karniadakis, G., Machine learning of space-fractional differential equations, *SIAM J. Sci. Comput.*, **41**(4), pp. 2485-2509, 2019.

[24] Esmaeilbeigi, M., Chatrabgoun, O. and Cheraghi, M., The Role of Hilbert-Schmidt SVD basis in Hermite-Birkhoff interpolation in fractional sense, *Comput. Appl. Math.*, **38**(82), pp. 1-20, 2019, DOI: 10.1007/s40314-019-0849-x.

[25] Sacks, J., Welch, W. J., Mitchell, T. J. and Wynn, H. P., Design and Analysis of Computer Experiments, *Statistical Scienice*, **4**(4), pp. 409-435, 1989.

[26] Esmaeilbeigi, M., Chatrabgoun, O. and Cheraghi, M., Fractional Hermite interpolation using RBFs in high dimensions over irregular domains with application,*J. Comput. Phys.*, **375**, pp. 1091-1120, 2018.

[27] Rasmussen, C. E. and Ghahramani, Z., *Occam's razor*, Adv. Neural. Inform. Process. Syst., pp. 294-300, 2001.

[28] Aronszajn, N., Theory of reproducing kernels, *Transactions of the American mathematical society*, **68**, pp. 337-404, 1950.

[29] Saitoh, S., *Theory of reproducing kernels and its application*, Longman, 1988.

[30] Berlinet, A. and Thomas-Agnan, C., *Reproducing kernel Hilbert spaces in probability and statistics*, Springer Science & Business Media, 2011.

[31] Datta, B. N., *Numerical Linear Algebra and Applications*, second edition, SIAM, Philadelphia, 2010.

[32] Podlubny, I., *Fractional differential equations: an introduction to fractional derivatives, fractional differential equations, to methods of their solution and some of their applications*, Academic press, 1998.