Check for updates

International Journal of Adaptive Control and Signal Processing

WILEY

RESEARCH ARTICLE OPEN ACCESS

dynoGP: Deep Gaussian Processes for Dynamic System Identification

Alessio Benavoli[1] | Marco Forgione[2] | Dario Piga[2] | Marco Zaffalon[2]

[1]School of Computer Science and Statistics, Trinity College Dublin, Ireland | [2]SUPSI, IDSIA - Dalle Molle Institute for Artificial Intelligence, Lugano, Switzerland

Correspondence: Alessio Benavoli (alessio.benavoli@tcd.ie)

Received: 16 January 2026 | **Revised:** 16 June 2026 | **Accepted:** 24 July 2026

Keywords: deep learning | Gaussian processes | stochastic linear time-invariant dynamical systems | system identification

ABSTRACT

In this work, we present a novel approach to system identification for dynamical systems, based on a specific class of deep Gaussian processes (deepGPs). These models are constructed by interconnecting linear dynamic GPs (equivalent to stochastic linear time-invariant dynamical systems) and static GPs (to model static nonlinearities). Our approach combines the strengths of data-driven methods, such as those based on neural network architectures, with the ability to output a probability distribution. This offers a more comprehensive framework for system identification that includes uncertainty quantification. Using both simulated and real-world data, we demonstrate the effectiveness of the proposed approach.

1 | Introduction

The application of neural networks to system identification dates back to the early 1990s, with seminal works by Chen et al. [1], Wray and Green [2], and Narendra and Parthasarathy [3]. These studies proposed the use of neural networks for black-box data-driven modeling and control of nonlinear dynamic systems, laying the foundation for future research in this area. The field has gained significant momentum in the last decade, with several notable advancements (see the survey paper [4] for a comprehensive discussion on the topic). Among available studies, we mention the 1-D convolutional networks to describe nonlinear finite impulse response (NFIR) models [5, 6], and the application of recurrent neural networks (RNNs) and long short-term memory (LSTM) networks for efficiently describing long-term dynamics [7]. Novel architectures, tailored to dynamical systems, have also emerged in recent years, such as: the autoencoder-based architectures [8, 9] which map input–output sequences into state variables; deep state-space models which preserve the interpretability of the networks and leverage, when available, prior physical knowledge [10, 11]; and dynoNet [12], composed of the interconnection of linear time-invariant (LTI) blocks described by transfer functions followed by static nonlinearities (activation functions or multi-layer perceptron). Similar to dynoNet, other architectures like structured state space sequence models (S4) [13] and linear recurrent units (LRUs) [14] have been proposed, made by connection of linear dynamical blocks followed by static nonlinearities. Unlike dynoNet, the linear blocks are described in a state space form, with proper parametrisation of the matrices designed to ensure efficiency during training.

A defining characteristic of architectures that integrate linear dynamical blocks with static nonlinear blocks is their universal approximation capability. Indeed, any continuous nonlinear operator with fading memory can be approximated using a layer of linear dynamical blocks followed by a nonlinear (polynomial) static block; see for instance [15, 16]. The system identification methods mentioned above, such as dynoNet, share this universal approximation property; however, they only provide point estimates of the system's output and lack a component

Abbreviations: dynoGP, deep dynamical Gaussian Process; GP, Gaussian Process; LTI, linear time invariant.

[Correction added on 09 September 2026, after first online publication: The author Marco Zaffalon has been included in this version.]

This is an open access article under the terms of the Creative Commons Attribution-NonCommercial-NoDerivs License, which permits use and distribution in any medium, provided the original work is properly cited, the use is non-commercial and no modifications or adaptations are made.
© 2026 The Author(s). *International Journal of Adaptive Control and Signal Processing* published by John Wiley & Sons Ltd.

for characterizing uncertainty, which is fundamental for reliable decision making (particularly in control applications). In contrast to deep neural networks, probabilistic models like deep Gaussian processes (deepGPs) [17] allow for the quantification of estimation uncertainty. These models combine the advantages of Gaussian Processes with the expressive power of deep neural networks, producing a probability distribution as output that can be leveraged for reliable decision-making. However, deepGPs have not yet been developed for system identification.

The objective of this paper is to merge ideas underlying dynoNet (or equivalently, S4 and LRU) with those of deepGP, aiming to develop dynamic deepGP models, which we call in this paper *dynoGP*. These models are constructed through the interconnection of dynamic GPs (specifically, as discussed in the rest of the paper, stochastic LTI dynamical systems) and static GPs. This approach seeks to leverage the strengths of both deterministic recursive neural network architectures and probabilistic deepGP models, potentially offering a comprehensive framework for system identification that includes uncertainty quantification.

This allows us to derive a powerful novel Bayesian method for the system identification of nonlinear dynamical systems. In our experiments, we compare this approach with other Bayesian methods, including GP-NARX (see, for instance, [18]), Sparse Bayesian MLP (multi-layer perceptron), and Sparse Bayesian LSTM [19].[1]

The rest of the paper is organised as follows. In Section 2.1, we provide an overview of GPs for regression, including a discussion on the selection of inducing points in sparse GP models, as this is essential for the dynoGP model proposed in this paper. In Section 2.2, we introduce the concept of stochastic LTI dynamical systems with Brownian process noise, exploring the connections between LTI systems and GP. We review literature demonstrating that the output trajectory of such systems is a Gaussian Process. We then present initial results, showing that for LTI systems with complex diagonal state-transition matrices, the mean and covariance functions of the GP describing the output trajectories can be simplified and computed in a closed form. In Section 3, we introduce the deep dynamical GP architecture (dynoGP), constructed by stacking layers of dynamic and static GPs. We employ inducing points to approximate the posterior predictive distribution through stochastic variational inference (SVI) [20]. Section 4 presents four examples, three involving real datasets, to demonstrate the effectiveness of the proposed dynoGP architecture, alongside comparisons with other learning algorithms. Detailed mathematical derivations of the paper's results and specifics on implementing the dynoGP models are discussed in the Appendix.

2 | Background and Novel Preliminaries

In this section, we review Gaussian processes, stochastic linear dynamical systems, and their connections. We then introduce some novel preliminary results that we exploit to derive our dynamic deep GP model.

2.1 | Gaussian Processes

Gaussian processes (GPs) are non-parametric models that can be employed to define prior distributions over functions [21, 22]. Unlike parametric models, such as neural networks, GPs automatically adapt in complexity as more data are observed. They require only a small number of hyperparameters, making them suitable for training on small datasets, and they naturally quantify uncertainty in their predictions.

To define a prior over a function $f : \mathcal{X} \to \mathbb{R}$, a GP assumes that, for every n points $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{X}$, the probability distribution $p(f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n))$ is a multivariate Gaussian, with mean vector $[\mu(\mathbf{x}_1), \ldots, \mu(\mathbf{x}_n)]$ and covariance matrix $Cov(f(\mathbf{x}_i), f(\mathbf{x}_j)) = K(\mathbf{x}_i, \mathbf{x}_j)$, for $i, j = 1, \ldots, n$. The functions $\mu(\mathbf{x})$ and $K(\mathbf{x}, \mathbf{x}')$ are the mean and the kernel function of the GP, respectively.[2] The mean function $\mu_\theta(\mathbf{x})$ and kernel $K_\theta(\mathbf{x}, \mathbf{x}')$ usually depend on hyperparameters $\boldsymbol{\theta} \in \Theta$. A typical example of kernel function is the Matern 3/2 kernel on $\mathbb{R}^d$. For $\mathbf{x}, \mathbf{x}' \in \mathcal{X} = \mathbb{R}^d$, it is defined as

$$K_\theta(\mathbf{x}, \mathbf{x}') = \sigma^2\left(1 + \sqrt{3}\gamma\right)\exp\left(-\sqrt{3}\gamma\right), \tag{1}$$

where $\gamma = \sqrt{\sum_{i=1}^{d}(x_i - x_i')^2/\ell_i^2}$ and $\boldsymbol{\theta} = [\ell_1, \ldots, \ell_d, \sigma^2]$ includes the lengthscale hyperparameters $\ell_i > 0$ (one for each dimension) and the scale parameter σ^2 [22, 23]. GPs, as priors over functions, have a natural Bayesian interpretation that makes them ideal for regression problems. If we assume that observations are the sum of a true function evaluated at some inputs and Gaussian noise, that is, $y_i = f(\mathbf{x}_i) + \varepsilon_i$ with $\varepsilon_i \sim N(0, \varsigma^2)$ for $i = 1, \ldots, n$, then the posterior distribution of f can be computed analytically. We can write the likelihood more compactly as:

$$p(y_1, \ldots, y_n | f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)) = N(\mathbf{y}_n | f(X), \varsigma^2 I_n),$$

where $\mathbf{y}_n = [y_1, \ldots, y_n]^\top$, $X = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^\top$ and I_n is the identity matrix of dimension n. Then, it can be easily shown, see for instance [22], that the predictive posterior for the value of $f(\mathbf{x}^*)$ at a new test point $\mathbf{x}^* \in \mathcal{X}$ is $GP(\mu_p, k_p)$, with mean and kernel function:

$$\mu_p(\mathbf{x}^*) = K_\theta(\mathbf{x}^*, X)(K_\theta(X, X) + \varsigma^2 I_n)^{-1}\mathbf{y}_n \tag{2}$$

$$k_p(\mathbf{x}^*, \mathbf{x}^*) = K_\theta(\mathbf{x}^*, \mathbf{x}^*) - K_\theta(\mathbf{x}^*, X)(K_\theta(X, X) + \varsigma^2 I_n)^{-1}K_\theta(X, \mathbf{x}^*), \tag{3}$$

where $K_\theta(X, X)$ is a matrix whose ijth element is defined as $(K_\theta(X, X))_{ij} = K_\theta(\mathbf{x}_i, \mathbf{x}_j)$ (similar for $K_\theta(\mathbf{x}^*, X)$). The variance of the likelihood, ς^2, is also treated as a hyperparameter and included in $\boldsymbol{\theta}$. The values of the hyperparameters $\boldsymbol{\theta}$ are typically estimated by maximum A-posteriori (MAP) inference:

$$\begin{aligned}&\arg\max_{\boldsymbol{\theta}} p((\mathbf{x}_i, y_i)_{i=1}^n | \boldsymbol{\theta})p(\boldsymbol{\theta}) \\ &\quad = \arg\max_{\boldsymbol{\theta}} N(\mathbf{y}_n, K_\theta(X, X) + \varsigma^2 I_n)p(\boldsymbol{\theta}),\end{aligned} \tag{4}$$

where $p(\boldsymbol{\theta})$ is a prior over the hyperparameters, see for instance [24] for the impact of the prior on hyperparameters selection. It is worth noticing that a GP, for particular choices of the kernel function, can be viewed as a neural network with an infinite number of hidden units. This result was first shown [25] for a single-layer and then generalised [26, 27] for fully connected

10991115, 0, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/acs.70141 by Zhaw Zurcher Hochschule Fur, Wiley Online Library on [12/09/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

neural networks with L hidden layers. Taking inspiration from the multi-layer structure of deep neural networks, there is a line of work which instead considers stacking GPs, known as deepGPs [17, 28], which can give rise to a richer class of probabilistic models beyond GPs. In contrast to highly parameterised deep models, deepGPs learn a hierarchical representation non-parametrically, requiring only a few hyperparameters to optimise. However, since deepGPs are not GPs, their posterior cannot be computed exactly in regression, and various types of variational approximations are used to approximate it [17, 20, 28].

A limitation of GPs is their high computational cost, with time complexity of $\mathcal{O}(n^3)$ and memory complexity of $\mathcal{O}(n^2)$, where n represents the size of the training data. However, several established methods can scale GPs to a complexity that is linear in the number of observations n. These methods are usually known as *sparse GP* methods and typically involve the use of inducing points in conjunction with variational inference [29–36]. Since, the inducing-point method is also pivotal for inference in deepGPs, we briefly provide an overview of *sparse GPs*. A sparse GP is built by augmenting the GP with a small number $m < n$ of inducing variables such that $\upsilon_i = f(\boldsymbol{\zeta}_i)$ with $\boldsymbol{\zeta}_i \in \mathcal{X}$, $\upsilon_i \in \mathbb{R}$ for $i = 1, \ldots, m$ [33] and $f \sim GP(\mu(\mathbf{x}), K_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{x}'))$. The pairs of inducing variables $\{\boldsymbol{\zeta}_i, \upsilon_i\}_{i=1}^m$ can be thought of as virtual observations. By conditioning the GP on $\{\boldsymbol{\zeta}_i, \upsilon_i\}_{i=1}^m$, we can compute the posterior predictive distribution $p(f(X)|Z, \Upsilon)$ at the real observations $\{\mathbf{x}_i, y_i\}_{i=1}^n$, that is

$$p(f(X)|Z, \Upsilon) = N(Q\Upsilon, K_{\boldsymbol{\theta}}(X, X) - QK_{\boldsymbol{\theta}}(Z, Z)Q^\top), \tag{5}$$

where $\Upsilon \sim N(\mathbf{0}, K_{\boldsymbol{\theta}}(Z, Z))$, $Q = K_{\boldsymbol{\theta}}(X, Z)K_{\boldsymbol{\theta}}^{-1}(Z, Z)$, $Z = [\boldsymbol{\zeta}_1, \ldots, \boldsymbol{\zeta}_m]^\top$ and $\Upsilon = [\upsilon_1, \ldots, \upsilon_m]^\top$. The inference problem of sparse GP is to learn the hyperparameters $\boldsymbol{\theta}$ of the kernel and the location of the virtual observations $\{\boldsymbol{\zeta}_i\}_{i=1}^m$ by maximizing a lower bound of the true log marginal likelihood [34]. The key point of this approach is that the virtual observations are defined to be variational parameters which are selected by minimizing the Kullback–Leibler divergence between the variational distribution and the exact posterior distribution over the latent function values. This prevents overfitting, since for $m \to n$ the variational approximation simply converges to the original GP posterior. Sparse GP have a time computational complexity of $\mathcal{O}(m^2 n)$. Note that, other approaches exist for sparse GP which are not based on inducing points, see for instance [37–39].

2.2 | Stochastic Linear Dynamical Systems

A stochastic LTI dynamical system is described by:

$$d\mathbf{z}(t) = A\mathbf{z}(t)dt + B\mathbf{u}(t)dt + Ld\boldsymbol{\beta}(t), \tag{6a}$$

$$y(t) = C\mathbf{z}(t) + D\mathbf{u}(t), \tag{6b}$$

where $y(t) \in \mathbb{R}$ is the output at time t, $\mathbf{z}(t) \in \mathbb{R}^{n_s}$ is the hidden state vector, $\boldsymbol{\beta}(t) \in \mathbb{R}^{n_l}$ is Brownian motion with unit diffusion, $\mathbf{u}(t) \in \mathbb{R}^{n_u}$ are n_u deterministic input signals, and $A \in \mathbb{R}^{n_s \times n_s}$, $B \in \mathbb{R}^{n_s \times n_b}$, $L \in \mathbb{R}^{n_s \times n_l}$, $C \in \mathbb{R}^{1 \times n_z}$, and $D \in \mathbb{R}^{1 \times n_d}$ are time-invariant matrices.[3] The initial state $\mathbf{z}(t_0) \sim N(\mathbf{m}(t_0), \Sigma(t_0, t_0))$ is assumed to be independent of the Brownian motion $\boldsymbol{\beta}(t)$. As shown for instance in [40], the solution to the linear stochastic differential Equation (6a) is a GP, hereafter denoted by $\mathbf{z}(t) \sim GP(\mathbf{m}(t), \Sigma(t, t'))$. Furthermore, the mean $\mathbf{m}(t)$ and the covariance $\Sigma(t, t)$ of the state vector at time t evolve according to the deterministic ordinary differential equations:

$$\frac{d}{dt}\mathbf{m}(t) = A\mathbf{m}(t) + B\mathbf{u}(t), \tag{7a}$$

$$\frac{d}{dt}\Sigma(t, t) = A\Sigma(t, t) + \Sigma(t, t)A^\top + LL^\top, \tag{7b}$$

with initial condition $\mathbf{m}(t_0), \Sigma(t_0, t_0)$ corresponding to the mean and covariance of $\mathbf{z}(t_0)$, respectively. The solutions to the differential equations above are:

$$\mathbf{m}(t) = \Psi(t, t_0)\mathbf{m}(t_0) + \int_{t_0}^{t} \Psi(t, \tau)B\mathbf{u}(\tau)d\tau, \tag{8a}$$

$$\Sigma(t, t) = \Psi(t, t_0)\Sigma(t_0, t_0)\Psi(t, t_0)^\top + \int_{t_0}^{t} \Psi(t, \tau)LL^\top\Psi(t, \tau)^\top d\tau, \tag{8b}$$

where $\Psi(t, t')$ is the transition matrix $\Psi(t, t') = e^{A(t-t')}$, and e denotes the *matrix exponential*. Furthermore, the covariance $\Sigma(t, t')$ between state vectors at distinct time instants t, t' is given by:

$$\begin{aligned}\Sigma(t, t') &= E[(\mathbf{z}(t) - \mathbf{m}(t))(\mathbf{z}(t') - \mathbf{m}(t'))^\top] \\ &= \begin{cases} \Sigma(t, t)\Psi^\top(t', t) & \text{if } t < t', \\ \Psi(t, t')\Sigma(t', t') & \text{if } t \geq t'. \end{cases}\end{aligned} \tag{9}$$

Owing to linearity of (6b), the output $y(t)$ is also a GP $y(t) \sim GP(m_{\boldsymbol{\theta}}(t), S_{\boldsymbol{\theta}}(t, t'))$. The covariance $S_{\boldsymbol{\theta}}(t, t')$ is:

$$\begin{aligned}S_{\boldsymbol{\theta}}(t, t') &= E[(y(t) - C\mathbf{m}(t) - D\mathbf{u}(t))(y(t') - C\mathbf{m}(t') - D\mathbf{u}(t))^\top] \\ &= C\Sigma(t, t')C^\top,\end{aligned} \tag{10}$$

while the mean is $m_{\boldsymbol{\theta}}(t) = C\mathbf{m}(t) + D\mathbf{u}(t)$.[4] Note that, the hyperparameters vector $\boldsymbol{\theta}$ of the GP will include the coefficients of the matrices A, B, L, C, D as well as the variance of the noise ς^2. These parameters are estimated via MAP as described in (4). Assuming the LTI system is stable, (7b) will have a steady-state solution Σ_∞, which is the solution of the Lyapunov equation: $A\Sigma_\infty + \Sigma_\infty A^\top + LL^\top = 0$. Then, the GP covariance (10) can be simplified to:

$$S_{\boldsymbol{\theta}}(t, t') = \begin{cases} C\Sigma_\infty\Psi^\top(t', t)C^\top & \text{if } t < t', \\ C\Psi(t, t')\Sigma_\infty C^\top & \text{if } t \geq t'. \end{cases} \tag{11}$$

Remark. It is important to notice that

- For a stable LTI system and a large t, the mean of the GP $m_{\boldsymbol{\theta}}(t) = C\mathbf{m}(t) + D\mathbf{u}(t)$ corresponds to the output $y(t)$ of the deterministic part (that is, assuming $L = \mathbf{0}$) of the LTI system (6) when driven by the input $\mathbf{u}(t)$. In other words, the mean is equal to the convolution between the impulse response of the LTI system and the input $\mathbf{u}(t)$.
- Equation (7b) provides the diagonal terms of the covariance matrix (that is, the covariance function of $\mathbf{z}(t)$), while the non-diagonal terms (that is, the covariance function between $\mathbf{z}(t)$ and $\mathbf{z}(t')$ with $t \neq t'$) are provided by (9).

10991115, 0, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/acs.70141 by Zhaw Zurcher Hochschule Fur, Wiley Online Library on [12/09/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

Therefore, together these equations fully specify the kernel of the GP defined by the stochastic LTI system. To show that, as an example, we report in Appendix A a well-known connection between stochastic LTI systems and 1-dimensional Matern 3/2 kernels [41], whose covariance function $S_\theta(t,t')$ is given in (1) with $\gamma = |t-t'|/\ell$ and $\ell > 0$ being the length-scale hyperparameter.

- Having illustrated the equivalence between stochastic LTI systems and GPs, $y(t) \sim GP(m(t), S(t,t'))$, given observations $y(t_1), \ldots, y(t_n)$ at time $t_1, \ldots, t_n$, the predictive posterior over y can be computed as described in (2) and (3). This allows us to calculate the predictive distribution $p(y(t^*)|data)$ for each time t^*. This computation is equivalent to performing filtering and smoothing using recursive methods such as the Kalman filter and smoother [42]. In GPs, however, this is achieved in a non-sequential manner without computing an estimate of the state, yet the resulting predictive distribution remains the same. Also, note that the estimate of the hyperparameters via MAP estimation is equivalent to what is commonly done in Kalman filtering. This is because the marginal likelihood in (4) coincides with the marginal likelihood computed via the Kalman filter.

State-space representations for other common kernels are provided in [41, 43]. In this paper, we focus on LTI dynamical systems. However, the connection between stochastic linear dynamical system and GPs also extends to time-variant cases, as shown, for instance, in [44, 45]. For example, [45] derived the linear time-variant representation of the single-layer neural network kernel defined in [25]. It is important to note that a GP defined by a stochastic linear dynamical system is indexed by time, meaning its mean and covariance functions depend on a one-dimensional covariate, that is, time. In contrast, in Section 2.1, we considered a GP indexed by a vector $\mathbf{x} \in \mathbb{R}^d$. Such GPs cannot be represented by stochastic differential equations. However, alternative representations exist, involving stochastic partial differential equations [40, 46] or assuming space/time separability [47]. This distinction is important for the aim of this paper which is building a hierarchical composition of GPs, as we will explain in Section 3.

2.2.1 | Efficient Computation of the Mean and Covariance

We will now discuss how to efficiently compute the mean and covariance functions in $GP(m_\theta(t), S_\theta(t,t'))$. We will assume that time has been discretised, $t = k\delta$, where $k = 0, 1, 2, \ldots, n$, and $\delta > 0$ is the sampling time. Then, we can reduce the computation of the $n \times n$ elements of the covariance function $S(t,t')$ for $t, t' \in \{k\delta : \ 0, 1, 2, \ldots, n-1\}$ to the computation of n elements. Indeed, for any t, from (11) we have that

$$S(t, t+k\delta) = C\Sigma_\infty(e^{Ak\delta})^\top C^\top = C\Sigma_\infty(\bar{A}^k)^\top C^\top, \tag{12}$$

where $\bar{A} = e^{A\delta}$. Therefore, the $n \times n$ covariance matrix can be constructed by computing the n elements $C\Sigma_\infty(\bar{A}^k)^\top C^\top$ for $k = 0, \ldots, n-1$ and leveraging stationarity. This implies that $S(t,t')$ depends only on $|t-t'|$, and consequently, the matrix $S(T,T)$, with $T = [t_1, \ldots, t_n]^\top$, contains at most n distinct elements. A similar approach can be applied to the computation of the mean. By assuming a zero-order hold (ZOH) interpolation between discretised inputs, we have that

$$m(k\delta) = C\left(\bar{A}^k \mathbf{m}(0) + \sum_{i=1}^{k} \bar{A}^{i-1}\overline{B}\mathbf{u}(k\delta)\right) + D\mathbf{u}(k\delta), \tag{13}$$

where $\overline{B} = (\bar{A} - I)A^{-1}B$ is the discretised B matrix. In the following, since we will only consider stable LTI systems, we will set $\mathbf{m}(0) = \mathbf{0}$. In both (12) and (13), the bottleneck in the computation is determined by $e^{Ak\delta} = \bar{A}^k$. A way to further accelerate the construction of the mean and kernel function is by restricting the structure of the A matrix.

2.2.2 | Complex Conjugate Pairs Based Structure

In this section, we derive novel preliminary results that will be used to construct our dynamical deepGP. In particular, we introduce an additive composition of two-dimensional stable linear dynamical systems capable of approximating any (rational) transfer function from $\mathbf{u}$ to y, while simultaneously enabling efficient computation of the matrix exponential $e^{Ak\delta}$. This structure, in turn, facilitates fast evaluation of the mean and kernel functions in the Gaussian process $GP(m_\theta(t), S_\theta(t,t'))$. Such efficiency is achieved by working with a dynamical system with a diagonal state matrix A with complex conjugate eigenvalues. Consider the following system with $n_s = 2$ states and complex-conjugate eigenvalues:

$$\begin{aligned} d\mathbf{z}_c(t) &= \overbrace{\begin{bmatrix} \lambda & 0 \\ 0 & \lambda^\dagger \end{bmatrix}}^{A_c} \mathbf{z}_c(t)dt + \overbrace{\begin{bmatrix} B \\ B^\dagger \end{bmatrix}}^{B_c} \mathbf{u}(t)dt + \overbrace{\begin{bmatrix} L \\ L^\dagger \end{bmatrix}}^{L_c} d\boldsymbol{\beta}(t), \\ y_c(t) &= \overbrace{\begin{bmatrix} c & c^\dagger \end{bmatrix}}^{C_c} \mathbf{z}_c(t) + D\mathbf{u}(t), \end{aligned} \tag{14}$$

with † denoting complex-conjugation, $A_c \in \mathbb{C}^{2\times 2}$, $B_c \in \mathbb{C}^{2\times n_u}$, $L_c \in \mathbb{C}^{2\times n_l}$, $\mathbf{z}_c(0) = [\chi, \chi^\dagger]^\top$ with $\chi \in \mathbb{C}$ and $\mathbf{u}(t), \boldsymbol{\beta}(t)$ are real-valued. We further assume that the real part of λ is negative for stability. We can transform the system (14) into the real form:

$$\begin{aligned} d\mathbf{z}_r(t) &= \overbrace{\begin{bmatrix} \Re[\lambda] & \Im[\lambda] \\ -\Im[\lambda] & \Re[\lambda] \end{bmatrix}}^{A_r} \mathbf{z}_r(t)dt + \overbrace{\begin{bmatrix} \sqrt{2}\Re[B] \\ -\sqrt{2}\Im[B] \end{bmatrix}}^{B_r} \mathbf{u}(t)dt \\ &\quad + \overbrace{\begin{bmatrix} \sqrt{2}\Re[L] \\ -\sqrt{2}\Im[L] \end{bmatrix}}^{L_r} d\boldsymbol{\beta}(t) \\ y_r(t) &= \overbrace{\begin{bmatrix} \sqrt{2}\Re[c] & -\sqrt{2}\Im[c] \end{bmatrix}}^{C_r} \mathbf{z}_r(t) + D\mathbf{u}(t), \end{aligned} \tag{15}$$

where $\mathbf{z}_r(t) \in \mathbb{R}^2$, with the transformation $\mathbf{z}_r = J\mathbf{z}_c$ with

$$J = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ \imath & -\imath \end{bmatrix}, \quad J^{-1} = J^H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & -\imath \\ 1 & \imath \end{bmatrix}, \tag{16}$$

10991115, 0, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/acs.70141 by Zhaw Zurcher Hochschule Fur, Wiley Online Library on [12/09/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

with $\imath$ being the complex unit and the superscript ^{H} denotes the Hermitian transpose. Note that, $\Re[x]$ and $\Im[x]$ denote the real part and, respectively, imaginary part of x. By exploiting $A_c = J^H A_r J$ and $L_c = J^H L_r$, it is easy to prove that the solutions of the Lyapunov equations for the two systems:

$$A_c S_\infty^{(c)} + S_\infty^{(c)} A_c^H + L_c L_c^H = 0, \tag{17}$$

$$A_r S_\infty^{(r)} + S_\infty^{(r)} A_r^H + L_r L_r^H = 0, \tag{18}$$

are related by $S_\infty^{(r)} = J S_\infty^{(c)} J^H$. Therefore, we have that

$$S_\theta(t,t') = \begin{cases} C_r S_\infty^{(r)} \Psi_r^H(t',t)(C^{(r)})^H \\ \quad = C_c S_\infty^{(c)} \Psi_c^H(t',t)(C^{(c)})^H & \text{if } t < t' \\ C_r \Psi_r(t,t') S_\infty^{(r)}(C^{(r)})^H \\ \quad = C_c \Psi_c(t,t') S_\infty^{(c)}(C^{(c)})^H & \text{if } t \geq t' \end{cases} \tag{19}$$

where $\Psi_c(t,t') = e^{A_c(t-t')}$ and $\Psi_r(t,t') = e^{A_r(t-t')}$. Note that, since A_c is diagonal, the matrix exponential $\Psi_c(t,t') = e^{A_c(t-t')}$ can be computed efficiently and also $\Psi_r(t,t')$ using $\Psi_r(t,t') = J\Psi_c(t,t')J^H$. This shows that, for the system (15), we can efficiently compute the elements of the kernel $S_\theta(t,t')$ by working in the complex domain, where the corresponding diagonal dynamic matrix A_c enables efficient computation of the matrix exponential (and also of Σ_∞ as shown in Lemma 1 in Appendix B). Indeed, it is straightforward to derive the following results.

Proposition 1. *The kernel (19) is equal to*:

$$S_\theta(t,t') = -e^{\lambda^\dagger \tau}\left(\frac{cLL^H c^\dagger}{\lambda + \lambda^\dagger} + \frac{c^\dagger L^\dagger L^H c^\dagger}{2\lambda^\dagger}\right) - e^{\lambda\tau}\left(\frac{cLL^\top c}{2\lambda} + \frac{c^\dagger L^\dagger L^\top c}{\lambda + \lambda^\dagger}\right), \tag{20}$$

with $\tau = |t' - t|$ and where L, λ are the parameters in (14).

The proof of this and next propositions can be found in Appendix B. For the computation of the mean function $m_\theta(t)$, it is easy to show that the following one-dimensional deterministic dynamical system has the same mean function as the one in (14).

Proposition 2. *The systems in (14) and (21) have the same mean function for $y(t)$.*

$$dx_c(t) = \overbrace{\begin{bmatrix}\lambda\end{bmatrix}}^{A_c} x_c(t)dt + \overbrace{\begin{bmatrix}B\end{bmatrix}}^{B_c} \mathbf{u}(t)dt$$

$$y(t) = 2\Re\left(\overbrace{\begin{bmatrix}c\end{bmatrix}}^{C_c} x_c(t)\right) + D\mathbf{u}(t). \tag{21}$$

The mean function is:

$$m_\theta(k\delta) = 2\Re\left(ce^{\lambda k\delta}\tilde{m}(0) + \sum_{i=1}^{k}\frac{c}{\lambda}e^{\lambda(i-1)\delta}(e^{\lambda\delta} - 1)B\mathbf{u}(i\delta)\right) + D\mathbf{u}(k\delta), \tag{22}$$

where $\tilde{m}(0)$ denotes the first component of $\mathbf{m}(0)$.

In (14), we defined a dynamical system with a two-dimensional state. We can build a n_s-dimensional system using an additive composition of $n_s/2$ two-dimensional systems. For instance, for $n_s = 4$, this results into the following 4-dimensional system:

$$d\mathbf{z}_c(t) = \overbrace{\begin{bmatrix}\lambda_1 & 0 & 0 & 0\\ 0 & \lambda_1^\dagger & 0 & 0\\ 0 & 0 & \lambda_2 & 0\\ 0 & 0 & 0 & \lambda_2^\dagger\end{bmatrix}}^{A_c}\mathbf{z}_c(t)dt + \overbrace{\begin{bmatrix}B_1\\ B_1^\dagger\\ B_2\\ B_2^\dagger\end{bmatrix}}^{B_c}\mathbf{u}(t)dt + \overbrace{\begin{bmatrix}L_1 & 0\\ L_1^\dagger & 0\\ 0 & L_2\\ 0 & L_2^\dagger\end{bmatrix}}^{L_c} d\boldsymbol{\beta}(t),\ y(t) = \overbrace{\begin{bmatrix}c_1 & c_1^\dagger & c_2 & c_2^\dagger\end{bmatrix}}^{C_c}\mathbf{z}_c(t) + D\mathbf{u}(t). \tag{23}$$

Proposition 3. *Consider the additive composition of $n_s/2$ two-dimensional systems of type (14), then the output $y(t)$ is distributed as $GP(m_\theta(t), S_\theta(t,t'))$, with mean and covariance function*:

$$m_\theta(k\delta) = \sum_{j=1}^{n_s/2} 2\Re\left(c_j e^{\lambda_j k\delta}\tilde{m}_j(0) + \sum_{i=1}^{k}\frac{c_j e^{\lambda_j(i-1)\delta}(e^{\lambda_j\delta} - 1)}{\lambda_j}B_j\mathbf{u}(i\delta)\right) + D\mathbf{u}(k\delta), \tag{24}$$

and

$$S_\theta(t,t') = \sum_{i=1}^{n_s/2}\left(-e^{\lambda_i^\dagger\tau}\left(\frac{c_i L_i L_i^H c_i^\dagger}{\lambda_i + \lambda_i^\dagger} + \frac{c_i^\dagger L_i^\dagger L_i^H c_i^\dagger}{2\lambda_i^\dagger}\right) - e^{\lambda_i\tau}\left(\frac{c_i L_i L_i^\top c_i}{2\lambda_i} + \frac{c_i^\dagger L_i^\dagger L_i^\top c_i}{\lambda_i + \lambda_i^\dagger}\right)\right) \tag{25}$$

with $\tau = |t' - t|$.

The additive composition of $n_s/2$ two-dimensional subsystems of type (14) yields a matrix A_c whose eigenvalues are (simple) complex conjugate pairs. This assumption does not limit expressiveness or generality, as proven in the following proposition.

Proposition 4. *The additive composition of $n_s/2$ two-dimensional subsystems of type (14) can approximate any impulse response from $\mathbf{u}$ to y of a stable, finite-dimensional LTI system.*

Hereafter, we present an example that exploits the proof of Proposition 4 to derive this approximation.

Example 1. Consider the transfer function

$$G(s) = \frac{s + 0.5}{(s+1)(s^2 + 0.4s + 9.04)^2},$$

which includes a complex-poles with multiplicity two and one real pole. The following additive composition of three two-dimensional subsystems of type (14):

10991115, 0, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/acs.70141 by Zhaw Zurcher Hochschule Fur, Wiley Online Library on [12/09/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

$$A_c = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.2-3\imath & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.2+3\imath & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.2-3\imath & 0 \\ 0 & 0 & 0 & 0 & 0 & -0.2+3\imath \end{bmatrix}$$

$$B_c = \begin{bmatrix} 0 \\ 0 \\ 0.016+745.1\imath \\ 0.016-745.1\imath \\ 0.577 \\ 0.577 \end{bmatrix}$$

$$C_c = \begin{bmatrix} -0.612 & -269.6 & 0.134+0.252\imath & 0.134-0.252\imath \\ & & 326.2-173.9\imath & 326.2+173.9\imath \end{bmatrix}$$

$$D_c = 0,$$

has (approximatively) the same transfer function. This is not a minimal state-space representation, since we use a zero to cancel one of the poles at −1. However, this non-minimality is compensated by the flexibility provided by the additive composition of $n_s/2$ two-dimensional subsystems of type (14), which does not require us to decide in advance whether to include real poles or complex-conjugate ones.

Proposition 4 proves the generality and expressiveness of the mean function of the GP in (24). By moving to the covariance function in (25), we will next establish its universality by showing that it can represent a Cauchy spectral mixture kernel.

Proposition 5. *Consider the covariance function in (25). For each $i = 1, \ldots, n_s/2$, assume that $L_i \in \mathbb{C}$, set $c_i = c_i^\dagger = c_i$ and*

$$\Re(L_i) = \frac{\Im(L_i)\Re(\lambda_i) + \sqrt{\Im(\lambda_i)^2\Im(L_i)^2 + \Im(L_i)^2\Re(\lambda_i)^2}}{\Im(\lambda_i)},$$

$$\Im(L_i) = \sqrt{-\frac{1}{2}\frac{\Im(\lambda_i)^2\Re(\lambda_i)}{\Im(\lambda_i)^2 + 2\Re(\lambda_i)^2 + \Re(\lambda_i)\sqrt{\Im(\lambda_i)^2 + \Re(\lambda_i)^2}}},$$

with $\Re(\lambda_i) < 0$ for stability. Then, the covariance (25) reduces to

$$S_\theta(t,t') = \sum_{i=1}^{n_s/2} c_i^2 e^{\Re(\lambda_i)\tau}\cos(\Im(\lambda_i)\tau), \tag{26}$$

with $\tau = |t'-t|$, which is known as the Cauchy spectral mixture kernel.

We recall that a Laplacian kernel is defined as $K(t,t') = c^2 e^{-\beta|t-t'|}$, where $\beta > 0$ is the lengthscale parameter and c^2 is the scaling parameter. Therefore, the Cauchy spectral mixture[5] kernel above is the product of Laplacian kernels and $\cos(\Im(\lambda_i)(t-t'))$ terms. This kernel has been proposed in [48] for linear system identification. In [49], it is shown that this kernel can approximate any stationary covariance function and, therefore, this proves the generality and expressiveness of the covariance function (25). Note that, the covariance function (25) has additional degrees of freedom compared to the kernel (26), including $\sin(\Im(\lambda_i)\tau)$ terms, which, as noted in [50] for random Fourier features, allow higher expressivity even for a small number of components $n_s/2$.

Having justified the choice of the mean (24) and covariance function (25), we will now present an illustrative example demonstrating the use of a GP (with this mean and covariance function) for linear system identification.

Example 2. We considered a two-dimensional input $\mathbf{u}(t) = [prbs(t,8), prbs(t,8)]$ consisting of two different random binary sequences in $\{-1,1\}$ with an hold length equal to eight steps. We then simulated a signal $y(t)$ using an LTI system similar to (23), but with a 10-dimensional state (the additive composition of five two-dimensional systems of type (14)). The parameters λ_i, $L_i, c_i \in \mathbb{C}$, and $B_i \in \mathbb{C}^2$ for $i = 1, \ldots, 5$ were randomly generated. The standard deviation ς of the measurement noise was set to 10% of the standard deviation of the LTI system's output. A sampling time of $\delta = 0.01$ was used, and data were generated over the interval $t = 0$ to $t = 25$. For system identification, we used the data for $t \leq 15$, shown in Figure 1 top. The remaining data for $t > 15$ were generated without process noise and used for testing the model. We assumed $y \sim GP(m_\theta(t), S_\theta(t,t'))$, where the mean and covariance functions are defined in (24) and (25), and $n_s = 20$. The vector $\boldsymbol{\theta}$ includes all the model's hyperparameters, namely: $\lambda_i, L_i, c_i \in \mathbb{C}$, and $B_i \in \mathbb{C}^2$ for $i = 1, \ldots, 10$, and ς. For generality, we assumed a system dimension of 20, although the true system used to generate the data had a dimension of 10.

We estimated the hyperparameters $\boldsymbol{\theta}$ using MAP, as described in (4), and computed the predictive posterior distribution, which is a GP. The posterior mean and covariance functions were calculated as in (2) and (3). Figure 1 bottom displays the predictive posterior and the 99.7% credible region for $5 \leq t \leq 25$. The figure includes predictions for the training data interval $5 \leq t \leq 15$ and predictions for the unseen data interval $15 \leq t \leq 25$. The model maintains good accuracy, as evidenced by its comparison with the true test data. At the beginning of the prediction task ($t = 5$), the prediction error is relatively large due to the transient phase. Since the GP model does not directly estimate the state, predictions assume an initial state of zero, requiring a few time steps for the model to converge.

Remark. LTI deterministic state-space models, such as HIPPO [13, 51, 52], have recently gained popularity in deep learning as alternatives to self-attention in transformers. These works primarily focus on parametrizing the state-transition matrix with structures that enable fast computation of the mean of the deterministic LTI system. This focus has led to the adoption of diagonal state transition matrices or diagonal plus low-rank structures. For the dynamic layer in dynoGP (discussed in the next section), we have exclusively considered diagonal matrices. These not only yield expressive state-space models but also facilitate efficient computation of both the mean and the covariance function of the GP, the latter requiring the solution of the Lyapunov equation. As we have shown these quantities can be computed in closed form in the diagonal case.

3 | Deep Dynamical Gaussian Processes

In this section, we introduce dynoGPs. First, we present the following notation. We refer to the GPs introduced in

10991115, 0, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/acs.70141 by Zhaw Zurcher Hochschule Fur, Wiley Online Library on [12/09/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

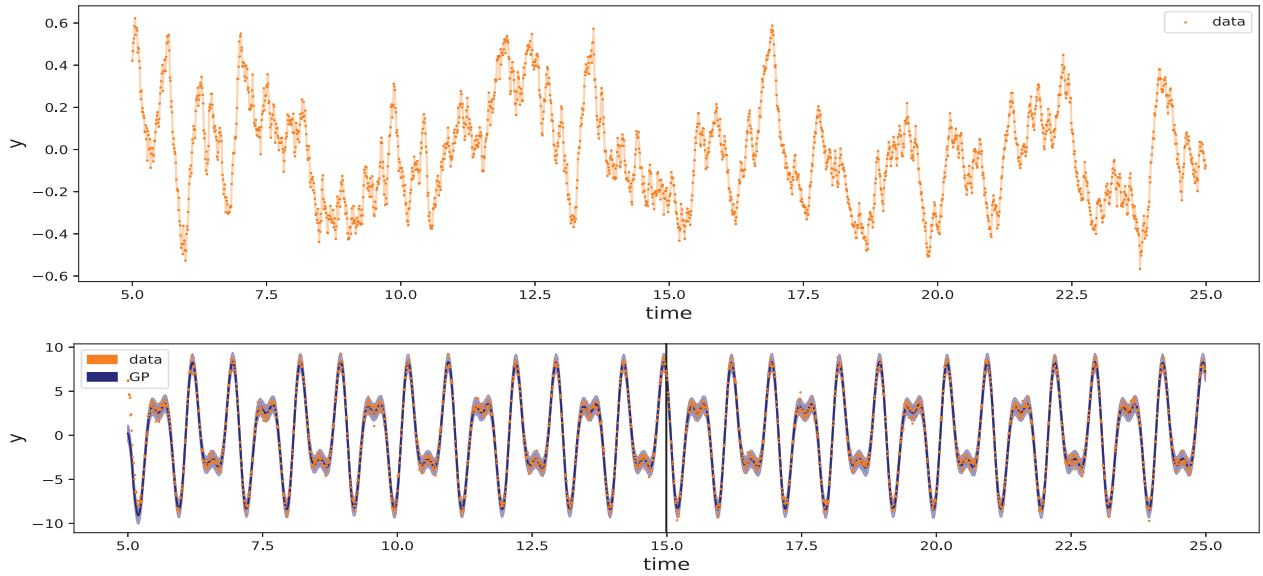


FIGURE 1 | Example: system identification of an LTI system using Gaussian Processes. The top plot shows the training data, while the bottom plot reports the predictions along with the 99.7% credible interval in blue. The values to the left of the vertical bar represent the last 1000 values of the training set, while those to the right correspond to the test set.

Section 2.1—where the covariate is a multidimensional vector $\mathbf{x} \in \mathbb{R}^d$ that does not include time—as *static GPs*, denoted as $GP(\mu_\theta, K_\theta)$. We instead refer to the GPs introduced in Section 2.2, where the covariate is time, as *dynamic GPs*, denoted as $GP(m_\theta, S_\theta)$, which can equivalently be rewritten as stochastic linear dynamical systems. A dynoGP is constructed by stacking layers of static and dynamic GPs in cascade. In this section, as an illustrative example, we will focus on the Wiener model. The general formulation is derived in Proposition 6. A Wiener model is defined as the cascade connection of an LTI dynamical model g followed by a static nonlinearity f. Inother words, a MISO (multiple input, single output) Wiener model is described by the following equation

$$y(t) = f(g(t)) + \epsilon_t, \tag{27}$$

where $\epsilon_t \sim N(0, \varsigma^2)$ is the measurement noise and $g(t)$ depends on the input signal $\mathbf{u}(t)$. A dynoGP model for (27) assumes that

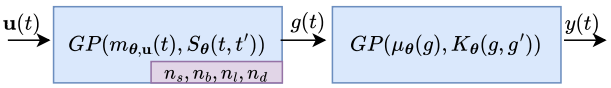
$$g \sim GP(m_{\theta,\mathbf{u}}(t), S_\theta(t,t')), \tag{28}$$

$$f \sim GP(\mu_\theta(g), K_\theta(g,g')), \tag{29}$$

where $\boldsymbol{\theta}$ is the vector of all hyperparameters including ς^2. The GP in (28) describes the distribution of trajectories $g(t)$ of an LTI dynamical system with mean function $m_{\theta,\mathbf{u}}(t)$ and covariance function $S_\theta(t,t')$. Note that, we have included the input signal $\mathbf{u}$ as a subscript to highlight the fact that the mean of the GP depends on the signal $\mathbf{u}$ from time 0 to t.

For each generated trajectory $g(t)$, $y(t)$ is assumed to be obtained by first sampling a static nonlinear function $f(g)$ from the GP in (29) and then adding i.i.d. Gaussian noise ϵ_t.

We introduce the following pictorial diagrams to define the architecture of a dynoGP. The diagram below represents a Wiener architecture. The first block corresponds to a dynamic GP, while the second block represents a static GP. The signature n_s, n_b, n_l, n_d in the dynamic block specifies the number of columns for the matrices A, B, L, D, respectively. A value of zero for any of these indicates that the corresponding matrix is zero.[6]



Given a set of (i) observations $y(t)$; (ii) values of the inputs $\mathbf{u}(t)$; at time $t = k\delta$ for $k = 1, \ldots, n$, the problem of inference in dynoGP consists of computing the posterior $p(f(g)|\{y(i\delta), \mathbf{u}(i\delta)\}_{i=1}^n)$. Given the model (27, 28, 29), we first start defining the joint distribution:

$$\begin{aligned} &p(y(\delta), \ldots, y(n\delta), f(g(\delta)), \ldots, f(g(n\delta)), g(\delta), \ldots, \\ &\quad g(n\delta)|\mathbf{u}(\delta), \ldots, \mathbf{u}(n\delta)) \\ &= p(y(\delta), \ldots, y(n\delta)|f(g(\delta)), \ldots, f(g(n\delta))) \\ &\quad p(f(g(\delta)), \ldots, f(g(n\delta))|g(\delta), \ldots, g(n\delta)) \\ &\quad p(g(\delta), \ldots, g(n\delta)|\mathbf{u}(\delta), \ldots, \mathbf{u}(n\delta)), \end{aligned} \tag{30}$$

which captures the generative process of the model, describing how the input $\mathbf{u}$ produce y. The first term in (30) is the likelihood:

$$\begin{aligned} &p(y(\delta), \ldots, y(n\delta)|f(g(\delta)), \ldots, f(g(n\delta))) \\ &= N\left(\begin{bmatrix} y(\delta) \\ \vdots \\ y(n\delta) \end{bmatrix}; \begin{bmatrix} f(g(\delta)) \\ \vdots \\ f(g(n\delta)) \end{bmatrix}, \varsigma^2 I_n\right), \end{aligned} \tag{31}$$

with I_n being the identity matrix of dimension n. The second term is the static GP (outer layer), and therefore, the conditional probability is a multivariate normal distribution:

10991115, 0, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/acs.70141 by Zhaw Zurcher Hochschule Fur, Wiley Online Library on [12/09/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

$$p(f(g(\delta)), \ldots, f(g(n\delta))|g(\delta), \ldots, g(n\delta)) = N\left(\begin{bmatrix} f(g(\delta)) \\ \vdots \\ f(g(n\delta)) \end{bmatrix}; \begin{bmatrix} \mu_\theta(g(\delta)) \\ \vdots \\ \mu_\theta(g(n\delta)) \end{bmatrix}, K(\mathbf{g}, \mathbf{g})\right), \tag{32}$$

where $\mathbf{g} = [g(\delta), \ldots, g(n\delta)]^\top$. Finally, the last term is the dynamic GP (inner layer), leading to the multivariate normal distribution:

$$p(g(\delta), \ldots, g(n\delta)|\mathbf{u}(\delta), \ldots, \mathbf{u}(n\delta)) = N\left(\begin{bmatrix} g(\delta) \\ \vdots \\ g(n\delta) \end{bmatrix}; \begin{bmatrix} m_{\theta,\mathbf{u}}(\delta) \\ \vdots \\ m_{\theta,\mathbf{u}}(n\delta) \end{bmatrix}, S_\theta(T, T)\right). \tag{33}$$

with $T = [t_1, \ldots, t_n]^\top$. Our goal is to compute the posterior distribution $p(f(g(\delta)), \ldots, f\left(g(n\delta)\right)|\{y(i\delta), \mathbf{u}(i\delta)\}_{i=1}^n)$ and the posterior predictive for $y(t)$. No analytical solution exists for either posterior. We then apply SVI [20, 30, 34] to approximate the posterior. First, we introduce inducing points (virtual observations) for each GP, as discussed in Section 2.1. These inducing points enable us to compute the posterior predictive distribution for each GP, which remains a GP when conditioned on the inducing points. We then define the joint probability density augmented with inducing points for each GP block:

$$p(\mathbf{y}, \mathbf{f}, \Upsilon_f, \mathbf{g}, \Upsilon_g|Z_f, Z_g, \mathbf{u}) = \underbrace{p(\mathbf{y}|\mathbf{f})}_{\text{likelihood}} \underbrace{p(\mathbf{f}|\Upsilon_f, Z_f, \mathbf{g})p(\Upsilon_f|Z_f)}_{\text{GP for static block}} \underbrace{p(\mathbf{g}|\Upsilon_g, Z_g, \mathbf{u})p(\Upsilon_g|Z_g)}_{\text{GP for dynamic block}} \tag{34}$$

where $\mathbf{y} = [y(\delta), \ldots, y(n\delta)]^\top$, Z_f, Υ_f are the vector of inducing points for the static GP on the function $f, \mathbf{f} = [f(g(\delta)), \ldots, f(g(n\delta))]^\top, Z_g, \Upsilon_g$ are the vector of inducing points for the dynamic GP on the function $g, \mathbf{g} = [g(\delta), \ldots, g(n\delta)]^\top$, and $\mathbf{u} = [\mathbf{u}(\delta), \ldots, \mathbf{u}(n\delta))]^\top$. We have already defined the likelihood term in (31). The second and third terms are

$$\begin{aligned} p(\mathbf{f}|\Upsilon_f, Z_f, \mathbf{g}) &= N(\mathbf{f}; Q_f\Upsilon_f, K_\theta(\mathbf{g}, \mathbf{g}) - Q_f K_\theta(Z_f, Z_f)Q_f^\top), \\ p(\Upsilon_f|Z_f) &= N(\Upsilon_f; \mathbf{0}, K_\theta(Z_f, Z_f)), \end{aligned} \tag{35}$$

with $Q_f = K_\theta(\mathbf{g}, Z_f)K_\theta^{-1}(Z_f, Z_f)$. The last two terms are

$$\begin{aligned} p(\mathbf{g}|\Upsilon_g, Z_g) &= N(\mathbf{g}; Q_g\Upsilon_g, S_\theta(T, T) - Q_g S_\theta(Z_g, Z_g)Q_g^\top), \\ p(\Upsilon_g|Z_g, \mathbf{u}) &= N(\Upsilon_g; m_{\theta,\mathbf{u}}, S_\theta(Z_g, Z_g)), \end{aligned} \tag{36}$$

with $Q_g = S_\theta(T, Z_g)S_\theta^{-1}(Z_g, Z_g)$.

We use the following variational distribution to approximate the posterior:

$$q(\mathbf{f}, \Upsilon_f, \mathbf{g}, \Upsilon_g|Z_f, Z_g, \mathbf{u}) = p(\mathbf{f}|\Upsilon_f, Z_f, \mathbf{g})q(\Upsilon_f) \times p(\mathbf{g}|\Upsilon_g, Z_g, \mathbf{u})q(\Upsilon_g) \tag{37}$$

where the two variational distributions $q(\Upsilon_f), q(\Upsilon_g)$ are assumed to be Gaussian and their mean and covariance matrix being the variational parameters. We can then derive a lower bound for the evidence as follows:

$$\begin{aligned} &\log p(\mathbf{y}) \\ &\geq \int q(\mathbf{f}, \Upsilon_f, \mathbf{g}, \Upsilon_g|Z_f, Z_g, \mathbf{u}) \\ &\quad \times \log \frac{p(\mathbf{y}, \mathbf{f}, \Upsilon_f, \mathbf{g}, \Upsilon_g|Z_f, Z_g, \mathbf{u})}{q(\mathbf{f}, \Upsilon_f, \mathbf{g}, \Upsilon_g|Z_f, Z_g, \mathbf{u})} d\mathbf{f}d\Upsilon_f d\mathbf{g}d\Upsilon_g \\ &= \int p(\mathbf{f}|\Upsilon_f, Z_f, \mathbf{g})q(\Upsilon_f)p(\mathbf{g}|\Upsilon_g, Z_g, \mathbf{u})q(\Upsilon_g) \\ &\quad \times \log \frac{p(\mathbf{y}|\mathbf{f})p(\Upsilon_f|Z_f)p(\Upsilon_g|Z_g)}{q(\Upsilon_f)q(\Upsilon_g)} d\mathbf{f}d\Upsilon_f d\mathbf{g}d\Upsilon_g \\ &= \int p(\mathbf{f}|\Upsilon_f, Z_f, \mathbf{g})q(\Upsilon_f)p(\mathbf{g}|\Upsilon_g, Z_g, \mathbf{u})q(\Upsilon_g) \\ &\quad \times \log(p(\mathbf{y}|\mathbf{f}))d\mathbf{f}d\Upsilon_f d\mathbf{g}d\Upsilon_g - KL[q(\Upsilon_f)||p(\Upsilon_f|Z_f)] \\ &\quad - KL[q(\Upsilon_g)||p(\Upsilon_g|Z_g, \mathbf{u})] \end{aligned} \tag{38}$$

where KL denotes the Kullback–Leibler divergence. The integral can be efficiently approximated using Monte Carlo sampling, as all the densities are Gaussian and they can be further simplified. Furthermore, because the bound factorises over the data, scalability can be achieved through the use of mini-batches. The hyperparameters θ, the inducing points locations Z_f, Z_g, and the variational parameters (that is, the mean and covariance of the Gaussian variational distributions $q(\Upsilon_f), q(\Upsilon_g)$) are estimated by maximizing the sum of this evidence lower bound and the logarithm of the prior over the hyperparameters θ, as in MAP inference.

Then, posterior predictive inferences can be computed from the variational distribution in (37) by replacing the time vector T with the prediction time vector T^* and considering the input $\mathbf{u}$ at the times in T^*.[7] Note that, in (37), the distributions $p(\mathbf{f}|\Upsilon_f, Z_f, \mathbf{g})$ and $p(\mathbf{g}|\Upsilon_g, Z_g, \mathbf{u})$ are both Gaussian, but the former depends on $\mathbf{g}$ in a nonlinear way. Therefore, to compute predictions for $\mathbf{f}$, we first sample $\mathbf{g}$ from $p(\mathbf{g}|\Upsilon_g, Z_g, \mathbf{u})$ and then compute the corresponding Gaussian distribution $p(\mathbf{f}|\Upsilon_f, Z_f, \mathbf{g})$ for each sample of $\mathbf{g}$. As a result, the posterior at the last layer is approximated by a Gaussian mixture, where each component has weight equal to one divided by the number of samples.

Example 3. For illustration, we again considered the LTI system in Example 2. However, in this case, we introduced a nonlinearity to the output before measuring it. Specifically, we generated the observations as $y'(t) = (y(t))^+ + \epsilon_t$, where $(\cdot)^+$ represents the positive part. Part of the training data ($5 < t < 15$) are shown in Figure 2 top, where the effect of nonlinearity can be easily seen. The remaining data for $t > 15$ were generated without process noise and used for testing the model. We used a dynoGP model to identify this system. We first considered the same model as in Example 2, a dynamic GP (that is, an LTI system). Figure 2 center shows the predictive posterior mean and 99.7% credible region for the last part of the training data and for the test data ($t > 15$). It can be observed that the model struggles to handle the nonlinearity, resulting in larger uncertainty for both the training and test data, and clearly incorrect predictions for the test data. Figure 2 bottom shows a dynoGP with a Wiener architecture, which incorporates the same dynamic GP as before, followed by a static GP.

10991115, 0, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/acs.70141 by Zhaw Zurcher Hochschule Fur, Wiley Online Library on [12/09/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License



FIGURE 2 | Example: system identification with dynoGP. Top: training data (output signal). Center: dynoGP with only a dynamic layer; values on the left of the vertical bar represent the last 1000 values of the training set, while those on the right correspond to the test set. Bottom: dynoGP with Wiener architecture. The last two plots show the posterior mean (blue line) and the 99.7% credible interval (lighter blue-region).

This approach significantly improves the model's accuracy as expected. Note that, we assumed a system dimension of $n_s = 20$ in the dynamic GP, although the true system used to generate the data had a dimension of 10. Oversizing the dimension of the state (and so the number of parameters) in the dynamic GP component does not lead to overfitting. This holds because dynoGP is a Bayesian model and, as such, its prior has a regularizing effect. We demonstrate this in Appendix C, where we provide the testing error for dynoGP using different dimensions $n_s \in \{2, 6, 10, 20\}$ of the state of the dynamic GP component. Finally, to assess the capacity of dynoGP to predict the response of the nonlinear system to input signals different from those used in training, we used the dynoGP-Wiener model (trained with the data in Figure 2 top) to predict the output of the nonlinear system with respect to the following input signals: (Case 1) $\mathbf{u}(t) = [\sin(3\pi t), \sin(5\pi t)]$; (Case 2) $\mathbf{u}(t) = [\text{sawtooth}(3\pi t), \text{sawtooth}(5\pi t)]$. In both cases, these signals are continuous and have range in $[-1, 1]$, while the signals used to train dynoGP were two independent pseudo-random binary sequences. Figure 3 compares the dynoGP predictions with the actual response of the nonlinear system for these input signals. The close agreement indicates that dynoGP can accurately generalise and predict the system output for novel input signals not used during training.

Remark.

- For dynoGP, in this paper, we always use a Matern 3/2 kernel [22] in the static layer. In [15], the universal approximation capability of the Wiener architecture is proven using a polynomial nonlinearity in the static layer. This would correspond to the use of a polynomial kernel in the static GP layer. However, the polynomial kernel can also be replaced with other universal kernels that exhibit better extrapolation properties (such as the Matern 3/2 kernel), that is, kernels whose induced function spaces generalise more reliably outside the range of the training data, rather than being dominated by rapidly growing polynomial terms.
- deepGPs often suffer from degenerate covariances [17], where each layer in the composition reduces the rank of the system. To address this issue, [20] proposed using an identity mean function in all intermediate GP layers. Our model does not appear to have this issue. This is because the dynamic GP

10991115, 0, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/acs.70141 by Zhaw Zurcher Hochschule Fur, Wiley Online Library on [12/09/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

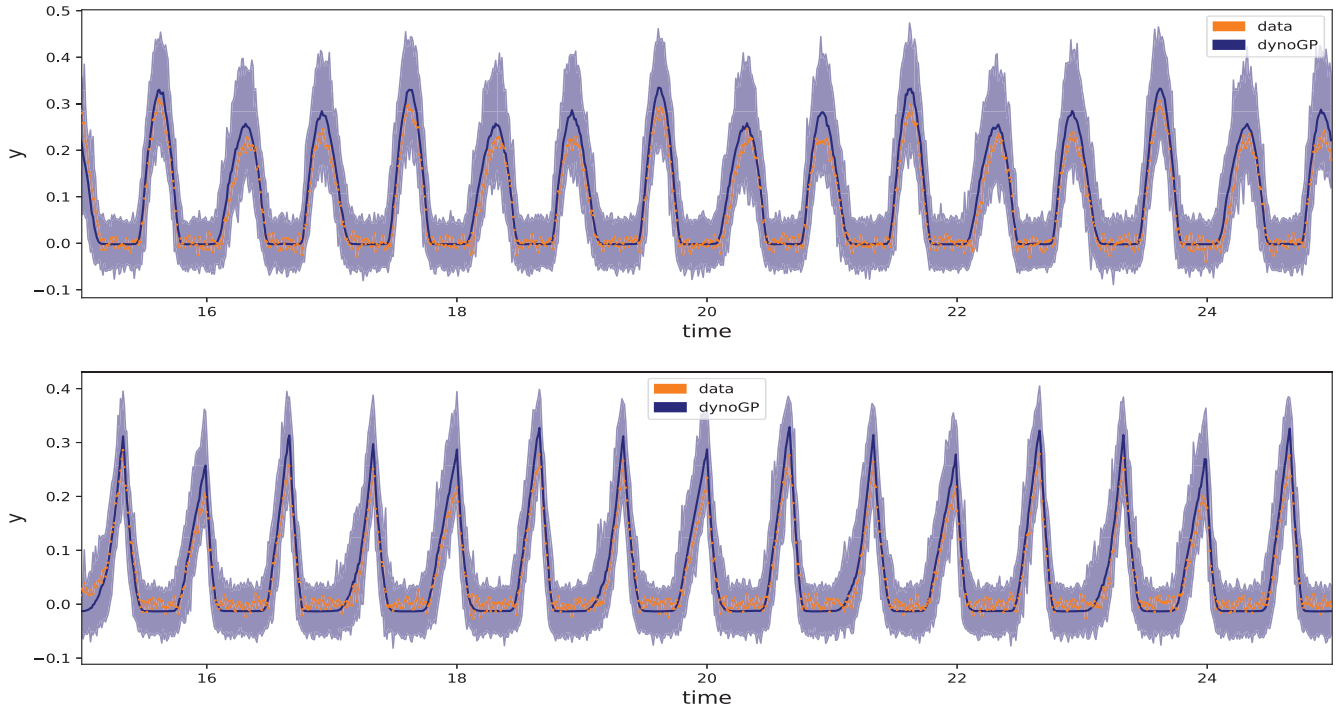
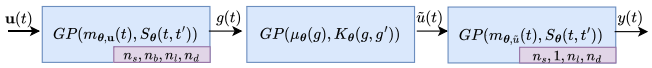


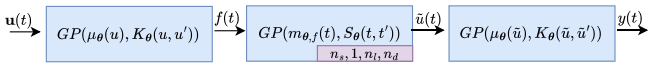
FIGURE 3 | We trained dynoGP with a Wiener architecture on the dataset of Figure 2 and use it to predict the response of the nonlinear system to the input signals $\mathbf{u}(t) = [\sin(3\pi t), \sin(5\pi t)]$ (top) and $\mathbf{u}(t) = [\text{sawtooth}(3\pi t), \text{sawtooth}(5\pi t)]$ (bottom). The plots compare the dynoGP predictions with the actual response of the nonlinear system for these input signals.

layer inherently has a non-zero non-constant mean, derived from the LTI system. For static GP, we use a constant mean function when the static layer is an intermediate/final layer and a linear mean function when it is an input layer.

The extension to architectures such as Wiener–Hammerstein (WH)



or Hammerstein–Wiener (HW)



can be similarly defined.[8] Note that, in the WH or HW case, the input can have a dimension $n_u > 1$, but the output of the dynamic or static GP has always dimension one. In other words, so far we have assumed that the static and dynamic have a MISO structure. Therefore, the other blocks following the first one have scalar signals as input and output.

Our model can be generalised to a MIMO (multiple input, multiple output) structure by incorporating multiple GPs per layer (of the same type, static or dynamic). As an example, see Figure 7, where the input signal $\mathbf{u}(t)$ is a vector of dimension 22. By employing three parallel MISO dynamic blocks in the first layer, the output $\mathbf{g}(t)$ becomes a vector of dimension 3, obtained by stacking the three scalar outputs of these blocks. Therefore, the static blocks in the second layer have an input of dimension 3. By stacking the output of these three MISO static layers, the resulting output $\tilde{\mathbf{u}}(t)$ is also a vector of dimension 3. We will use these more general architectures in the experiment section. For this general architecture, we can prove the following result.

Proposition 6. *The variational distribution and the evidence lower bound have the same functional form as in Equations* (37) *and*(38)*, up to replacing the model-specific parameters and architecture by their generalized counterparts.*

Note that, *skip connections*[9] can be easily introduced to further enhance the model's flexibility. We implemented *dynoGP* in *GPyTorch* [53]. Additional details for the selection of the inducing points, the prior over the hyperparameters, and the link to the GitHub repository are provided in Appendix D.

4 | Numerical Experiments

To demonstrate the effectiveness of the proposed dynoGP architecture, we consider the following case studies:

- identification of a simulated Wiener system;
- identification of the WH benchmark with process noise [54];
- identification the coupled-electric drives benchmark [55];
- forecast of electricity demand [56].

The performance of the estimated models is measured in terms of *root mean square error* (RMSE) $\sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$, *mean absolute error* (MAE) $\frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$, and *continuous ranked probability score* (CRPS) $\frac{1}{N}\sum_{i=1}^{N}\int_{-\infty}^{\infty}(F_i(x) - H(x - y_i))^2\,dx$, where y_i is the

10991115, 0, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/acs.70141 by Zhaw Zurcher Hochschule Fur, Wiley Online Library on [12/09/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

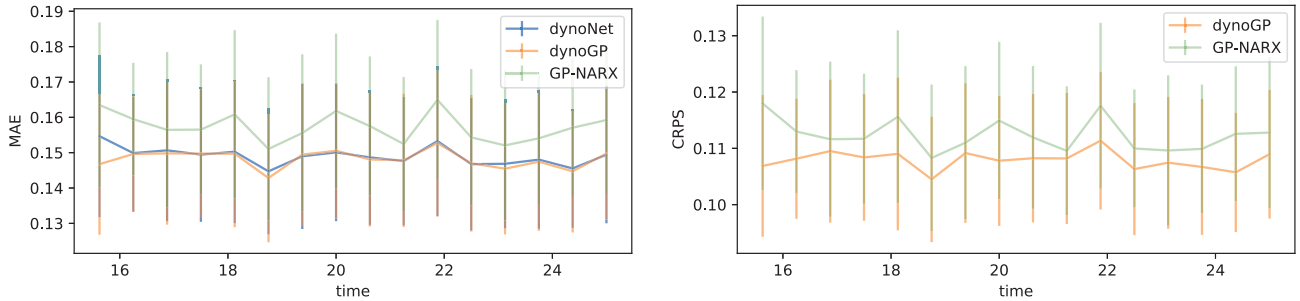


FIGURE 4 | Predictive MAE and CRPS for $15 \leq t \leq 25$ for dynoNet, dynoGP, and GP-NARX for simulated data from a Wiener system.

observed output at time stamp i, $\hat{y}_i$ is the predicted output, N is the number of observations, H is the Heaviside step function, and $F_i(x)$ is the cumulative distribution function associated with the prediction of the ith output sample. It is worth remarking that the RMSE and the MAE are computed using the mean of the estimated posterior, while the CRPS allows measuring the performance of the approach by evaluating the entire probabilistic output. This metric is particularly useful for evaluating predictions that express uncertainty in their outputs, providing a measure of forecast quality that accounts for both the entire probability distribution. In case of point estimate, the CRPS coincides with the MAE. In the following, we compute the CRPS by sampling from the posterior distribution. In the simulations, we compare the accuracy of dynoGP, dynoNet, GP-NARX, SB-MLP, and SB-LSTM. Importantly, since dynoGP is a Bayesian model, it provides a measure of uncertainty of its own predictions. For this reason, apart from dynoNet, we only compare it with other Bayesian methods. The comparison with dynoNet is reported as an accuracy-performance reference, as dynoNet can be seen as the deterministic counterpart of dynoGP. Therefore, the performance of dynoNet in terms of pure accuracy allows one to also assess the performance of dynoGP relative to other deterministic system identification methods.[10]

For dynoNet, we employ the same architecture and dimensions for the dynamic layer as dynoGP. For the static layer, we use a feedforward network with one hidden layer containing 32 neurons. For dynoGP, we use a Matern 3/2 kernel in the static layer. For the choice of architecture (interconnection of linear and static blocks), we consistently use a LIN-NONLIN-LIN structure, with variations only in the number of blocks. An exception is the first simulation, where we employed a Wiener architecture, as it was used to generate the data.

GP-NARX is a GP model that uses past inputs and measured outputs as covariates, with input and output lags of order n_{lags}. It is worth noting that, in GP-NARX, the testing phase involves iteratively constructing the covariate using past predicted outputs, as the measured outputs are unavailable during prediction.

Sparse Bayesian MLP and LSTM (SB-MLP and SB-LSTM) are Bayesian deep neural networks whose posterior distribution is approximated by using Laplace approximation. These architectures, as described in [19], are designed specifically for SISO systems. Consequently, we will report the comparison with SB-MLP and SB-LSTM only for the numerical experiments relative to SISO systems.

Further details about: (i) the implementation of dynoGP used in the experiments are provided in Appendix D; (ii) the procedure used for hyperparameters selection for dynoGP, GP-NARX and SB-MLP, and SB-LSTM are provided in Appendix D: Hyperparameters Selection; the comparison of the computational time are provided in Appendix D: Computational Time.

4.1 | Simulated Wiener System With Process Noise

In the first experiment, we simulated data from a Wiener system, which consists of an LTI system followed by the nonlinearity $(\cdot)^+$ (that is, positive part). We considered a setup similar to those in Examples 2 and 3. Specifically, we used a two-dimensional input signal and simulated an output signal $y(t)$ using an LTI system with a 5-dimensional state, followed by a quadratic nonlinearity. The LTI system has a transition matrix defined as diagonal plus rank-one, $-\Lambda - \mathbf{v}\mathbf{v}^\top$, where the elements of Λ and $\mathbf{v}$ were independently and uniformly generated in $[0, 1]$. The remaining parameters, C, B, and L, were independently sampled from normal distributions with variances 1, 1, and 0.1, respectively. The standard deviation ς of the measurement noise was set to 10% of the standard deviation of the system's output $y(t)$. A sampling time of $\delta = 0.01$ was used, and data were generated over the interval $t \in [0, 25]$. For system identification, data from $t \leq 15$ were used. The remaining data for $t > 15$, generated without process noise, were used for testing the model. We performed 50 Monte Carlo simulations in total. To identify the system and predict the test data, we used a dynoGP with a Wiener architecture, as described in Section 3. The dynamic GP layer was initialized with a system dimension of 10. For GP-NARX, we used input and output lags of order $n_{lags} = 5$.

Figure 4 top shows the MAE for the test data, showing the median and its standard deviation across the 50 Monte Carlo simulations as a function of time. The large error bars are caused by a high level of measurement noise. DynoGP and dynoNet provide a better performance than GP-NARX. Note that, in the initial stages of the prediction task ($15 < t < 16.5$, corresponding to 150 values), the performance of dynoNet is worse than dynoGP. This is theoretically expected, as dynoGP exhibits higher accuracy in short-term forecasting. The reason is that dynoNet estimates only the deterministic part of the system, while the system's dynamics also depends on the process noise, which drives the output signal in the initial part of the testing task (fading memory effect). GP-NARX's higher MAE is due to usage of noisy covariates in the training data, which has an impact in the testing, where

10991115, 0, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/acs.70141 by Zhaw Zurcher Hochschule Fur, Wiley Online Library on [12/09/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

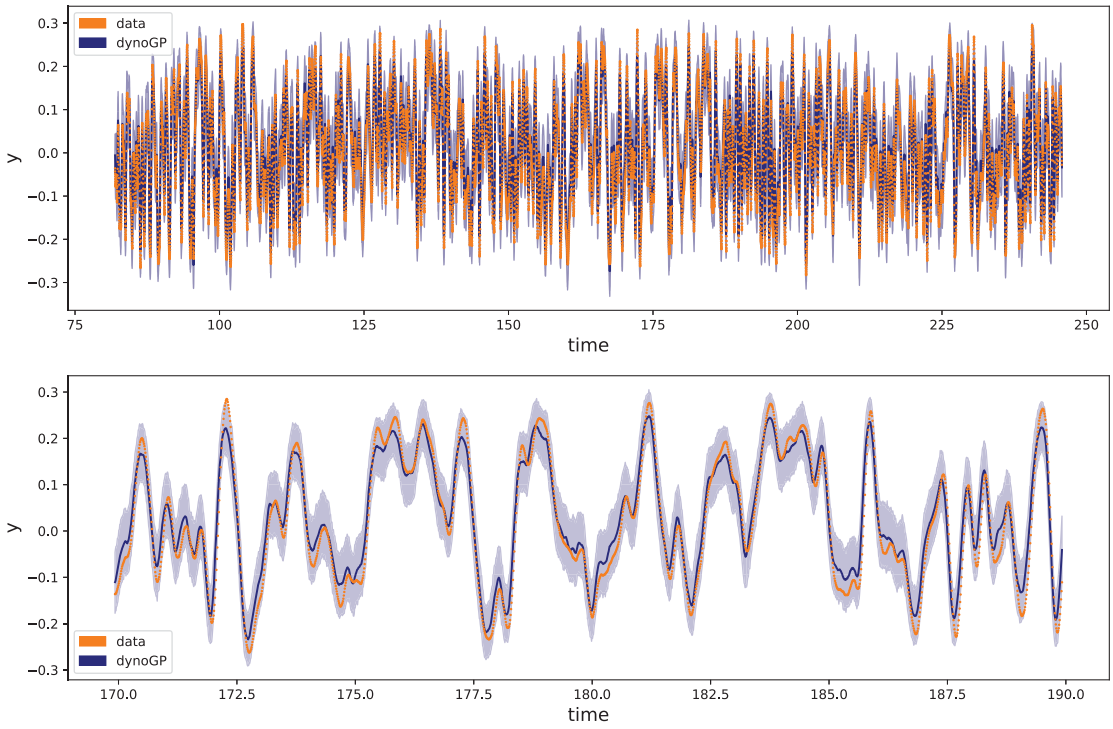


FIGURE 5 | Wiener–Hammerstein model with process noise. The output signal is in orange, the predictive posterior mean and 95% credible interval for dynoGP are shown for all testing data (top) and for a subset of testing data (bottom).

past simulated (instead of measured) outputs are used. Figure 4 bottom shows the CRPS for the two probabilistic methods. We observe that dynoGP provides better probabilistic predictions than GP-NARX.

4.2 | WH With Process Noise

As a second case study, we consider the identification of the SISO WH benchmark system described in [54]. The system is characterised by a static nonlinearity sandwiched between two LTI blocks. A process noise enters the system before the static nonlinearity, which makes the benchmark particularly challenging.

For identification via dynoGP, we consider the following WH architecture:

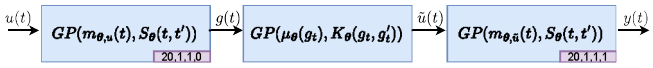


Figure 5 top displays the output signal on the test dataset alongside the dynoGP posterior mean and the 95% credible interval obtained by dynoGP. For enhanced visualization, a subset of the test set is presented in Figure 5 bottom. A comparison with the dynoNet, GP-NARX, SB-MLP, and SB-LSTM is detailed in Table 1, focusing on the performance metrics RMSE, MAE, and CRPS. It is notable that the model's uncertainty is significantly lower compared to GP-NARX, SB-MLP, and SB-LSTM, as evidenced by the lower CRPS score. Additionally, both dynoGP and dynoNet architectures outperform GP-NARX, SB-MLP, and SB-LSTM in terms of RMSE and MAE, with dynoGP slightly outperforming dynoNet.

4.3 | Coupled Electric Drives

As a third case study, we consider the identification of the SISO coupled-electric drives benchmark [55], which involves two electric motors driving a pulley via a flexible belt. The pulley is supported by a spring, introducing a lightly damped dynamic mode. The input is the sum of the voltage applied to the motors, while the angular speed of the pulley is the output of interest. The dataset comprises 500 samples, with the first 400 used for training and the remaining 100 for testing.

We considered the following dynoGP architecture:

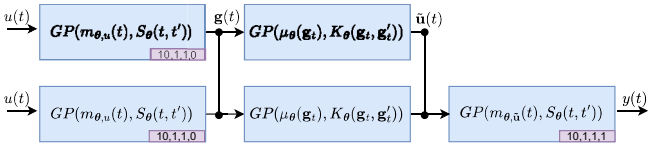


Figure 6 top illustrates the input signal, while Figure 6 bottom presents the output signal (in orange) alongside the dynoGP posterior mean and the 95% credible interval (in blue). The values on the left of the vertical bar correspond to the 400 data points in the training set, whereas those on the right represent the test set. It is evident that the model uncertainty is relatively high, which can be attributed to the small size of the training dataset (only 400 data points). Nevertheless, the model's uncertainty is notably lower compared to GP-NARX, as evidenced by the comparison

10991115, 0, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/acs.70141 by Zhaw Zurcher Hochschule Fur, Wiley Online Library on [12/09/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

TABLE 1 | Performance comparison of dynoGP, dynoNet, GP-NARX, SB-MLP, and SB-LSTM in RMSE, MAE, and CRPS across benchmark datasets: Wiener–Hammerstein (WH), coupled electric drives (CED), and electricity demand forecast (ELE).

	dynoGP			dynoNet		GP-NARX			SB-MLP			SB-LSTM		
Datasets	**RMSE**	**MAE**	**CRPS**	**RMSE**	**MAE**	**RMSE**	**MAE**	**CRPS**	**RMSE**	**MAE**	**CRPS**	**RMSE**	**MAE**	**CRPS**
WH	0.029	0.024	0.016	0.034	0.028	0.120	0.096	0.077	0.044	0.036	0.032	0.096	0.077	0.052
CED	0.147	0.109	0.081	0.105	0.075	0.247	0.174	0.132	0.159	0.103	0.084	0.181	0.156	0.113
ELE	291	215	197	297	235	376	273	258	NA	NA	NA	NA	NA	NA

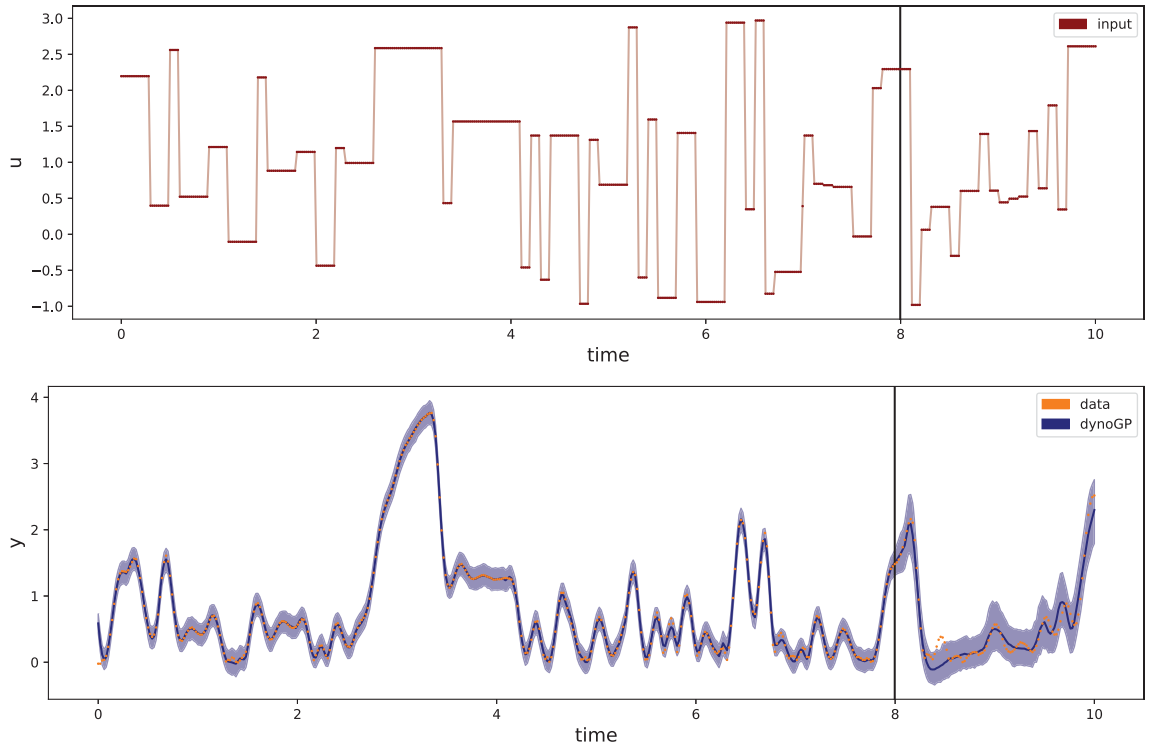


FIGURE 6 | CED: The first plot shows the input signal. Values on the left of the vertical bar represent the 400 values of the training set, while those on the right correspond to the test set. The second plot shows the values of the output signal (in orange) and the predictive posterior mean and 95% credible interval for dynoGP in blue.

of the CRPS of the two models in Table 1. The other metrics are also reported in Table 1. Both dynoGP and dynoNet outperform the GP-NARX, SB-MLP, and SB-LSTM architectures, while the performance of dynoNet is slightly better than dynoGP. However, we remind that while dynoNet provides a point estimate, dynoGP provides additional information consisting of the probability distribution of the predicted output.

4.4 | Electricity Demand Forecast

Finally, we considered the problem of forecasting electricity demand (in gigawatt) as a function of weather-temperature (in Celsius). The dataset comprises electricity demand in the state of Victoria, Australia, for the years 2014 and 2015. It includes two time series recorded at 30-minute intervals: electricity demand and maximum temperature. Daily electricity demand is strongly influenced by temperature, with higher consumption observed on colder days due to heating and on hotter days due to air conditioning. This relationship exhibits nonlinearity due to threshold effects, emphasizing the necessity of employing both non-linear and dynamic models to capture the underlying patterns accurately. For this reason, we use an architecture which combines linear dynamical GPs and static GPs. In particular, we consider the architecture in Figure 7.

For our analysis, we used 18 months of data for training (comprising 26,304 values) and 6 months for testing (8736 values). The inputs to the model included maximum temperature, an indicator variable denoting whether a day was a public holiday, and Fourier harmonics:

$$[\cos(2\pi t m_i), \sin(2\pi t m_i), \ldots, \cos(2\pi t k_i m_i), \sin(2\pi t k_i m_i)]$$

for $t = 0, \delta, 2\delta, \ldots$ and $\delta = 1/(48 \cdot 365)$ and three terms for the three periods: daily ($m_1 = 365$, $k_1 = 4$), weekly ($m_2 = 52.14$, $k_2 = 4$), and yearly ($m_2 = 1$, $k_2 = 2$). In total, we included 22 input signals for forecasting electricity demand. It is important to note that this setup is not entirely realistic, as one of the input, maximum temperature, is unavailable for more than 15 days ahead, which

10991115, 0, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/acs.70141 by Zhaw Zurcher Hochschule Fur, Wiley Online Library on [12/09/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

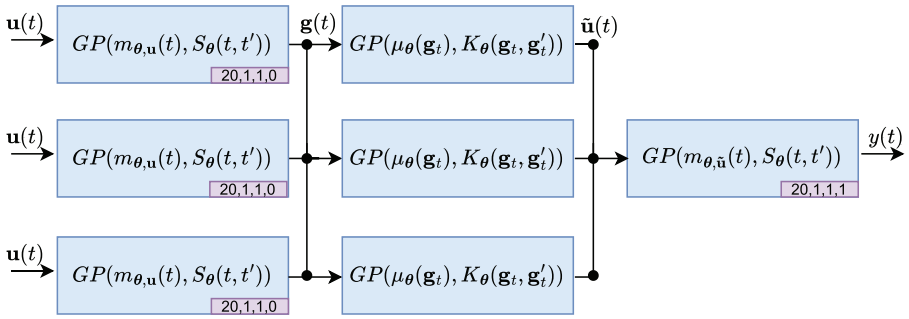


FIGURE 7 | Architecture used in the electricity demand forecast task.

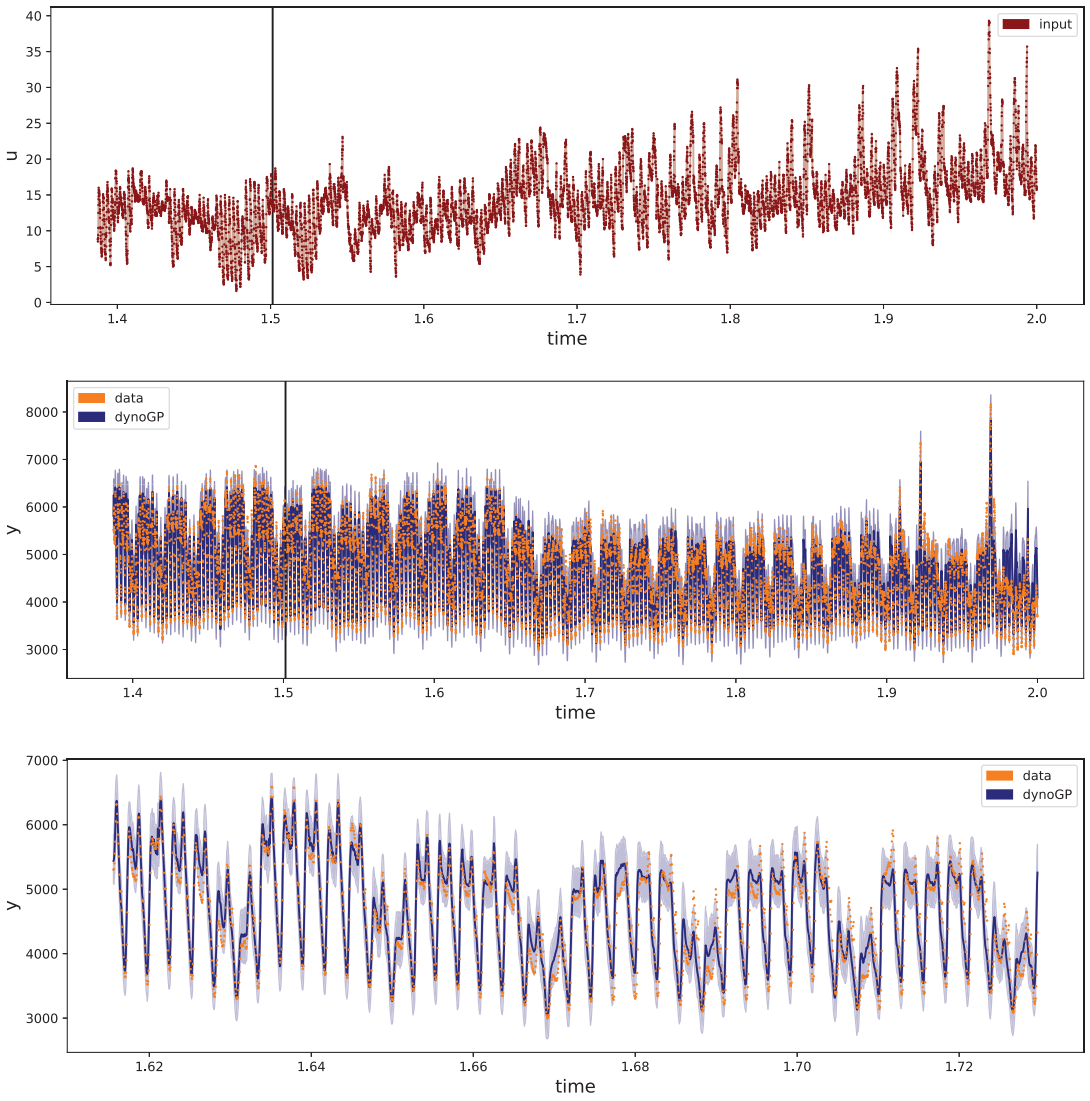


FIGURE 8 | Electricity demand dataset. The first plot shows the maximum temperature (input). Values on the left of the vertical bar represent the last 2000 values of the training set, while those on the right correspond to the test set. The second plot includes the corresponding values of the electricity demand (in orange) and shows the predictive posterior mean and 95% credible interval for dynoGP. The last plot is a zoomed-in view for a subset of the prediction task.

is the typical weather forecasting horizon. Instead, our testing set includes predictions for up to 6 months ahead. In this section, our primary objective is to use this time series to learn the corresponding dynamical system and test dynoGP performance for long time horizon and in a scenario with many input signals.

Figure 8 (first and second plot) shows a subset of the input temperature and electricity demand time series. The values on the left of the vertical bar represent the last 2000 observations of the training set, while those on the right correspond to the test set. The third plot in Figure 8 displays the posterior mean and

10991115, 0, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/acs.70141 by Zhaw Zurcher Hochschule Fur, Wiley Online Library on [12/09/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

95% credible interval for dynoGP (the third plot is a zoomed-in view). The results indicate that the model predicts electricity demand accurately even 6 months ahead. Notably, it performs exceptionally well in forecasting peak demands, which are crucial for ensuring the stability of the electricity grid. Note that, the model's performance diminishes in the last part of the testing set. This pattern is not evident from the temperature, suggesting the presence of other influencing inputs not captured by the model. As in the other benchmarks, a comparison with dynoNet and GP-NARX in terms of RMSE, MAE, and CRPS is also provided in Table 1.

5 | Conclusions

The deep dynamic GP architecture (dynoGP) presented in this paper is based on a time-dependent GP process associated to stochastic LTI dynamical systems with a complex diagonal state-transition matrix. This setup facilitates an efficient and closed-form expression for the mean and covariance function of the dynamic GP. The linear time-invariant GP can be cascaded with other GP blocks, both static and dynamic, to form the dynoGP architecture, where the posterior distribution of the predicted output can be approximated through SVI.

As future work, we plan to explore three key aspects. First, we aim to investigate model selection for the dynoGP architecture by leveraging Pareto smoothed importance sampling leave-one-out cross-validation (PSIS-LOO) [57]. Exact cross-validation requires re-fitting the model on different training sets, which can be computationally expensive. In contrast, PSIS-LOO can be efficiently computed using samples from the posterior distribution. Second, we intend to develop a principled approach for initializing the parameters of the dynamic and static GP layers. Our idea is to employ a hierarchical Bayesian framework, as discussed in [58, 59] for GP-based time-series forecasting. Additionally, we believe that this approach can be adapted to extend dynoGP for metalearning tasks. Metalearning involves training models to quickly adapt to new tasks using only a small amount of data, which aligns well with the flexibility and expressiveness of Gaussian processes. By leveraging the hierarchical Bayesian approach, we can encode prior knowledge about task distributions into the model, enabling it to generalise across tasks more effectively. This could make dynoGP particularly well-suited for applications where task-specific data is limited but shared structure across tasks can be exploited. Third, we plan to apply dynoGP to various sequence modeling problems (e.g., images, audio, text, time-series), leveraging its similarity to S4 models (like S4, Mamba, which have demonstrated strong performance on established benchmarks), while providing a full probabilistic output.

Funding

This publication has emanated from research conducted with the financial support of Research Ireland (Grant No. 24/US/3978). This work was supported as a part of NCCR Automation, a National Centre of Competence in Research, funded by the Swiss National Science Foundation (Grant No. 51NF40_225155).

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

The data that support the findings of this study are openly available in dynoDeepGP at https://github.com/benavoli/dynoDeepGP.

Endnotes

[1]Sparse Bayesian MLP and LSTM models are limited to SISO systems. In contrast, DynoGP and GP-NARX can handle MISO systems and, through parallelisation, MIMO systems as well.

[2]We use the notation $K(\mathbf{x}, \mathbf{x}')$ with $\mathbf{x}$ and $\mathbf{x}'$ to denote the evaluation of the kernel at two different points.

[3]We use n_b, n_d to denote the number of columns of $\boldsymbol{B}$ and $\boldsymbol{D}$, but obviously if $\mathbf{u}(t) \in \mathbb{R}^{n_u}$ then either $n_b = n_d = n_u$.

[4]To simplify the notation, we denote the mean of $y(t)$ by $m_\theta(t)$ (not in bold).

[5]The name Cauchy comes from the fact that this kernel can be derived from Bochner's theorem using a mixture of Cauchy distributions as its spectral density, see [48, 49].

[6]If n_b or n_d is different from zero, then they are equal to n_u.

[7]To mitigate the effect of the initial condition, we typically perform predictions over the interval $\tilde{T} = [T_{\text{end}}, T^*]$, where T_{end} denotes the last 200 time points of T.

[8]n_d can be zero if we remove the input feedthrough term. We usually assume that $n_l = 1$.

[9]Connections that bypass one or more layers and feed an earlier layer's output directly into a later layer.

[10]See, for instance, the results published at https://www.nonlinearbenchmark.org/results-reporting.

References

1. S. Chen, S. A. Billings, and P. M. Grant, "Non-Linear System Identification Using Neural Networks," *International Journal of Control* 51, no. 6 (1990): 1191–1214.

2. J. Wray and G. Green, "Neural Networks for Nonlinear Model Structures," *IEEE Transactions on Control Systems Technology* 2, no. 1 (1994): 31–41.

3. K. S. Narendra and K. Parthasarathy, "Identification and Control of Dynamical Systems Using Neural Networks," *IEEE Transactions on Neural Networks* 1, no. 1 (1990): 4–27.

4. G. Pillonetto, A. Aravkin, D. Gedon, L. Ljung, A. H. Ribeiro, and T. B. Schön, "Deep Networks for System Identification: A Survey," *Automatica* 171 (2025): 111907.

5. C. Andersson, A. H. Ribeiro, K. Tiels, N. Wahlström, and T. B. Schön, "Deep Convolutional Networks in System Identification," in *2019 IEEE 58th Conference on Decision and Control (CDC)* (IEEE, 2019), 3670–3676.

6. S. Bai, J. Z. Kolter, and V. Koltun, "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling," 2018 arXiv preprint arXiv:180301271.

7. Y. Wang and A. Gonzalez, "System Identification Using Deep Learning: State-Of-The-Art and Future Research Directions," *Annual Reviews in Control* 43 (2017): 201–213.

8. G. I. Beintema, R. Tóth, and M. Schoukens, "Nonlinear State-Space Identification Using Deep Encoder Networks," in *Learning for Dynamics and Control* (PMLR, 2021), 241–250.

9. D. Masti and A. Bemporad, "Learning Nonlinear State–Space Models Using Autoencoders," *Automatica* 129 (2021): 109666.

10991115, 0, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/acs.70141 by Zhaw Zurcher Hochschule Fur, Wiley Online Library on [12/09/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

10. D. Gedon, N. Wahlström, T. B. Schön, and L. Ljung, "Deep State Space Models for Nonlinear System Identification," *IFAC PapersOnline* 54, no. 7 (2021): 481–486.

11. M. Forgione and D. Piga, "Model Structures and Fitting Criteria for System Identification With Neural Networks," in *2020 IEEE 14th International Conference on Application of Information and Communication Technologies (AICT)* (IEEE, 2020), 1–6.

12. M. Forgione and D. Piga, "*dynoNet*: A Neural Network Architecture for Learning Dynamical Systems," *International Journal of Adaptive Control and Signal Processing* 35, no. 4 (2021): 612–626.

13. A. Gu, K. Goel, A. Gupta, and C. Ré, "On the Parameterization and Initialization of Diagonal State Space Models," *Advances in Neural Information Processing Systems* 35 (2022): 35971–35983.

14. A. Orvieto, S. L. Smith, A. Gu, et al., "Resurrecting Recurrent Neural Networks for Long Sequences," in *International Conference on Machine Learning* (PMLR, 2023), 26670–26698.

15. S. Boyd and L. Chua, "Fading Memory and the Problem of Approximating Nonlinear Operators With Volterra Series," *IEEE Transactions on Circuits and Systems* 32, no. 11 (1985): 1150–1161.

16. M. Schetzen, *The Volterra and Wiener Theories of Nonlinear Systems* (Krieger Publishing Co., Inc., 2006).

17. A. Damianou and N. D. Lawrence, "Deep Gaussian Processes," in *Artificial Intelligence and Statistics* (PMLR, 2013), 207–215.

18. M. Champneys, G. I. Beintema, R. Tóth, M. Schoukens, and T. J. Rogers, "Baseline Results for Selected Nonlinear System Identification Benchmarks," *IFAC-PapersOnLine* 58, no. 15 (2024): 474–479.

19. H. Zhou, C. Ibrahim, W. X. Zheng, and W. Pan, "Sparse Bayesian Deep Learning for Dynamic System Identification," *Automatica* 144 (2022): 110489.

20. H. Salimbeni and M. Deisenroth, "Doubly Stochastic Variational Inference for Deep Gaussian Processes," *Advances in Neural Information Processing Systems* 9 (2017): 1–12.

21. A. O'Hagan, "Curve Fitting and Optimal Design for Prediction," *Journal of the Royal Statistical Society. Series B, Statistical Methodology* 40, no. 1 (1978): 1–24.

22. C. E. Rasmussen and C. K. Williams, *Gaussian Processes for Machine Learning* (MIT Press, 2006).

23. M. Soleimani, M. Esmaeilbeigi, R. Cavoretto, and A. De Rossi, "Analyzing the Effects of Various Isotropic and Anisotropic Kernels on Critical Heat Flux Prediction Using Gaussian Process Regression," *Engineering Applications of Artificial Intelligence* 133 (2024): 108351, https://www.sciencedirect.com/science/article/pii/S0952197624005098.

24. M. Esmaeilbeigi, O. Chatrabgoun, A. Daneshkhah, and M. Shafa, "On the Impact of Prior Distributions on Efficiency of Sparse Gaussian Process Regression," *Engineering With Computers* 39, no. 4 (2023): 2905–2925.

25. C. Williams, "Computing With Infinite Networks," *Advances in Neural Information Processing Systems* 9 (1996): 295–301.

26. Y. Cho and L. Saul, "Kernel Methods for Deep Learning," in *Advances in Neural Information Processing Systems*, vol. 22, ed. Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta (Curran Associates, Inc., 2009).

27. J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. N. Sohl-Dickstein, "Deep Neural Networks as Gaussian Processes," in *International Conference on Learning Representations* (Curran Associates, Inc., 2018).

28. T. Bui, D. Hernández-Lobato, J. Hernandez-Lobato, Y. Li, and R. Turner, "Deep Gaussian Processes for Regression Using Approximate Expectation Propagation," in *International Conference on Machine Learning* (PMLR, 2016), 1472–1481.

29. M. Bauer, M. van der Wilk, and C. E. Rasmussen, "Understanding Probabilistic Sparse Gaussian Process Approximations," *Advances in Neural Information Processing Systems*, 29 (2016): 1533–1541.

30. J. Hensman, N. Fusi, and N. D. Lawrence, "Gaussian Processes for Big Data," in *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence UAI'13* (AUAI Press, 2013), 282–290.

31. D. Hernandez-Lobato and J. M. Hernandez-Lobato, "Scalable Gaussian Process Classification via Expectation Propagation," in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics* (PMLR, 2016), 168–176, https://proceedings.mlr.press/v51/hernandez-lobato16.html.

32. J. Quiñonero-Candela and C. E. Rasmussen, "A Unifying View of Sparse Approximate Gaussian Process Regression," *Journal of Machine Learning Research* 6, no. Dec (2005): 1939–1959.

33. E. Snelson and Z. Ghahramani, "Sparse Gaussian Processes Using Pseudo-Inputs," *Advances in Neural Information Processing Systems* 29 (2006): 1257–1264.

34. M. Titsias, "Variational Learning of Inducing Variables in Sparse Gaussian Processes," in *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, vol. 5, ed. D. van Dyk and M. Welling (PMLR, 2009), 567–574.

35. M. Schuerch, D. Azzimonti, A. Benavoli, and M. Zaffalon, "Recursive Estimation for Sparse Gaussian Process Regression," *Automatica* 120 (2020): 109–127.

36. M. Schuerch, D. Azzimonti, A. Benavoli, and M. Zaffalon, "Correlated Product of Experts for Sparse Gaussian Process Regression," *Machine Learning* 112, no. 5 (2023): 1411–1432.

37. C. G. Kaufman, M. J. Schervish, and D. W. Nychka, "Covariance Tapering for Likelihood-Based Estimation in Large Spatial Data Sets," *Journal of the American Statistical Association* 103, no. 484 (2008): 1545–1555.

38. M. Katzfuss and J. Guinness, "A General Framework for Vecchia Approximations of Gaussian Processes," *Statistical Science* 36, no. 1 (2021): 124–141.

39. M. Esmaeilbeigi, O. Chatrabgoun, R. Cavoretto, A. De Rossi, and M. Shafa, "Sparse Gaussian Process Regression via Compactly Supported Kernels: A Trade-Off Between Accuracy and Computational Efficiency," *Information Sciences* 717 (2025): 122306.

40. S. Särkkä and A. Solin, *Applied Stochastic Differential Equations*, vol. 10 (Cambridge University Press, 2019).

41. J. Hartikainen and S. Särkkä, "Kalman Filtering and Smoothing Solutions to Temporal Gaussian Process Regression Models," in *2010 IEEE International Workshop on Machine Learning for Signal Processing* (IEEE, 2010), 379–384.

42. A. Carron, M. Todescato, R. Carli, L. Schenato, and G. Pillonetto, "Machine Learning Meets Kalman Filtering," in *2016 IEEE 55th Conference on Decision and Control (CDC)* (IEEE, 2016), 4594–4599.

43. J. Loper, D. Blei, J. P. Cunningham, and L. Paninski, "A General Linear-Time Inference Method for Gaussian Processes on One Dimension," *Journal of Machine Learning Research* 22, no. 234 (2021): 1–36.

44. A. Solin and S. Särkkä, "Explicit Link Between Periodic Covariance Functions and State Space Models," in *Artificial Intelligence and Statistics* (PMLR, 2014), 904–912.

45. A. Benavoli and M. Zaffalon, "State Space Representation of Non-Stationary Gaussian Processes," 2016 arXiv preprint arXiv:160101544.

46. F. Lindgren, H. Rue, and J. Lindström, "An Explicit Link Between Gaussian Fields and Gaussian Markov Random Fields: The Stochastic Partial Differential Equation Approach," *Journal of the Royal Statistical Society, Series B: Statistical Methodology* 73, no. 4 (2011): 423–498.

10991115, 0, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/acs.70141 by Zhaw Zurcher Hochschule Fur, Wiley Online Library on [12/09/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

47. M. Todescato, A. Carron, R. Carli, G. Pillonetto, and L. Schenato, "Efficient Spatio-Temporal Gaussian Regression via Kalman Filtering," *Automatica* 118 (2020): 109032.

48. M. Zorzi and A. Chiuso, "The Harmonic Analysis of Kernel Functions," *Automatica* 94 (2018): 125–137.

49. Y. Zhang and C. Hua, "Spectral Mixture Kernels for Bayesian Optimization," 2025 arXiv preprint arXiv:250517393.

50. D. J. Sutherland and J. Schneider, "On the Error of Random Fourier Features," 2015 arXiv preprint arXiv:150602785.

51. A. Gu, T. Dao, S. Ermon, A. Rudra, and C. Ré, "Hippo: Recurrent Memory With Optimal Polynomial Projections," *Advances in Neural Information Processing Systems* 33 (2020): 1474–1487.

52. A. Gupta, A. Gu, and J. Berant, "Diagonal State Spaces Are as Effective as Structured State Spaces," *Advances in Neural Information Processing Systems* 35 (2022): 22982–22994.

53. J. Gardner, G. Pleiss, K. Q. Weinberger, D. Bindel, and A. G. Wilson, "Gpytorch: Blackbox Matrix-Matrix Gaussian Process Inference With GPU Acceleration," *Advances in Neural Information Processing Systems* 31 (2018): 1–11.

54. M. Schoukens and J. P. Noël, "Three Benchmarks Addressing Open Challenges in Nonlinear System Identification," *IFAC PapersOnline* 50, no. 1 (2017): 446–451.

55. T. Wigren and M. Schoukens, *Coupled Electric Drives Data Set and Reference Models* (Department of Information Technology, Uppsala University, 2017).

56. R. Godahewa, C. Bergmeir, G. Webb, R. Hyndman, and P. Montero-Manso, "Australian Electricity Demand Dataset" (2017), Zenodo, 10.5281/zenodo.4659727.

57. A. Vehtari, A. Gelman, and J. Gabry, "Practical Bayesian Model Evaluation Using Leave-One-Out Cross-Validation and WAIC," *Statistics and Computing* 27 (2017): 1413–1432.

58. G. Corani, A. Benavoli, and M. Zaffalon, "Time Series Forecasting With Gaussian Processes Needs Priors," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (Springer, 2021), 103–117.

59. A. Benavoli and G. Corani, *State Space Approximation of Gaussian Processes for Time Series Forecasting* (Springer International Publishing, 2021), 21–35.

60. R. A. Horn and C. R. Johnson, *Matrix Analysis* (Cambridge University Press, 2012).

61. C. M. Carvalho, N. G. Polson, and J. G. Scott, "Handling Sparsity via the Horseshoe," in *Artificial Intelligence and Statistics* (PMLR, 2009), 73–80.

62. M. Jankowiak, G. Pleiss, and J. Gardner, "Deep Sigma Point Processes," in *Conference on Uncertainty in Artificial Intelligence* (PMLR, 2020), 789–798.

63. D. Ming, D. Williamson, and S. Guillas, "Deep Gaussian Process Emulation Using Stochastic Imputation," *Technometrics* 65, no. 2 (2023): 150–161.

Appendix A: Deriving the 1-Dimensional Matern 3/2 Kernel From a Stochastic LTI System

Consider the LTI system in (6) with

$$A = \begin{bmatrix} 0 & 1 \\ -\frac{3}{\ell^2} & -2\frac{\sqrt{3}}{\ell} \end{bmatrix}, \quad L = \sigma \begin{bmatrix} 0 \\ \sqrt{\frac{36}{\ell^3}} \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 \end{bmatrix}, \tag{A1}$$

with $\boldsymbol{\theta} = [\sigma^2, \ell]$, $\ell > 0$, and $B = \mathbf{0}$. The steady-state variance is

$$\Sigma_\infty = \sigma^2 \begin{bmatrix} \sqrt{3} & 0 \\ 0 & \frac{\sqrt{27}}{l^2} \end{bmatrix}, \tag{A2}$$

and

$$\Psi(t,t') = e^{A(t-t')} = \begin{bmatrix} e^{-\frac{\delta}{\ell}}(\frac{\delta}{\ell}+1) & \frac{\delta}{\sqrt{3}}e^{-\frac{\delta}{\ell}} \\ -\frac{\sqrt{3}\delta}{\ell^2}e^{-\frac{\delta}{\ell}} & e^{-\frac{\delta}{\ell}}(1-\frac{\delta}{\ell}) \end{bmatrix}, \tag{A3}$$

with $\delta = \sqrt{3}(t-t')$. Therefore, it is easy to verify that

$$\begin{aligned} C\Sigma_\infty(e^{A(t-t')})^\top C^\top &= Ce^{A(t'-t)}\Sigma_\infty C^\top \\ &= \sqrt{3}\sigma^2 e^{-\frac{|\delta|}{\ell}}\left(\frac{|\delta|}{\ell}+1\right), \end{aligned} \tag{A4}$$

which is proportional to $K_\theta(t,t')$ given in (1).

Appendix B: Proofs

Lemma 1. *The solution of the Lyapunov Equation* (17) *is*:

$$\Sigma_\infty^{(c)} = -\begin{bmatrix} \frac{LL^H}{\lambda+\lambda^\dagger} & \frac{LL^\top}{2\lambda} \\ \frac{L^\dagger L^H}{2\lambda^\dagger} & \frac{L^\dagger L^\top}{\lambda+\lambda^\dagger} \end{bmatrix}, \tag{B1}$$

where L, λ are the parameters in (14).

Proof of Lemma 1. The Lyapunov equation is

$$\begin{aligned} &A\Sigma_\infty + \Sigma_\infty A^H + LL^H \\ &= -\begin{bmatrix} \lambda & 0 \\ 0 & \lambda^\dagger \end{bmatrix}\begin{bmatrix} \frac{LL^H}{\lambda+\lambda^\dagger} & \frac{LL^\top}{2\lambda} \\ \frac{L^\dagger L^H}{2\lambda^\dagger} & \frac{L^\dagger L^\top}{\lambda+\lambda^\dagger} \end{bmatrix} - \begin{bmatrix} \frac{LL^H}{\lambda+\lambda^\dagger} & \frac{LL^\top}{2\lambda} \\ \frac{L^\dagger L^H}{2\lambda^\dagger} & \frac{L^\dagger L^\top}{\lambda+\lambda^\dagger} \end{bmatrix}\begin{bmatrix} \lambda^\dagger & 0 \\ 0 & \lambda \end{bmatrix} \\ &\quad + \begin{bmatrix} LL^H & LL^\top \\ L^\dagger L^H & L^\dagger L^\top \end{bmatrix} = 0. \end{aligned} \tag{B2}$$

□

Proof of Proposition 1. We use Lemma 1 and the expression of the kernel function defined in (19).

For $t \le t'$, the kernel function defined in (19) is equal to:

$$\begin{aligned} S_\theta(t,t') &= -\begin{bmatrix} c & c^\dagger \end{bmatrix}\begin{bmatrix} \frac{LL^H}{\lambda+\lambda^\dagger} & \frac{LL^\top}{2\lambda} \\ \frac{L^\dagger L^H}{2\lambda^\dagger} & \frac{L^\dagger L^\top}{\lambda+\lambda^\dagger} \end{bmatrix}\begin{bmatrix} e^{\lambda^\dagger(t'-t)} & 0 \\ 0 & e^{\lambda(t'-t)} \end{bmatrix}\begin{bmatrix} c^\dagger \\ c \end{bmatrix} \\ &= -e^{\lambda^\dagger(t'-t)}\left(\frac{cLL^Hc^\dagger}{\lambda+\lambda^\dagger} + \frac{c^\dagger L^\dagger L^H c^\dagger}{2\lambda^\dagger}\right) \\ &\quad - e^{\lambda(t'-t)}\left(\frac{cLL^\top c}{2\lambda} + \frac{c^\dagger L^\dagger L^\top c}{\lambda+\lambda^\dagger}\right). \end{aligned} \tag{B3}$$

For $t' < t$, the kernel function is equal to:

$$\begin{aligned} S_\theta(t,t') &= -\begin{bmatrix} c & c^\dagger \end{bmatrix}\begin{bmatrix} e^{\lambda(t-t')} & 0 \\ 0 & e^{\lambda^\dagger(t-t')} \end{bmatrix}\begin{bmatrix} \frac{LL^H}{\lambda+\lambda^\dagger} & \frac{LL^\top}{2\lambda} \\ \frac{L^\dagger L^H}{2\lambda^\dagger} & \frac{L^\dagger L^\top}{\lambda+\lambda^\dagger} \end{bmatrix}\begin{bmatrix} c^\dagger \\ c \end{bmatrix} \\ &= -e^{\lambda(t-t')}\frac{cLL^Hc^\dagger}{\lambda+\lambda^\dagger} - e^{\lambda^\dagger(t-t')}\frac{c^\dagger L^\dagger L^H c^\dagger}{2\lambda^\dagger} - e^{\lambda(t-t')}\frac{cLL^\top c}{2\lambda} \\ &\quad - e^{\lambda^\dagger(t-t')}\frac{c^\dagger L^\dagger L^\top c}{\lambda+\lambda^\dagger} \\ &= -e^{\lambda(t-t')}\left(\frac{cLL^Hc^\dagger}{\lambda+\lambda^\dagger} + \frac{cLL^\top c}{2\lambda}\right) \\ &\quad - e^{\lambda^\dagger(t-t')}\left(\frac{c^\dagger L^\dagger L^\top c}{\lambda+\lambda^\dagger} + \frac{c^\dagger L^\dagger L^H c^\dagger}{2\lambda^\dagger}\right). \end{aligned} \tag{B4}$$

□

10991115, 0, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/acs.70141 by Zhaw Zurcher Hochschule Fur, Wiley Online Library on [12/09/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

Proof of Proposition 2. The mean function of the system

$$dx_c(t) = \overbrace{\begin{bmatrix}\lambda\end{bmatrix}}^{A_c} x_c(t)dt + \overbrace{\begin{bmatrix}B\end{bmatrix}}^{B_c}\mathbf{u}(t)dt$$
$$y(t) = 2\Re\left(\overbrace{\begin{bmatrix}c\end{bmatrix}}^{C_c} x_c(t)\right) + D\mathbf{u}(t), \tag{B5}$$

is:

$$m_\theta(k\delta) = 2\Re\left(ce^{\lambda k\delta}\tilde{m}(0) + \sum_{i=1}^{k}\frac{c}{\lambda}e^{\lambda(i-1)\delta}(e^{\lambda\delta}-1)B\mathbf{u}(i\delta)\right) + D\mathbf{u}(k\delta). \tag{B6}$$

Now consider the system in (14), its mean function is:

$$\begin{aligned}
&m(k\delta) - D\mathbf{u}(k\delta)\\
&= C\left(\bar{A}^k\mathbf{m}(0) + \sum_{i=1}^{k}\bar{A}^{i-1}\overline{B}\mathbf{u}(i\delta)\right),\\
&= \begin{bmatrix}c & c^\dagger\end{bmatrix}\left(\begin{bmatrix}e^{\lambda\delta} & 0\\ 0 & e^{\lambda^\dagger\delta}\end{bmatrix}\mathbf{m}(0) + \sum_{i=1}^{k}\begin{bmatrix}e^{\lambda\delta(i-1)} & 0\\ 0 & e^{\lambda^\dagger\delta(i-1)}\end{bmatrix}\right.\\
&\quad\left.\cdot\begin{bmatrix}\frac{1}{\lambda} & 0\\ 0 & \frac{1}{\lambda^\dagger}\end{bmatrix}\begin{bmatrix}e^{\lambda\delta}-1 & 0\\ 0 & e^{\lambda^\dagger\delta}-1\end{bmatrix}\begin{bmatrix}B\\ B^\dagger\end{bmatrix}\mathbf{u}(i\delta)\right)\\
&= ce^{\lambda\delta}\tilde{m}(0) + c^\dagger e^{\lambda^\dagger\delta}\tilde{m}^\dagger(0) + \sum_{i=1}^{k}\left(\frac{c}{\lambda}e^{\lambda\delta(i-1)}(e^{\lambda\delta}-1)B\right.\\
&\quad\left.+\frac{c^\dagger}{\lambda^\dagger}e^{\lambda^\dagger\delta(i-1)}(e^{\lambda^\dagger\delta}-1)B^\dagger\right)\mathbf{u}(i\delta)\\
&= 2\Re\left(ce^{\lambda\delta}\tilde{m}(0) + \sum_{i=1}^{k}\frac{c}{\lambda}e^{\lambda\delta(i-1)}(e^{\lambda\delta}-1)B\mathbf{u}(i\delta)\right)
\end{aligned} \tag{B7}$$

where $\tilde{m}(0)$ denotes the first component of $\mathbf{m}(0)$. □

Proof of Proposition 3. The result follows from Proposition 2 via block-independent stacking of the dynamical system and additive composition of the measurement equation. □

Proof of Proposition 4. It is well known that the set of real block-diagonal matrices whose blocks are either 1×1 real scalars or 2×2 real blocks associated with complex conjugate eigenvalue pairs is dense in the space of real matrices [60, Corollary 3.4.1.10]. Consequently, any real matrix A_r, which cannot be transformed into this block-diagonal form, can be approximated arbitrarily well by such a block-diagonal matrix via an arbitrarily small perturbation. Therefore, the additive composition of $n_s/2$ two-dimensional systems of type (14) can approximate any impulse response from $\mathbf{u}$ to y of stable, finite-dimensional LTI systems. Indeed, consider the transfer function $G(s)$ of an arbitrary stable finite-dimensional LTI system, then

- If $G(s)$ includes poles with multiplicity one, let n_1 be the number of real-poles and n_2 be the number of complex-conjugate poles. We can then build a $n_s/2$ two-dimensional systems of type (14) (with $n_s = 2n_2 + 2n_1$) whose transfer function (from $\mathbf{u}$ to y) is equal to $G(s)$. To do that, it is enough to build the diagonal canonical form for the following transfer function

$$G(s)\prod_{i=1}^{n_1}(s-p_i),$$

where p_i are the real-poles. In other words, since the additive composition of $n_s/2$ two-dimensional systems of type (14) can only have real-poles with multiplicity two, we cancel them with zeros.

- If $G(s)$ includes poles with multiplicity greater than one, we first perturb them by a small perturbation to reduce their multiplicity to one and then we apply the approach described above. □

Proof of Proposition 5. Consider the kernel:

$$S_\theta(t,t') = -e^{\lambda^\dagger\tau}\left(\frac{cLL^Hc^\dagger}{\lambda+\lambda^\dagger} + \frac{c^\dagger L^\dagger L^H c^\dagger}{2\lambda^\dagger}\right) - e^{\lambda\tau}\left(\frac{cLL^\top c}{2\lambda} + \frac{c^\dagger L^\dagger L^\top c}{\lambda+\lambda^\dagger}\right), \tag{B8}$$

and observe that for $c = c^\dagger = c$, it reduces to

$$S_\theta(t,t') = -c^2e^{\lambda^\dagger\tau}\left(\frac{LL^H}{\lambda+\lambda^\dagger} + \frac{L^\dagger L^H}{2\lambda^\dagger}\right) - c^2e^{\lambda\tau}\left(\frac{LL^\top}{2\lambda} + \frac{L^\dagger L^\top}{\lambda+\lambda^\dagger}\right). \tag{B9}$$

Since $L\in\mathbb{C}$, then

$$\begin{aligned}
S_\theta(t,t') = -\frac{c^2e^{\Re(\lambda)\tau}}{\Re(\lambda)(\Im(\lambda)^2+\Re(\lambda)^2)}&\left((2\Im(\lambda)\Im(L)\Re(\lambda)\Re(L)\right.\\
&+2\Re(\lambda)^2\Re(L)^2+\Im(\lambda)^2(\Im(L)^2+\Re(L)^2))\cos[\Im(\lambda)\tau]\\
&+\Re(\lambda)(-2\Im(L)\Re(\lambda)\Re(L)+\Im(\lambda)(-\Im(L)^2\\
&\left.+\Re(L)^2))\sin[\Im(\lambda)\tau]\right).
\end{aligned}$$

For

$$\Re(L) = \frac{\Im(L)\Re(\lambda) + \sqrt{\Im(\lambda)^2\Im(L)^2 + \Im(L)^2\Re(\lambda)^2}}{\Im(\lambda)}$$

the term that multiplies $\sin[\Im(\lambda)\tau]$ is equal to zero. Therefore, we have that

$$\begin{aligned}
S_\theta(t,t') = &-2c^2e^{\Re(\lambda)\tau}\cos[\Im(\lambda)\tau]\\
&\times\frac{\Im(L)^2\left(\Im(\lambda)^2+2\Re(\lambda)\left(\Re(\lambda)+\sqrt{\Im(\lambda)^2+\Re(\lambda)^2}\right)\right)}{\Im(\lambda)^2\Re(\lambda)}.
\end{aligned} \tag{B10}$$

Therefore, we set

$$\Im(L) = \sqrt{-\frac{1}{2}\frac{\Im(\lambda)^2\Re(\lambda)}{\Im(\lambda)^2+2\Re(\lambda)^2+\Re(\lambda)\sqrt{\Im(\lambda)^2+\Re(\lambda)^2}}},$$

which is a real number as $\Re(\lambda)<0$. Therefore, we have that

$$S_\theta(t,t') = c^2e^{\Re(\lambda)\tau}\cos[\Im(\lambda)\tau], \tag{B11}$$

and with $ns/2$ components,

$$S_\theta(t,t') = \sum_{i=1}^{n_s/2}c_i^2e^{\Re(\lambda_i)\tau}\cos[\Im(\lambda_i)\tau]. \tag{B12}$$

□

Proof of Proposition 6. The stochastic variational inference developed for the Wiener architecture can be extended to any general architecture. First, we notice that multiple GPs per layer (of the same type, static or dynamic), can be represented as a single joint GP (static or dynamic) with independent components. Therefore, without loss of generality, we can assume that there is only one GP per layer. Furthermore, without loss of generality, we can also assume we have L layers, with the odd-numbered ones being dynamic and the even-numbered ones static. Then, the joint can be written as

$$\begin{aligned}
&p(\mathbf{y},\{\mathbf{f},\Upsilon_f\}_{l=2,4,\dots,L},\{\mathbf{g},\Upsilon_g\}_{l=1,3,\dots,L}|\mathbf{Z}_f,\mathbf{Z}_g,\mathbf{u})\\
&= \underbrace{p(\mathbf{y}|\mathbf{f})}_{\text{likelihood}}\prod_{l=2,4,\dots,L}\underbrace{p(\mathbf{f}^l|\Upsilon_f^l,Z_f^l,\mathbf{g}^{l-1})p(\Upsilon_f^l|Z_f^l)}_{\text{GP for static block}}\\
&\times\underbrace{p(\mathbf{g}^{l-1}|\Upsilon_g^{l-1},Z_g^{l-1},\mathbf{f}^{l-2})p(\Upsilon_g^{l-1}|Z_g^{l-1})}_{\text{GP for dynamic block}}
\end{aligned} \tag{B13}$$

10991115, 0, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/acs.70141 by Zhaw Zurcher Hochschule Fur, Wiley Online Library on [12/09/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

where Z_f^l, Υ_f^l are the vector of inducing points for the static GP in the layer l, $\mathbf{f}^l = [f^l(g_\delta^{l-1})), \ldots, f^l(g_{n\delta}^{l-1}]^\top$, $Z_g^{l-1}, \Upsilon_g^{l-1}$ are the vector of inducing points for the dynamic GP in the layer $l-1$, $\mathbf{g}^{l-1} = [g_\delta^{l-1}(f_\delta^{l-2}), \ldots, g_{n\delta}^{l-1}(f_{n\delta}^{l-2})]^\top$, and $\mathbf{f}^0 = \mathbf{u} = [\mathbf{u}(\delta), \ldots, \mathbf{u}(n\delta))]^\top$. The likelihood term is in (31). The second and third terms are

$$\begin{aligned} p(\mathbf{f}^l|\Upsilon_f^l, Z_f^l, \mathbf{g}^{l-1}) &= N(\mathbf{f}^l; Q_f^l \Upsilon_f^l, K_{\theta^l}(\mathbf{g}^{l-1}, \mathbf{g}^{l-1}) \\ &\quad - Q_f^l K_{\theta^l}(Z_f^l, Z_f^l) Q_f^{l\top}), \\ p(\Upsilon_f^l|Z_f^l) &= N(\Upsilon_f^l; \mathbf{0}, K_{\theta^l}(Z_f^l, Z_f^l)), \end{aligned} \tag{B14}$$

with $Q_f^l = K_{\theta^l}(\mathbf{g}^{l-1}, Z_f^l) K_{\theta^l}^{-1}(Z_f^l, Z_f^l)$. The last two terms are

$$\begin{aligned} p(\mathbf{g}^{l-1}|\Upsilon_g^{l-1}, Z_g^{l-1}) &= N(\mathbf{g}^{l-1}; Q_g^{l-1}\Upsilon_g^{l-1}, S_{\theta^{l-1}}(T,T) \\ &\quad - Q_g^{l-1} S_{\theta^{l-1}}(Z_g^{l-1}, Z_g^{l-1}) Q_g^{l-1\top}), \\ p(\Upsilon_g^{l-1}|Z_g^{l-1}, \mathbf{f}^{l-2}) &= N(\Upsilon_g^{l-1}; m_{\theta^{l-1}}^{l-1}(\mathbf{f}^{l-2}), S_{\theta^{l-1}}(Z_g^{l-1}, Z_g^{l-1})), \end{aligned} \tag{B15}$$

with $Q_g^{l-1} = S_{\theta^{l-1}}(T, Z_g^{l-1}) S_{\theta^{l-1}}^{-1}(Z_g^{l-1}, Z_g^{l-1})$. We use the following variational distribution to approximate the posterior:

$$\begin{aligned} &q(\{\mathbf{f}, \Upsilon_f\}_{l=2,4,\ldots,L}, \{\mathbf{g}, \Upsilon_g\}_{l=1,3,\ldots,L} | \mathbf{Z}_f, \mathbf{Z}_g, \mathbf{u}) \\ &= \prod_{l=2,4,\ldots,L} p(\mathbf{f}^l|\Upsilon_f^l, Z_f^l, \mathbf{g}^{l-1}) q(\Upsilon_f^l) \\ &\quad \times p(\mathbf{g}^{l-1}|\Upsilon_g^{l-1}, Z_g^{l-1}, \mathbf{f}^{l-2}) p(\Upsilon_g^{l-1}) \end{aligned} \tag{B16}$$

where the two variational distributions $q(\Upsilon_f^l), q(\Upsilon_g^{l-1})$ are assumed to be Gaussian and their mean and covariance matrix being the variational parameters for each layer. We can then derive a lower bound for the evidence similarly to (38):

$$\begin{aligned} &\log p(\mathbf{y}) \\ &\geq \int \log(p(\mathbf{y}|\mathbf{f}^L)) \prod_{l=2,4,\ldots,L} p(\mathbf{f}^l|\Upsilon_f^l, Z_f^l, \mathbf{g}^{l-1}) q(\Upsilon_f^l) p(\mathbf{g}^{l-1}|\Upsilon_g^{l-1}, \\ &\quad \times Z_g^{l-1}, \mathbf{f}^{l-2}) q(\Upsilon_g^{l-1}) d\mathbf{f}^{2:2:L} d\Upsilon_f^{2:2:L} d\mathbf{g}^{1:2:L} d\Upsilon_g^{1:2:L} \\ &\quad - \sum_{l=2,4,\ldots,L} KL[q(\Upsilon_f^l)||p(\Upsilon_f^l|Z_f^l)] \\ &\quad - \sum_{l=1,3,\ldots,L} KL[q(\Upsilon_g^l)||p(\Upsilon_g^l|Z_g^l, \mathbf{f}^{l-1})] \end{aligned} \tag{B17}$$

where $\mathbf{f}^{2:2:L} = [\mathbf{f}^2, \mathbf{f}^4, \ldots, \mathbf{f}^L]$, and similarly for the others. □

Appendix C: Additional Figures for Example 2

We fitted the dataset in Figure 2 using dynoGP with a Wiener architecture with different dimensions in the LIN block. The dimensions n_s are 2, 6, 10, 20. The posterior predictions: mean (blue line) and the 99.7% credible interval (lighter blue-region) are shown in Figure C1. It is evident that the model needs a $n_s \geq 3$ to fit the data well. Increasing the dimensions n_s does not change the posterior. This is also evident from this table:

n_s	ELBO	RMSE test set
2	0.17	0.058
6	0.22	0.015
10	0.21	0.015
20	0.22	0.016

which shows the value of the evidence-lower-bound (ELBO) that we maximise when fitting the parameters in dynoGP, along with the RMSE on the test set. The ELBO increases from $n_s = 1$ to $n_s = 3$ and then remains constant for $n_s \geq 6$. Similarly, the RMSE on the test set remains constant for $n_s \geq 6$. This indicates that if we run dynoGP with a sufficiently large n_s, the performance remains stable. The small differences in the ELBO (and RMSE) are due to the mini-batches based training which generates a small of variability in the final value of the ELBO (and RMSE). The dimension of the LTI component of the system we used to generate the data is 10. However, since the parameters of the system are randomly generated, the effective dimension of the system may in practice be lower (due to fast dynamics that contribute only marginally to the response function). This is the reason why DynoGP fits it well already with $n_s = 6$.

Appendix D: Implementation

We implemented *dynoGP* in *GPyTorch* [53], and the implementation is available at https://github.com/benavoli/dynoDeepGP. The main difference compared to a standard DeepGP is that the dynamic GP layer has a covariate time, regardless of its location within the deep network. This also implies that the inducing points for dynamic layers are scalar and represent time. Since time is ordered and the inputs of the dynamic layers are functions of time, it must be carefully managed in the implementation. For this reason, we do not optimise the inducing point locations for dynamic GP layers but only for static layers. We set the inducing points for the dynamic layer as follows. Let $T = [t_1, t_2, \ldots, t_n]$ be the time points of the observations. The inducing points are randomly selected from T without resampling, based on a categorical distribution with unnormalised probabilities $p_i \propto \log(1 + t_i/t_n)$. This ensures that time instants closer to t_n have a higher probability of being sampled, which improves accuracy in forecasting future values.

We used the following number of inducing points for the dynamic layer and static layer in the numerical experiments: identification of a simulated Wiener system (800 for the dynamic layer and 200 for the static layer); identification of the WH benchmark (1000 for the dynamic layers and 300 for the static layer); identification the coupled-electric drives benchmark (250 for the dynamic layers and 175 for the static layer); forecast of electricity demand (700 for the dynamic layers and 200 for the static layer).

We use uniform priors over all hyperparameters except for the matrices L_j and $L_j^\dagger$ in the LTI system. For these, we set a horseshoe prior [61] with a scale parameter of 0.1. The horseshoe prior is introduced to facilitate model selection between a deterministic and a stochastic LTI system in each of the dynamic GP layers. Its flat, Cauchy-like tails allow L to take large values a posteriori, while its infinitely tall spike at the origin enforces strong shrinkage for elements of L that are close to zero.

Hyperparameters Selection

GP-NARX is a GP model that uses past inputs and measured outputs as covariates, with input and output lags of order n_{lags}. We fit GP-NARX using an automatic relevance determination (ARD) squared-exponential kernel. Therefore, we only need to choose a sufficiently large n_{lags}. Even if this value exceeds the memory of the true underlying nonlinear system, GP-NARX can discard the unnecessary covariates (past lagged inputs and outputs) via the ARD kernel. In the experiments, we increased n_{lags} until the marginal likelihood of GP-NARX stopped getting better.

SB-MLP and SB-LSTM only work for SISO dynamical systems. For the architectures selection, we followed [19]. For SB-MLP, we used two hidden layers with 50 neurons each. For SB-LSTM, we used a single LSTM layer with 10 units. The lag was set to 10.

For dynoGP, we used a LIN-NONLIN-LIN architecture with varying state dimensions. We tried three values of the dimension n_s of the linear layers (4, 10, 20) and selected the dimension with better evidence lower bound in the training data. For the selection of the number of LTI blocks in the first LIN layer, we tried different values 1, 2, 3, ... until the evidence lower bound optimised during training stopped improving significantly compared to the previous value. We kept the number of LIN blocks in the last layer fixed to one. We used the same architecture for dynoNet.

Computational Time

Table C1 shows the per-batch computation time for dynoGP on an NVIDIA GeForce RTX 4060 GPU for the four problems discussed

10991115, 0, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/acs.70141 by Zhaw Zurcher Hochschule Fur, Wiley Online Library on [12/09/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

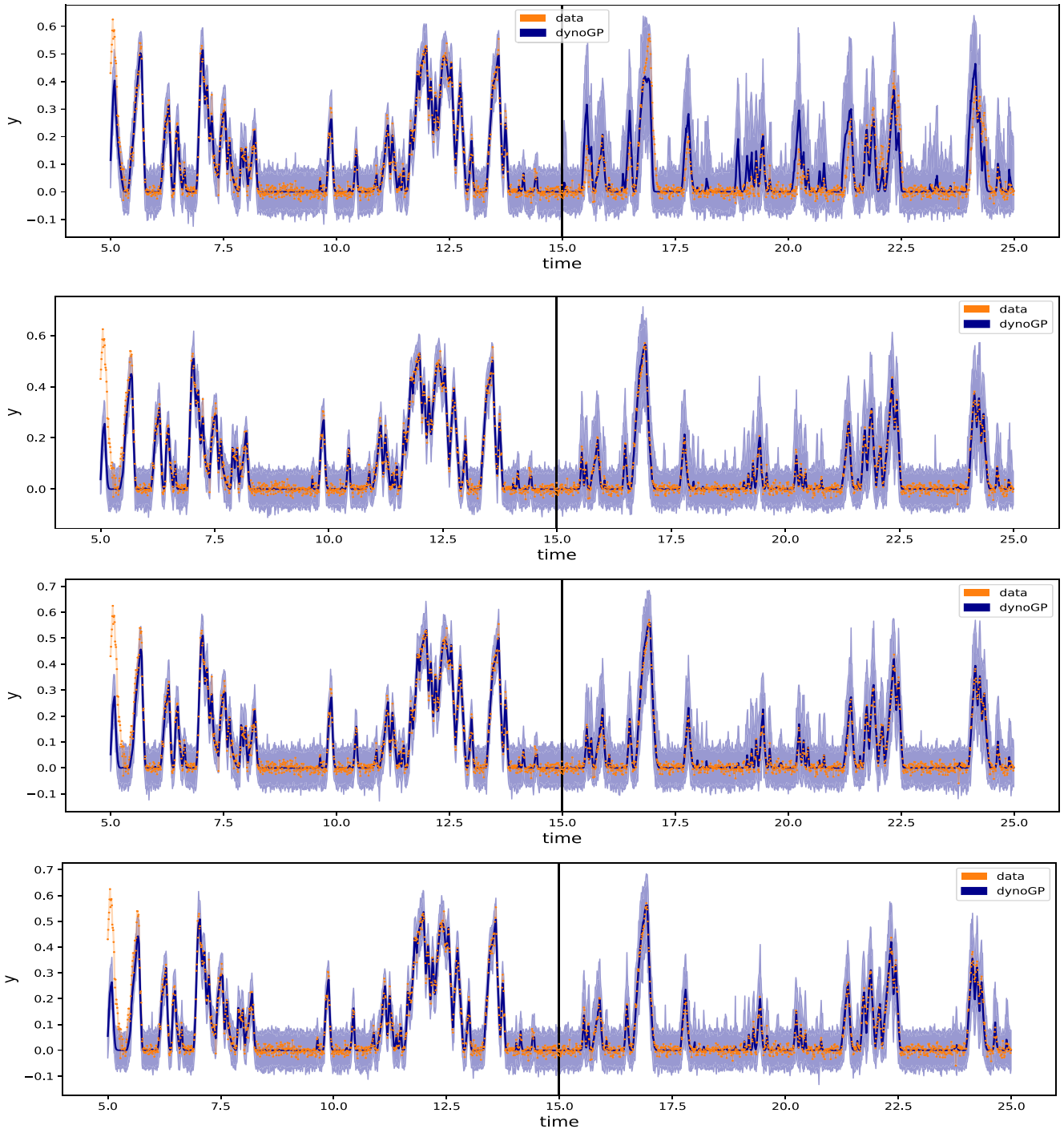


FIGURE C1 | DynoGP with a Wiener architecture with different dimensions in the LIN block. The dimensions n_s are 2, 6, 10, 20 from top to bottom.

previously. The differences in time are primarily due to the varying complexity of the architectures (number of linear and static blocks and number of inputs).

The number of training iterations through the dataset and the number of mini-batches per iteration are: simulated Wiener system (2000, 4); identification of the WH benchmark (1200, 8); identification the coupled-electric drives benchmark (2000, 1); forecast of electricity demand (500, 25). This enables the calculation of the overall training time.

The mini-batch implementation of dynoGP allows it to scale to thousands of observations. However, dynoGP is significantly slower than GP-NARX, SB-MLP, and SB-LSTM, since it relies on a variational approximation of the posterior, which is slower for instance than the Laplace

TABLE C1 | Computational Time Per Mini-Batch.

Datasets	Mini-batch size	Time (s)
Wiener	375	0.7
WH	1024	3
CED	400	1
ELE	1053	6

10991115, 0, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/acs.70141 by Zhaw Zurcher Hochschule Fur, Wiley Online Library on [12/09/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

approximation employed by SB-MLP and SB-LSTM. For instance, in the CED case, the total computational time for dynoGP is $1 \cdot 1 \cdot 2000 \approx 2000$ s. For SB-MLP and dynoNet, it is around 10 s. For SB-LSTM, it is approximatively 450 s. For GP-NARX, it is approximatively 100 s.

Nevertheless, training for dynoGP can be performed off-line, after which prediction can be computed fastly. There are also faster ways to train a DeepGP [62, 63], which we will explore in future work.

10991115, 0, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/acs.70141 by Zhaw Zurcher Hochschule Fur, Wiley Online Library on [12/09/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License