Approximate GP Inference for Nonlinear Dynamical System Identification Using Data-Driven Basis Set

HINA SHOUKAT1, MUHAMMAD TAHIR1, (Member, IEEE), AND KHURRAM ALI2, (Member, IEEE)
1Department of Electrical Engineering, Lahore University of Management Sciences, Lahore 54792, Pakistan
2Department of Electrical and Computer Engineering, COMSATS University Islamabad, Lahore 54000, Pakistan
Corresponding author: Muhammad Tahir (tahir@lums.edu.pk)

ABSTRACT
We address the problem of nonlinear dynamical system identification in state space formulation using an approximate Gaussian process (GP) regression framework where the basis for GP are learned from data. Approximate GP inference is used to address the high computational cost of exact GP frameworks, and is based on basis function expansion concept. We address the design of these basis functions using data and eigenvalue decomposition framework. The need for such design arises in practical applications where the collected data has outliers, discontinuities and other non-stationary characteristics which limit not only the applicability but also the performance of traditional GP framework with inherent stationary assumptions. Using the available data set, a kernel eigenvalue problem is formulated which is solved using Monte Carlo techniques to construct basis functions. Then, a low-rank approximation to the exact GP is obtained by an approach similar to kernel principle component analysis (k-PCA). The coefficients in the basis function expansion and other unknown parameters are learned from data using sequential Monte Carlo technique. The proposed GP framework for dynamical system identification is tested and validated against fixed basis framework using simulations and experimental data.

INDEX TERMS
Gaussian processes regression, non-stationary kernel, kernel PCA, sequential Monte Carlo, kernel eigenvalue problem.

I. INTRODUCTION
In practical industry, a number of critical operations such as process control [1], [2], state estimation [3], process monitoring [4], fault diagnosis [5], gross error detection and data reconciliation [6] are mostly model based where a good model describes the underlying process dynamics with an accurate and correct mathematical formulation. Apart from industrial applications, many other areas such as economic data, biology and life sciences also require accurate identification of the underlying dynamical system model [7]. For most of these phenomenon, especially industrial, it is really difficult to synthesize the underlying models based on first-principles based learning due to their enormous complexity [8]. In such cases, the so called data driven black-box and grey-box modeling approaches have been emerged as promising and viable alternatives [9], [10]. Although, a lot of progress has already been made in data based system identification methods, there are still issues such as the presence of discontinuities and outliers in the data, dynamic nonlinearities and process noise modeling which make these techniques very difficult to apply for practical problems [8], [11], [12]. If the underlying system to be identified can be well modeled as linear, many efficient techniques are available in the literature for model identification [13]. However, in most of the real world applications, the underlying dynamical phenomenon exhibits a highly nonlinear structure whose identification from the observed input-output data set, has already been recognized as a challenging task [12].

A large number of research efforts have been made for nonlinear system identification especially by proposing specific system classes such as based on Volterra models [14], nonlinear ARMAX models [13], neural networks [15], fuzzy models [16], [17], linear parameter varying (LPV) models [18], nonlinear error in variable (EIV) models [19], Hammerstein-Wiener block structured models [20] and...
state-space models (SSM) [21]. In these classes, parametric form of identification relies strongly on model parameter estimation for a given fixed model structure candidate [22]. However, the potential model candidates are difficult to find, especially for complex high dimensional dynamical systems where complete knowledge of system dynamics is not known. Also, a pre-assumed model structure cannot capture the varying dynamics of underlying process or system accurately. On the other hand, non-parametric approaches provide a compelling alternative where they try to fit a single model that can adapt its complexity to the data size [23]. Gaussian process (GP) regression is a powerful Bayesian non-parametric approach which does not take any particular assumptions about the nonlinear function to be identified rather it views the function as a realization of a Gaussian random process [24].

GP regression is a powerful tool for functional inference that directly assumes distribution over unknown function. However, the major practical limitations of exact GP is that it suffers from a cubic time complexity $O(N^3)$ where $N$ corresponds to total number of training data samples [24]. This limits the applicability of GP in its direct form, especially when the size of training data is large; hence some approximation is needed. A recent and comprehensive review of scalable GP is provided in [25]. System identification can be performed wither in online or in off-line settings. In both cases, the underlying process usually involves dynamics. For such type of problems, recurrent GP models have been proposed in the literature such as GP-based nonlinear autoregressive models with exogenous inputs (GP-NARX) models [26] and GP-based state space models (GP-SSM) [27]. GP-SSM are superior than GP-NARX models, especially in handling the so-called error in variables problem of observation noise. Moreover, there exist fast algorithm which can speed up the learning process as well [28].

Standard GP framework has a common underlying assumption of stationarity, i.e., the covariance between function values is the function of only the distance between them; it is thus invariant to the location of input value pairs. However, this assumption of global smoothness is often violated in practice, e.g., in modeling spatial processes that contain discontinuities as in the geo-sciences, robotics, mobility mining etc. Standard GP with fixed basis functions cannot learn sharp discontinuities. Mackay [29], and Bengio et al. [30] argued that the meaningful representations in high dimensional data can be automatically discovered using adaptive basis functions. Based on similar lines, this problem of non-stationarity is dealt by making adaptations to the stationary GP methodology. An interesting approach towards this is piecewise Gaussian process such as Treed GP [31]. It involves partitioning the input space into disjoint regions for each discontinuity such that the problem becomes almost stationary and then training GP to model data on each of those regions. Similar approach is also used by Svensson in his work [32]. However, this approach has underlying assumption of known number of discontinuity points and their locations, which further enhances the computational complexity.

In this paper, we present a new formulation for approximate GP based on basis function expansion for high dimensional dynamical systems identification using a set of basis functions which are learned from data. We extend the reduced rank GP to use any covariance function by formulating kernel eigenvalue problem which is solved using Monte Carlo techniques. Using this approach, we are able to incur any type of kernel - non-stationary to stationary kernels- in our formulation. An important aspect of our approach is that single choice of the kernel and a single set of hyperparameters will be sufficient to model the data. It is well known that among all orthogonal basis functions including trigonometric basis functions, eigenfunctions provide most compact representation [33]. Following this, the main advantage with our approach over others present in the literature e.g. [32], is that our approach seeks to estimate the optimal Karhunen-Loeve (KL) eigenbasis for reduced rank GP approximation instead of restricting to trigonometric basis. Secondly, we use reduced rank approximation, similar to kernel Principal Component Analysis (PCA), that uses small subset of eigenfunctions for robustness. For learning proposed GP based state-space model, we perform full Bayesian computations employing computationally efficient particle filter method [34]. We assess the quality of our identification approach by comparing it with the formulation in [32]. Comprehensive experiments on real world and synthetic data, show the effectiveness and feasibility of our approach. To the best of our knowledge, it is the first time that both stationary and non-stationary GPs have been learned using single set of basis functions. The true advantage of the proposed framework lies in identifying functions which do not obey the assumption of global smoothness rather they exhibit discontinuities more frequently. Note that there exists some works in the literature which use similar type of frameworks but for other types of models such as to synthesize a periodic latent force [35]. Our work deals with the design of basis functions from data in GP-SSM context.

The rest of the paper is organized as follows. We present the basic overview of the problem formulation in section II along with its extension to SSM. Section III outlines the proposed framework for designing basis function inside GP based SSM. The learning framework is also described in the same section where the unknown parameters of SSM are learned along with system states using Bayesian learning. In section IV, we present the numerical results and carry out the discussion to compare the performance of the proposed framework with a reference framework. Based on this discussion, some conclusions are drawn in section V.

II. GP REGRESSION AND PROBLEM FORMULATION

A. REGULAR GP REGRESSION

GP regression deals with approximating a scalar-valued function $f(x)$ through a number $T$ of input-output training
points, denoted as \((x_1, y_1), (x_2, y_2), \ldots, (x_T, y_T)\) which are assumed to be generated from the model
\[
y_t = f(x_t) + \epsilon_t \quad \text{with} \quad \epsilon_t \sim \mathcal{N}(0, \sigma^2)
\]
(1)
where \(x_t \in \mathcal{R}^{n_x}\) is input and \(y_t\) is the scalar output variable. The main idea behind GP based regression framework is to use it as a way to encode prior information over the function \(f(\cdot)\) such that it can be viewed as a realization of a Gaussian process with the mean function \(m(x) = E[f(x)]\) and the covariance function (also called kernel) \(k(x, x') = E[(f(x) - m(x))(f(x') - m(x'))]\) as
\[
f(x) \sim \mathcal{GP}(m(x), k(x, x'))
\]
(2)
where \(x\) represents any point in the domain of the function \(f(\cdot)\). We can collect \(T\) total points as \(X = \{x_1, x_2, \ldots, x_T\}\), to obtain \(T\) latent values of the function \(f(\cdot)\) at these points as \(f = f(X) = \{f(x_1), f(x_2), \ldots, f(x_T)\}\). Similarly, corresponding outputs can be collected as \(y = \{y_1, y_2, \ldots, y_T\}\) such that
\[
y = f + \epsilon \quad \text{with} \quad \epsilon \sim \mathcal{N}(0, \sigma^2I)
\]
(3)
According to the definition in (2), \(f\) is a \(T\)-variate Gaussian random variable given as [24]
\[
f \sim \mathcal{N}(m(X), K(X, X))
\]
(4)
where the mean vector is
\[
m(X) = \begin{bmatrix} m(x_1) \\ m(x_2) \\ \vdots \\ m(x_T) \end{bmatrix}
\]
(5)
and the covariance matrix \(K(X, X)\) is
\[
K(X, X) = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_T) \\ \vdots & \ddots & \vdots \\ k(x_T, x_1) & \cdots & k(x_T, x_T) \end{bmatrix}
\]
(6)
In order to predict the function values at specific test points collected as \(X^* = \{x_1^*, x_2^*, \ldots, x_N^*\}\), and \(f^* = f(X^*)\) with \(N\) being the total number of test points, we assume a joint Gaussian prior as
\[
\begin{bmatrix} f \\ f^* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m(X) \\ m(X^*) \end{bmatrix}, \begin{bmatrix} K(X, X) & K(X, X^*) \\ K(X^*, X) & K(X^*, X^*) \end{bmatrix}\right)
\]
(7)
\[
\begin{bmatrix} f \\ f^* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m \\ m^* \end{bmatrix}, \begin{bmatrix} K & K^* \\ K^* & K^{**} \end{bmatrix}\right)
\]
(8)
where (8) introduces shorthand notations for the quantities in (7). From (8), the posterior distribution of both \(f\), and \(f^*\) given \(y\) can be found using (9),
\[
\begin{bmatrix} f \\ f^* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu \\ \mu^* \end{bmatrix}, \begin{bmatrix} \Sigma & \Sigma^* \\ \Sigma^{**} & \Sigma^{**} \end{bmatrix}\right)
\]
(9)
with
\[
\mu = \left(\frac{1}{\sigma^2}I + K^{-1}\right)^{-1} K^{-1} \begin{bmatrix} m \\ \mathbf{y} \end{bmatrix}
\]
\[
\Sigma = \left(\frac{1}{\sigma^2}I + K^{-1}\right)^{-1}
\]
\[
\Sigma^* = \sigma^2 I \left(\frac{1}{\sigma^2}I + K^{-1}\right)^{-1} \Sigma
\]
\[
\Sigma^{**} = K^{**} \left(\frac{1}{\sigma^2}I + K^{-1}\right)^{-1} \sigma^2 I
\]
\[
\Sigma^{**} = K^{**} \left(\frac{1}{\sigma^2}I + K^{-1}\right)^{-1} \sigma^2 I
\]
Note that in (7) for joint Gaussian prior, the mean function is mostly set to zero and the covariance function has a parametric form that encodes a prior notion of smoothness or stationarity on \(f(\cdot)\). The most common choice for the covariance function is the squared exponential covariance function given by
\[
k(x, x') = \sigma^2 \exp\left(-\frac{\|x - x'\|^2}{2\ell^2}\right)
\]
(10)
where \(\sigma^2\) and \(\ell\) are the hyper-parameters. We can combine them in a parameter vector \(\theta_{hyp} = [\sigma^2 \ell]^T\). The hyper-parameter vector needs to be marginalized out or set by the user [24]. The squared exponential is a stationary covariance function which results in a stationary GP model. It works well as long as the unknown function is globally smooth and does not exhibit non-stationarities such as jumps and discontinuities. However, practical systems might exhibit such type of behavior which will affect the performance of GP with stationary covariance function. In order to illustrate this point further, we consider the estimation of following two functions using regular GP regression with squared-exponential covariance function. Both functions are defined for \(-5 \leq x \leq 5\) and are given as
\[
f_1(x) = \sin(3x)
\]
\[
f_2(x) = \begin{cases} 20 & \text{for } 0 \leq x \leq 1 \\
-1 & \text{for } 1 < x \leq 2 \\
3 & \text{for } 2 < x \leq 3 \\
\sin(3x) & \text{otherwise} \end{cases}
\]
Here, function \(f_1(x)\) is globally smooth function while \(f_2(x)\) has jumps or discontinuities. The noisy data observations are collected according to (1) using \(\sigma^2 = 0.03\), and \(T = 150\) for both functions. Using these noisy data observations, function values are estimated using (9). We evaluate the prediction performance using \(N_{test} = 150\) test points in terms of root mean square error (RMSE) metric defined in section IV. Fig. 1 displays the estimation results using squared-exponential covariance function for GP prior. It is clear that the performance of regular GP regression is very good for predicting the values of \(f_1(\cdot)\) while the error has increased many times for \(f_2(\cdot)\). The notion of global smoothness as assumed by squared-exponential covariance function is not more applicable and hence the performance is deteriorated. In this paper, we address this problem where the unknown function is expanded in terms of basis functions.
whose structure is dictated from data itself which allows to capture the jumps and discontinuities in the unknown function with enhanced accuracy.

B. GP-SSM

In this section, we extend the model in (1) which can incorporate standard SSM in order to learn nonlinear state transition function. For that purpose, consider the following standard nonlinear SSM

\[
\begin{align*}
x_t &= f(x_{t-1}, u_{t-1}) + q_{t-1} \\ y_t &= g(x_t) + r_t
\end{align*}
\]

(11)

where \( t \) indicates the time index, \( u_{t-1} \in \mathbb{R}^{n_u} \) is the system input or control signal, \( x_t \in \mathbb{R}^{n_x} \) is the system state vector, \( y_t \in \mathbb{R}^{n_y} \) is the system output, \( q_{t-1} \sim \mathcal{N}(0, Q) \) is the process noise vector, and \( r_t \sim \mathcal{N}(0, R) \) is the measurement noise vector. The vector-valued unknown function \( f(.) \) in (11), is viewed as a collection of \( n_x \) scalar valued functions \( f_i(.) \) for \( i = 1, 2, \ldots, n_x \) as

\[
f(.) = \begin{bmatrix} f_1(.) \\ f_2(.) \\ \vdots \\ f_{n_x}(.) \end{bmatrix}
\]

(13)

The vector valued functions \( f(.) \) and \( g(.) \) are state transition function and observation function respectively. In normal state estimation problem, we assume that these functions are known along with system noise statistics, and the goal is to obtain an estimate of the state vector. The goal of system identification, however, is to estimate these functions along with system state and noise statistics which makes the problem really challenging. Two common approaches towards identifying models are parametric and non-parametric frameworks. Parametric frameworks are the classical approaches which try to replace \( f(.) \) and \( g(.) \) with much simple parametric models. However, it is really difficult to identify an appropriate parametric model which can reproduce all the complexity of the underlying phenomenon. Towards this end, nonparametric forms of models do not utilize any a priori model structure rather they obtain a model structure directly from the available training data. These methods are much more successful due to availability of huge amount of datasets recorded using sensors or obtained through numerical simulations [7], [36], [37]. Exact GP based frameworks are nonparametric in nature but suffer from the problem of higher computational complexity. Starting from (11) and (12), below, we will formulate a GP based statistical regression framework which takes as an input a set of input-output data samples for \( t = 1, 2, \ldots, T \), and the prior knowledge about \( n_x \), and estimates the state transition function \( f(.) \) along with the process noise covariance matrix \( Q \). In this treatment, we treat \( g(.) \) as a known function, although the developed approach can also be extended towards its identification.

The vector-valued unknown function \( f(.) \) in (11), is viewed as a collection of \( n_x \) scalar valued functions \( f_i(.) \in \mathbb{R} \) for \( i = 1, 2, \ldots, n_x \) as

\[
f(.) = \begin{bmatrix} f_1(.) \\ f_2(.) \\ \vdots \\ f_{n_x}(.) \end{bmatrix}
\]

(13)

Considering the identification of any single \( f_i(.) \) scalar valued function, we view it as a realization of a Gaussian process. Further, we can use a low-dimensional, parametric model for \( f_i(.) \) by representing the GP in terms of \( M \)-rank truncation of basis function expansion, parameterized as

\[
f_i(x, u) = \sum_{m=1}^{\infty} w^{(i)}_m \phi_m \approx \sum_{m=1}^{M} w^{(i)}_m \phi_m
\]

(14)
where \( \phi_m \) are **appropriate** basis functions whose weights are given by \( w_m^{(i)} \) for \( m = 1, 2, \ldots, M \). We can write combination of SSM in (11) and basis function expansion in (14) as

\[
\mathbf{x}_t = \begin{bmatrix}
  w_1^{(1)} & w_2^{(1)} & \cdots & w_M^{(1)} \\
  w_1^{(2)} & w_2^{(2)} & \cdots & w_M^{(2)} \\
  \vdots & \vdots & \ddots & \vdots \\
  w_1^{(n)} & w_2^{(n)} & \cdots & w_M^{(n)} \\
\end{bmatrix}
\begin{bmatrix}
  \phi_1 \\
  \phi_2 \\
  \vdots \\
  \phi_M \\
\end{bmatrix} + \mathbf{q}_{t-1}
\]

The model in (15) is in a parametric form which allows us to make inference on a rather large set of parameters contained in unknown weight matrix along with the system state \( \mathbf{x}_t \), and process noise covariance matrix \( \mathbf{Q} \). This identification then helps in identifying the function \( f(.) \) according to (14), provided basis functions are known.

### III. PROPOSED FRAMEWORK

In this section, we describe the procedure to obtain the model parameters starting from the model structure in (15). This involves the design of basis functions which are derived from data itself, and Bayesian learning step according to the overall system block diagram as shown in Fig. 2.

**FIGURE 2. Proposed framework.**

### A. BASIS FUNCTION DESIGN

There are many different forms of basis functions available in the literature e.g., polynomials, Gaussian kernels and Fourier basis. However all of these different alternates either apply to stationary GP or non-stationary GP but not to both. The reason for this lies in the design formulation of these functions. As an example, Fourier basis functions result as eigenfunctions to the Laplacian operator. We provide a systematic approach towards choosing a set of basis function and prior on \( w_j \). Using KL expansion for the given kernel, we form an explicit link between GP and the basis function expansion which is used to assign prior over the weights of basis functions. This is in contrast to the approach given in [38] which strictly requires the kernel to be isotropic and stationary. Our idea is to use the compact representation for GP prior using truncated Mercer expansion of its kernel in terms of the eigenfunctions of kernel. These eigenfunctions can then be used as basis functions as they happen to be orthonormal.

1) KERNEL EIGENFUNCTION APPROXIMATION

Any given covariance function or kernel of GP can be expanded w.r.t. a measure \( p(y) \) in terms of its eigenvalues \( \lambda_j \) and orthonormal eigenfunctions \( \psi_j(y) \) as [24]

\[
\int k(y, z)\psi_j(y)p(y)dy = \lambda_j \psi_j(z)
\]

where \( y \) and \( z \) are any two points in the domain of kernel. Assuming a uniform \( p(y) \) for available \( N \) points, we can approximate (16) as

\[
\frac{1}{N} \sum_{n=1}^{N} k(y_n, z)\psi_j(y_n) \approx \lambda_j \psi_j(z)
\]

Now, we can substitute \( z = z_n \) for \( n = 1, 2, \ldots, N \) into (17) to obtain the eigenvalue problem in \( N \times N \) matrix form which produces the estimate for \( j \)-th eigenfunction as

\[
\psi_j(z) \approx \sqrt{N} \sum_{n=1}^{N} k(y_n, z)\psi_j(y_n)
\]

\[
= \sqrt{N} k^T(y)\psi_j(y_n)
\]

where \( k^T(y) = [k(y_1, z), \ldots, k(y_N, z)] \), \( \psi_j(y_n) \) is \( j \)-th eigenvector of \( N \times N \) covariance function matrix evaluated at \( N \) data points, corresponding to eigenvalue \( \lambda_j \), and \( \psi_j(y_n) \approx \sqrt{N} \psi_j(y_n) \). We can interpret \( j \)-th eigenfunction in (20), as linear combination of kernel values evaluated at \( N \) data points, where the weights of linear combinations are provided by \( \psi_j(y_n) \). There are in general as many eigenfunctions as data points \( N \) i.e., \( j = 1, 2, \ldots, N \). Plainly, increasing the number of sample points increases the quality of approximation. However as \( N \) becomes larger, performing eigenvalue decomposition of kernel matrix may be computationally infeasible.

2) FINDING KERNEL EIGENFUNCTIONS FOR GP-SSM

In order to keep the computational complexity low, we propose to find eigenfunctions in off-line settings. Consider the scenario where we know the range on which our state vector \( \mathbf{x} \in \mathcal{R}^{n_x} \) may lie. Also, from the available data, we can get an idea about the ranges of our input vector \( \mathbf{u} \in \mathcal{R}^{n_u} \) and output vector \( \mathbf{y} \in \mathcal{R}^{n_y} \). Let \( \mathbf{a}_x \) and \( \mathbf{b}_y \) denote the minimum and maximum range values for \( \mathbf{x} \), \( \mathbf{a}_u \) and \( \mathbf{b}_u \) denote the corresponding minimum and maximum range values for \( \mathbf{u} \), and \( \mathbf{a}_y \) and \( \mathbf{b}_y \) denote the corresponding minimum and maximum range values for \( \mathbf{y} \). For \( \mathbf{x} \), we select \( N \) equally spaced points over the respective ranges of each state variable. For \( \mathbf{u} \) and \( \mathbf{y} \), we resample the available data to \( N \) equally spaced points over their respective ranges and generate \( N \) samples for each input \( u_k \) for \( k = 1, 2, \ldots, n_u \), and output \( y_l \) for \( l = 1, 2, \ldots, n_y \) time series. Once we have generated these points, we form \( N \times N \) kernel covariance matrix for each state, input and output variable. Performing eigenvalue decomposition of these \( n_x + n_u + n_y \) kernel matrices produces \( N(n_x + n_u + n_y) \) eigenfunctions according to (20). We label \( N \) eigenfunctions corresponding to each state variable \( \chi_j \) as \( \psi_j^{(n)} \) for \( j = 1, 2, \ldots, N \). Similarly, we label \( N \) eigenfunctions corresponding to each input variable \( u_k \) as \( \psi_j^{(u_k)} \) for
j = 1, 2, . . . , N, and those corresponding to each output variable \( y_j \) as \( \psi_j^{(y)} \) for \( j = 1, 2, . . . , N \).

3) REDUCED RANK APPROXIMATION
In general there are \( N \) eigenfunctions corresponding to each state, input and output variable. However, we keep only top \( M \) eigenfunctions which correspond to \( M \) highest eigenvalues; hence reducing the number of eigenfunctions from \( N(n_x + n_u + n_y) \) to \( M(n_x + n_u + n_y) \). Further, we stack the corresponding eigenvalues in a matrix \( V \).

4) FINDING \( \phi_m \)
At this point, we have \( M \) basis eigenfunctions corresponding to each state \( x_i \), each input \( u_k \), and each output variable \( y_l \) as \( \psi_m^{(x)} \), \( \psi_m^{(u)} \), and \( \psi_m^{(y)} \) respectively where \( m = 1, 2, . . . , M \). Considering the design of basis function for our GP prior in (15), we propose to design \( m \)-th basis function \( \phi_m \) as a tensor product of these coordinate basis functions as

\[
\phi_m = \psi_m^{(x_1)} \otimes \ldots \otimes \psi_m^{(x_n)} \otimes \psi_m^{(u_1)} \otimes \ldots \otimes \psi_m^{(u_n)} \otimes \psi_m^{(y_1)} \otimes \ldots \otimes \psi_m^{(y_n)}
\]  

(21)

where the symbol \( \otimes \) denotes the element-by-element product of vectors. Overall procedure to design basis functions as described, is also illustrated in Fig. 3.

B. MODIFIED GP-SSM
At this point, we have \( M \) basis functions (vectors) as \( \phi_m \in \mathbb{R}^N \) where \( m = 1, 2, . . . , M \). We need to incorporate them inside the available GP-SSM in (15). In order to incorporate the designed basis functions (vectors), we propose to modify the GP-SSM in (15) to the one given in (22), as shown at the bottom of this page. This modification allows us to use a vector \( \phi_m \) inside (15). In (22), \( x_i = [x_1 \ldots x_n]^T \in \mathbb{R}^N, w_m^{(i)} \in \mathbb{R}^N \) and \( q_i \in \mathbb{R}^{n_i} \) for all \( i = 1, \ldots, n_x, m = 1, \ldots, M, \) and \( j = 1, \ldots, N \). Each row of the matrix \( X_i \) in (22) contains \( N \) realizations of the same state variable which will be reduced to one estimate using systematic resampling step during the Bayesian learning, as explained in next section. This extended formulation of SSM enables to not only incorporate the designed basis functions during inference and learning process but also provides much more reliable state trajectory estimates.

C. BAYESIAN INFERENCE AND LEARNING
Given input \( u_t \) and output \( y_t \) data samples for \( t = 1, 2, . . . , T \), our goal now is to estimate unknown parameters \( \theta = [A, Q] \), along with state trajectory \( x_{1:T} \) using the model in (22).

Thus, learning process can be divided into two steps. First, we fix a set of parameters \( \theta \) and estimate the state trajectory...
using sequential Monte Carlo (SMC) based technique. In the second step, using current state trajectory estimate, we use Bayesian learning to seek full posterior distribution of unknown parameters by alternating between samples from the distributions \( p(x_{1:T}|y_{1:T}, \theta) \) and \( p(\theta|x_{1:T}, y_{1:T}) \). For that purpose, we need posterior for \( \theta \).

1) CONJUGATE PRIOR

For matrix \( A \), we use the prior as matrix-normal \((\mathcal{MN})\) distribution given as [39]

\[
A \sim \mathcal{MN}(\mathbf{0}, \mathbf{U}, \mathbf{V})
\]

(23)

where \( \mathbf{0} \) is the mean matrix which is an all zero matrix, \( \mathbf{U} \) is right covariance matrix which is set equal to \( Q \) due to conjugacy property, and \( \mathbf{V} \) is left covariance matrix having eigenvalues of corresponding eigenfunctions as its diagonal elements. Similarly, we assume inverse-Wishart \( \mathcal{IW} \) prior for matrix \( \mathbf{Q} \) which happens to be the conjugate prior [32], as

\[
\mathbf{Q} \sim \mathcal{IW}(v_o, \Lambda_o)
\]

(24)

where \( v_o \) is the degree of freedom, and \( \Lambda_o \) is a positive definite matrix. The choice of \( \mathcal{IW} \) distribution for covariance matrix estimation is very common. Combining (23) and (24), we can express the joint prior on our unknown parameters as matrix-normal-inverse-Wishart \((\mathcal{MN}\mathcal{IW})\) distribution as

\[
\theta \sim \mathcal{MN}\mathcal{IW}(0, V, v_o, \Lambda_o)
\]

(25)

For joint Gaussian likelihood, this form of prior is a conjugate prior for our Bayesian learning problem i.e. the posterior will also be in the same form as that of (25).

2) POSTERIOR OF \( \theta \)

Given the available data \( \{u_{1:T}, y_{1:T}\} \), and an estimate of state trajectory \( x_{0:T} \), the likelihood can be written as

\[
p(x_{0:T}, y_{1:T}|\theta) = \prod_{t=1}^{T} p(y_t|x_t, \theta) \prod_{t=1}^{T} p(x_t|x_{t-1}, \theta) p(x_0|\theta)
\]

(26)

\[
\propto \prod_{t=1}^{T} p(x_{t+1}|x_t, \theta)
\]

(27)

\[
= \prod_{t=1}^{T} \mathcal{N}(x_t|A\Phi_t, Q)
\]

(28)

When expressed log-likelihood form, (28) becomes

\[
\log p(x_{0:T}, y_{1:T}|\theta) \propto \sum_{t=1}^{T} \log \mathcal{N}(x_t|A\Phi_t, Q)
\]

(29)

which can further be simplified to (30) as

\[
\log p(x_{0:T}, y_{1:T}|\theta) \\
\quad \propto -\frac{T\nu}{2} \log 2\pi - \frac{T}{2} \log |Q| \\
\quad - \frac{1}{2} \text{trace} \left( Q^{-1} \left( \Psi_1 - A \Psi_2^T - \Psi_2 A^T + A \Psi_3 A^T \right) \right)
\]

(30)

with

\[
\Psi_1 = \sum_{t=1}^{T} X_t X_t^T \\
\Psi_2 = \sum_{t=1}^{T} X_t \Phi_t^T \\
\Psi_3 = \sum_{t=1}^{T} \Phi_t \Phi_t^T
\]

Given (25) and (30), the posterior of \( \theta \) can be found analytically in a closed form as [32]

\[
\begin{align*}
\log p(A, Q|x_{0:T}, y_{1:T}) &= \mathcal{MN}\mathcal{IW}(\Psi_2(\Psi_3 + V^{-1})^{-1}, \\
&\quad (\Psi_3 + V^{-1})^{-1}, \Lambda_o + \Psi_1 - \Psi_2(\Psi_3 + V^{-1})^{-1}\Psi_2^T, v_o + T)
\end{align*}
\]

(31)

3) LEARNING

We now have equation (31) for sampling \( A \) and \( Q \) given state trajectory \( x_{0:T} \). In order to estimate the state trajectory, we use particle Gibbs with ancestor sampling (PGAS) i.e. an SMC algorithm, given model parameters \( A \) and \( Q \). These two steps are actually combined in the proposed framework summarized in Algorithm 1 and is also demonstrated in Fig. 4.

| Algorithm 1 | Bayesian Learning Using PGAS |
|-------------|-----------------------------|
| **Input**   | Data: \{u_{1:T}, y_{1:T}\}  |
| Basis functions: \( \phi_m(x, u) \) \forall m |
| Prior on \( \theta \) as in (25) |
| Number of iterations: \( N_{\text{iter}} \) |
| **Output**  | Estimates of \( x_{0:T}, A \) and \( Q \) |
| **Begin**   | Get current values of \( A \) and \( Q \) |
| for \( n = 1, \ldots, N_{\text{iter}} \) do | Get estimate of \( X_t \) using PGAS as described in [40] |
| for \( t = 1, \ldots, T \) do | Sample \( A \) and \( Q \) using (31) |
| end | Obtain state trajectory \( x_{0:T} \) using systematic resampling [41] of all \( X_t \) \forall \( t \) |
| **End**     | |

IV. RESULTS AND DISCUSSION

In this section, we evaluate the proposed framework using different simulation examples and real world data. We also compare the results of the proposed framework against Svensson method [32]. In all the following examples, we separate the data into training and testing data and compare the results in terms of following two performance metrics: RMSE, and
mean negative log probability (MNLP), obtained on testing data. These metrics are defined as

$$\text{RMSE} = \sqrt{\frac{1}{N_{\text{test}}} \sum_{n=1}^{N_{\text{test}}} (f(x^*_n, u^*_n) - \hat{f}(x^*_n, u^*_n))^2}$$

$$\text{MNLP} = \frac{1}{N_{\text{test}}} \sum_{n=1}^{N_{\text{test}}} \frac{1}{\sigma^2} \left( f(x^*_n, u^*_n) - \hat{f}(x^*_n, u^*_n) \right)^2 + \log \sigma^2 + \log_2 \pi$$

where $\hat{f}(\cdot)$ is the estimated value of the true function value $f(\cdot)$ at $n$-th test data point $\{x^*_n, u^*_n\}$, and $\sigma^2$ is the variance of test outputs. In all of the results that follow, we obtained the optimal values of hyper parameters for GP using maximum-likelihood method in offline settings as described in [32], [42], for both proposed framework and reference framework. We also report the averaged RMSE, MNLP and execution time results in Table 1 which have been computed after performing the simulations 20 times. Number of iteration, $N_{\text{iter}}$, during Bayesian learning are fixed to 1000 for all the results. In all the results that follow, we have used squared-exponential kernel, given in (10), for both methods.

A. SIMULATION EXAMPLE 1: SINC

We first consider a numerical example where the actual SSM is given by

$$x_t = 10 \text{sinc} \left( \frac{x_t-1}{7} \right) + q_t \quad q_t \sim \mathcal{N}(0, Q)$$

$$y_t = x_t + r_t \quad r_t \sim \mathcal{N}(0, R)$$

The unknown function to be learnt is $f(x) = 10 \text{sinc} \left( \frac{x-1}{7} \right)$ whereas $g(\cdot)$ is assumed to be known. We set $Q = 0.1$ and $R = 4$. The number of basis functions i.e. $M$ is selected to be 20. The results are presented in Fig. 5. It is clear from the results that our proposed framework is able to identify the system model with reasonable accuracy. While comparing the performance with the reference framework, it is clear that the method in [32] has poor prediction quality near boundary points as compared to our method. This is due to the fact that the reference framework involves eigenfunctions and eigenvalues within a compact domain. We have not made such an assumption about domain of eigenfunctions so our model shows improved prediction accuracy indicating low variance near boundary points as well. The simulation was repeated 20 times and average RMSE and MNLP were computed using actual function values. The results are reported in Table 1. These averaged results indicate a significant gain in the accuracy performance of the proposed framework. Further, the variance of the model estimate is much lower as compared to the model obtained using the reference method which is clear from the posterior confidence intervals.

B. SIMULATION EXAMPLE 2: PIECEWISE

Our second example aims to illustrate the cases where stationary GP is inappropriate for functional inference. We may come across transition function which is smooth but not at certain points. There may be an outage or discontinuity. If we try to model such a discontinuous function using stationary kernels, it may results in poor function approximation. Below we illustrate the idea that with our proposed framework, it becomes possible to identify the discontinuity points and predict the function really well even using stationary kernel. For this purpose, we consider the following $f(\cdot)$ inside SSM
in (34)

\[
f(x) = \begin{cases} 
\sin(e^{-x}) & x < 1 \\
x^2 & 1 \leq x \leq 9 \\
e^{-x^2} & 9 \leq x \leq 15 \\
x & x \geq 15 
\end{cases}
\]  

(36)

The considered function is discontinuous which presents a challenge to common GP based inference frameworks as most of them assume the notion of global smoothness. Fig. 6 shows the results using our proposed framework in comparison to Svensson method for the case where \( Q = R = 0.1 \), and \( M = 20 \) for both methods. Clearly, we see the true advantage of the proposed framework in such situations due to the fact that instead of using fixed trigonometric basis for function expansion within GP framework, the set of basis functions is learned from data by solving kernel eigenvalue problem numerically. The averaged RMSE and MNLP for this example are reported in Table 1. The performance gain obtained in terms of RMSE and MNLP come at a cost of designing and solving eigenvalue problem specific to each example/dataset.

C. REAL DATA EXAMPLE 1: WATER TANK

To demonstrate the practical application of our proposed framework, we consider a data set collected from physical system consisting of two water tanks. The SSM to be identified is of the form

\[
x_1_t = f_1(x_{t-1}, u_{t-1}) + q_{1_{t-1}} \\
x_2_t = f_2(x_{t-1}, u_{t-1}) + q_{2_{t-1}} \\
y_t = x_2_t + r_t
\]  

(37) \hspace{1cm} (38) \hspace{1cm} (39)

Here we have two dimensional state space model where state 1 and 2 indicate the water flow into tanks 1 and 2 respectively. The basic setup is water flow into the tank 1 is controlled by applying the input voltage \( u \) to the pump. Water flow into the second tank is also observed as an output. The setup is shown in Fig. 7. In the considered water tank system, there may be a scenario where an overflow occurs and water from tank 1 goes partly into the tank 2 and partly into the tank 1, which is nothing but an indication of a discontinuity. At first, data was collected from the observations of the input voltage and and tank 2 water flow. This training data was used to learn the model of the system using designed algorithms. Then another data set called the test data was recorded during another hour with only input voltage fed into the tank and simulated output was obtained. The simulated output was compared with the actual measured output for both methods. The averaged results in terms of RMSE and MNLP are reported in Table 1 which indicate that the proposed model is able to predict output rather well as compared with the Svensson model.

D. REAL DATA EXAMPLE 2: ELECTRIC DRIVE

Second real world dataset that we consider for comparison is electric drive dataset as reported in [43]. The system consists of two electric motors that drive a pulley using a flexible belt. The pulley is held by a spring, resulting in a lightly damped dynamic mode. The electric drives can be individually controlled allowing the tension and the speed of the belt to be simultaneously controlled. The available data sets are short, which constitute a challenge when performing inference and identification. In learning the unknown function, we adopted the same approach as it was for Water Tank data set. The results for identifying the unknown system are summarized in Table 1 where average RMSE, MNLP, and execution time is reported. Again we observe some performance gain
TABLE 1. Averaged results for all examples.

| Function                     | Method    | Basis functions, \( M \) | RMSE   | MNLP    | Time [s] |
|------------------------------|-----------|---------------------------|--------|---------|----------|
| Simulation Example 1: Sinc   | Svensson  | 20                        | 0.6064 | -8.0154 | 110      |
|                             | Proposed  | 20                        | 0.4056 | -20.9103| 100      |
| Simulation Example 2: Piecewise | Svensson  | 20                        | 15.332 | -3.705  | 180      |
|                             | Proposed  | 20                        | 2.576  | -10.469 | 110      |
| Water Tank Data Set          | Svensson  | 5                         | 2.501  | -3.241  | 120      |
|                             | Svensson  | 10                        | 2.612  | -3.102  | 150      |
|                             | Proposed  | 5                         | 0.671  | -3.781  | 100      |
|                             | Proposed  | 10                        | 0.668  | -3.974  | 100      |
| Electric Drive Data Set      | Svensson  | 5                         | 0.5615 | -0.6271 | 115      |
|                             | Svensson  | 10                        | 0.6052 | -0.5848 | 200      |
|                             | Proposed  | 5                         | 0.3865 | -1.0346 | 100      |
|                             | Proposed  | 10                        | 0.3885 | -1.0421 | 100      |

FIGURE 9. Electric drive data: simulated output vs. real output, \( M = 5 \).

in terms of RMSE and MNLP of the proposed framework over the reference method. The gain is clearly attributed to the fact that instead of using fixed trigonometric basis for function expansion within GP framework, the set of basis functions is learned from data by solving kernel eigenvalue problem numerically. This validates the adaptive power of our proposed GP as compared to Stationary GP in [32] that smoothens out the curve.

E. COMPLEXITY COMPARISON

In order to compare the complexity of the proposed method with the reference method, we use an experimental comparison of the time consumed by both methods in simulations and experimental datasets is shown in Table 1. The machine used to run the experiments consists of Intel Core-i5-4590S CPU (3 GHz) and 8-GB memory, with Windows 10 and Matlab R2019b. It can be seen from the table that the proposed method has almost same complexity as the reference method. This is due to the fact that the basis function used in our method are designed using eigenvalue decomposition, and those used in the reference method has almost same complexity as the reference method. This is due to the fact that the basis function used in our method are designed using eigenvalue decomposition, and those used in the reference method are found by evaluation of trigonometric functions along with evaluation of spectral densities for smooth GP prior initialization. For small datasets, the complexity of both methods is almost comparable where the proposed method shows less complexity for small basis function design scenarios. However, both methods will become impractical if the number of training data is large which is the main issue with GP based frameworks.

V. CONCLUSION

In this paper, we considered the problem of nonlinear dynamical model identification in state space formulation under GP framework where we have proposed to design and use the basis functions from data itself. Using eigenfunctions as basis for state space model, we have demonstrated that the performance of the GP framework can be improved significantly, especially in cases where the underlying function is discontinuous. The performance improvement has been demonstrated using different simulations and real world datasets.

One of the main limitation of the proposed framework could be the use of eigenvalue decomposition for finding eigenfunctions which are then used as basis functions. This might pose a challenge for the application of the proposed framework to cases and scenarios where the system dimension is too large. Future work may look into the ways to expedite this step somehow so that it can also be applicable to large dimensional datasets.

REFERENCES

[1] Q. Li, F. Pan, Z. Zhao, and J. Yu, “Process modeling and monitoring with incomplete data based on robust probabilistic partial least square method,” IEEE Access, vol. 6, pp. 10160–10168, 2018.
[2] G. Lee, S. S. Srinivasa, and M. T. Mason, “GP-ILQG: Data-driven robust optimal control for uncertain nonlinear dynamical systems,” 2017, arXiv:1705.05344. [Online]. Available: http://arxiv.org/abs/1705.05344
[3] H. Karimipour and V. Dinavahi, “Robust massively parallel dynamic state estimation of power systems against cyber-attack,” IEEE Access, vol. 6, pp. 2984–2995, 2018.
[4] G. Wang, S. Yin, and O. Kaynak, “An LWPR-based data-driven fault detection approach for nonlinear process monitoring,” IEEE Trans. Ind. Informat., vol. 10, no. 4, pp. 2016–2023, Nov. 2014.
[5] Y. Hong, M. Kim, H. Lee, J. J. Park, and D. Lee, “Early fault diagnosis and classification of ball bearing using enhanced kurtogram and Gaussian mixture model,” IEEE Trans. Instrum. Meas., vol. 68, no. 12, pp. 4746–4755, Dec. 2019.
[6] S. Narasimhan and C. Jordache, Data Reconciliation and Gross Error Detection: An Intelligent Use of Process Data. Houston, TX, USA: Gulf Professional Publishing, 1999.
[7] G. Pillonetto, F. Dinuzzo, T. Chen, G. De Nicolao, and L. Ljung, “Kernel methods in system identification, machine learning and function estimation: A survey,” Automatica, vol. 50, no. 3, pp. 657–682, Mar. 2014.
[8] H. Kodamana, B. Huang, R. Ranjan, Y. Zhao, R. Tan, and N. Sammaknejad, “Approaches to robust process identification: A review and tutorial of probabilistic methods,” J. Process Control, vol. 66, pp. 68–83, Jun. 2018.
