Physica D 462 (2024) 134146

Contents lists available at ScienceDirect

Physica D

journal homepage: www.elsevier.com/locate/physd

Likelihood-based generalization of Markov parameter estimation and multiple shooting objectives in system identification

Nicholas Galioto *, Alex Arkady Gorodetsky

Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI 48109, USA

ARTICLE INFO

Communicated by R. Kuske

Keywords:
System identification
Measures of model fit
Robust estimation
Identification methods
Model fitting

ABSTRACT

This paper considers system identification (ID) of linear and nonlinear non-autonomous systems from noisy and sparse data. We analyze an optimization objective derived from Bayesian inference for the dynamics of hidden Markov models. We then relate this objective to that used in several state-of-the-art approaches for both linear and nonlinear system ID. In the former, we analyze least squares approaches for Markov parameter estimation, and in the latter, we analyze the multiple shooting approach. We demonstrate the limitations of the optimization problems posed by these existing methods by showing that they can be seen as special cases of the proposed optimization objective under certain simplifying assumptions: conditional independence of data and zero model error. Furthermore, we observe that the proposed approach has improved smoothness and inherent regularization that make it well-suited for system ID and provide mathematical explanations for these characteristics' origins. Finally, numerical simulations demonstrate a mean squared error over 8.7 times lower compared to multiple shooting when data are noisy and/or sparse. Moreover, the proposed approach identifies accurate and generalizable models even when there are more parameters than data or when the system exhibits chaotic behavior.

1. Introduction

Learning accurate and generalizable models of dynamical systems is of interest in a wide variety of research areas. In a previous work [1], we analyzed a Bayesian system identification (ID) algorithm that uses a hidden Markov model (HMM) with stochastic dynamics to model the underlying system. Often, the process noise in a stochastic model represents external disturbances to the system's nominal dynamics, but we showed that it can be used as a measure of model uncertainty that is often missing in system ID. Based on the promising results achieved in that work, we seek to extend the applicability and further investigate the capabilities of the Bayesian algorithm.

This work extends the applicability of the Bayesian method from [1] in two main ways. The first is improved expressivity of the parameterized model classes. In [1], the most expressive dynamics model was a polynomial expansion with 20 unknown parameters, but in this manuscript, we seek to train neural networks with hundreds of parameters. The greater expressivity of neural networks allows for estimation of a wider range of systems without requiring intensive modeling, but it also introduces more complicated loss landscapes and intensifies the challenge of generalization, especially for small and noisy training datasets.

The second point of extension is the type of systems considered. The examples and analysis in [1] were of autonomous systems, but non-autonomous systems are of interest to many scientific and engineering fields as well. Non-autonomous systems arise not only in controls-based applications [2] but also more broadly in applications such as modeling the spread of infectious diseases [3] and changes in lake temperatures [4]. Applying external inputs to a system introduces an added layer of complexity because estimation methods must distinguish between the effects of the system dynamics and those of the external forcing. Indeed, extensions to control systems have been a topic of interest for methods originally designed for autonomous systems [5,6]. In this paper, we consider non-autonomous systems and provide new theory and comparisons to prevalent system ID approaches within this class of systems.

Additionally, this work investigates the capability of the Bayesian algorithm in addressing the following critical challenges in system ID: (1) dealing with error accumulation in the objective that leads to optimization difficulties, (2) finding generalizable models with few data, and (3) deriving a measure of model fit that is robust to model, measurement, and parameter uncertainty. Although our primary focus is on nonlinear systems, we illustrate these challenges in both linear and nonlinear system ID. We include linear system ID because its

* Corresponding author.
E-mail addresses: ngalioto@umich.edu (N. Galioto), goroda@umich.edu (A.A. Gorodetsky).

https://doi.org/10.1016/j.physd.2024.134146
Received 27 July 2023; Received in revised form 23 February 2024; Accepted 20 March 2024
Available online 24 March 2024

methodologies are often applied to nonlinear systems for purposes such as creating inexpensive surrogate models or studying local behavior, e.g., equilibrium point stability. For linear problems we consider estimation of Markov parameters needed for the eigensystem realization algorithm (ERA) [7]. For nonlinear problems, we consider multiple shooting (MS) objectives [8]. We demonstrate that these approaches are special cases derived from, often limiting, assumptions within the likelihood of the HMM model. Our specific contributions include:

(1) establishment that many Markov parameter estimation methods implicitly assume that the data are conditionally independent given the initial condition, inputs, and Markov parameters (Proposition 2),
(2) illustration of how the variance parameters in the Bayesian posterior affect the smoothness of the optimization surface similarly to the time horizon parameter in MS (Section 3.2.4),
(3) empirical demonstration of significant performance gains by using the Bayesian posterior compared to these other approaches (Section 4).

To support these contributions, we also

- extend the Bayesian algorithm used in [1] to nonlinear, non-autonomous systems using highly-expressive neural network parameterizations,
- demonstrate that the implicit assumption of conditional independence leads to worse estimates (Sections 3.1.2 and 4.2),
- exemplify how state-of-the-art machine learning methods can be improved by rigorous modeling of uncertainty when training on small and noisy datasets (Section 4.3),
- illustrate the utility of accounting for model errors when estimating chaotic systems and specifically that the probabilistic modeling of model uncertainty used in the Bayesian method is more effective than the *ad hoc* approach of the MS method (Section 4.4),
- establish that least squares-based metrics can assign unfavorable rankings to strong model estimates identified by the Bayesian posterior (Section 4.4),
- identify how regularization arises in the marginal likelihood (Eq. (5)) and empirically demonstrate that the Bayesian posterior consequently resists overfitting (Section 4.5).

1.1. Related work

Next, we provide a literature review on works that have sought to address the challenges of error accumulation, generalizability, and uncertainty modeling. The focus of this review will be on approaches similar to those of the Bayesian algorithm.

We begin the review with work that addresses error accumulation in the objective function. The formation of local extrema due to error accumulation is a well-known problem in system ID [9] and there exist various heuristic approaches for addressing it by discounting the data in one form or another. One example is simulated annealing [10], which smooths the posterior by scaling down the influence of the likelihood by a discount factor. As optimization progresses, the likelihood is gradually scaled up to its full weight according to a "cooling schedule." This algorithm can be difficult to use in practice since its effectiveness strongly depends on the cooling schedule, which must be chosen by the user and is largely problem-dependent. Furthermore, the algorithm addresses only the effects of error accumulation and not the underlying cause, and as a result, the posterior is still filled with local minima. A variation known as data annealing [11] starts with training data from a short time period and introduces more data over a gradually-increasing time period. This has a similar "cooling" effect of incrementally increasing the influence of the likelihood on the posterior. Using data from a shorter training period at the early iterations can prevent the large error accumulation associated with long simulation times, but it still lacks a mechanism for handling error accumulation once the full dataset has been introduced.

Other algorithms [12] directly address error accumulation by simulating the system output at many different initial times and using only the data within a specified time horizon of the initial times to evaluate the fit of each trajectory. An example of an extreme case of this is standard and exact DMD [13]. In these algorithms, the Koopman operator is estimated by finding the best linear mapping of the data forward only one step in time, leading to a convex optimization problem. Algorithms that use a time horizon greater than one fall generally under the category of MS. In such algorithms, the selected time horizon usually depends on the approximate time scale of the system. The main issue with these algorithms is that they do not allow for flexibility in the case that the state components have different time scales. In contrast, the Bayesian algorithm studied in this paper can account for error accumulated at different rates within the state by estimating a process noise covariance matrix. The covariance matrix has the additional advantages of being continuous to allow for greater precision and being able to be tuned automatically during optimization.

The third challenge of handling model, measurement, and parameter uncertainty was discussed in depth in a previous work [1]. The traditional Bayesian system ID approach [14] models measurement uncertainty with a noise term in the system output and models parameter uncertainty as a posterior distribution over the parameters. In [1], we improved upon this framework by introducing a process noise term in the dynamics model as a measure of epistemic model uncertainty. We showed how this new posterior could be evaluated with an existing algorithm and provided an analysis of the algorithm's computational complexity. We also showed how fundamental system ID objectives can be seen as limiting cases of the negative log posterior as either model or measurement uncertainty approach zero, thereby offering a new perspective on the optimality of system ID objective functions.

1.2. Paper outline

The rest of the paper is organized as follows. Section 2 details the probabilistic formulation of the dynamics model and provides the algorithm for computing the unnormalized log posterior. In Section 3, background and analysis is provided on Markov parameter estimation methods and MS. Section 4 presents the results of four numerical experiments. Section 5 discusses the limitations and potential future extensions of the Bayesian approach. Lastly, conclusions are given in Section 6.

2. Background

In this section, we provide notation, the HMM framework, the recursive algorithm for efficiently evaluating the likelihood, and a representative example from a linear system.

2.1. Notation

The following notation is used in the paper. Matrices are represented with uppercase and bold font $\mathbf{A}$ and vectors with lowercase and bold font $\mathbf{x}$. Matrices and vectors are indexed with square brackets, e.g., the (i,j)-th element of $\mathbf{A}$ is denoted $\mathbf{A}[i,j]$. The norm of a vector $\mathbf{x}$ weighted by a positive definite matrix $\mathbf{W}$ is defined as $\|\mathbf{x}\|_{\mathbf{W}}^2 := \mathbf{x}^*\mathbf{W}^{-1}\mathbf{x}$, where * denotes the transpose. The 2-norm of a vector is denoted as $\|\cdot\|_2$, and its induced matrix norm is defined as $\|\mathbf{A}\|_2 := \sup_{\mathbf{x}\neq\mathbf{0}} \|\mathbf{A}\mathbf{x}\|_2/\|\mathbf{x}\|_2$. To denote an element-wise absolute value, we use $|\cdot|$. The notation $\mathcal{N}(\mathbf{m},\mathbf{P})$ denotes a normal distribution with mean $\mathbf{m}$ and covariance $\mathbf{P}$. If $\mathbf{y} = |\mathbf{x}|$ and $\mathbf{x}$ is distributed as $\mathcal{N}(\mathbf{0},\mathbf{P})$, then $\mathbf{y}$ follows a half-normal distribution denoted as half–$\mathcal{N}(\mathbf{0},\mathbf{P})$. If $\mathbf{y} = \log\mathbf{x}$ and $\mathbf{y}$ is distributed as $\mathcal{N}(\mathbf{m},\mathbf{P})$, then $\mathbf{x}$ follows a log-normal distribution denoted as log–$\mathcal{N}(\mathbf{m},\mathbf{P})$.

2.2. Hidden Markov model

Consider the parameterized HMM [15] approximation of an unknown dynamical system

$$\mathbf{x}_{k+1} = \Psi(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\theta}_\Psi) + \boldsymbol{\xi}_k, \qquad \boldsymbol{\xi}_k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}(\boldsymbol{\theta}_\Sigma)), \tag{1a}$$

$$\mathbf{y}_k = h(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\theta}_h) + \boldsymbol{\eta}_k, \qquad \boldsymbol{\eta}_k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Gamma}(\boldsymbol{\theta}_\Gamma)), \tag{1b}$$

with uncertain initial condition $\mathbf{x}_0(\boldsymbol{\theta}_{\mathbf{x}_0})$. The states are denoted by $\mathbf{x}_k \in \mathbb{R}^{d_x}$, the measurements by $\mathbf{y}_k \in \mathbb{R}^{d_y}$, and the inputs by $\mathbf{u}_k \in \mathbb{R}^{d_u}$. The subscript $k \in \mathbb{Z}_+ \cup \{0\}$ is an index corresponding to time $t_k \in [0, \infty)$. The function $\Psi : \mathbb{R}^{d_x} \times \mathbb{R}^{d_u} \times \mathbb{R}^{d_\theta} \mapsto \mathbb{R}^{d_x}$ models the dynamics of the hidden state, and $h : \mathbb{R}^{d_x} \times \mathbb{R}^{d_u} \times \mathbb{R}^{d_\theta} \mapsto \mathbb{R}^{d_y}$ is an observation function that maps the hidden space to the observable space. The additive, zero-mean Gaussian noise in the dynamics $\boldsymbol{\xi}_k \in \mathbb{R}^{d_x}$ and in the observations $\boldsymbol{\eta}_k \in \mathbb{R}^{d_y}$ are independent and identically distributed realizations of the process and measurement uncertainty, respectively. The uncertain parameters $\boldsymbol{\theta} = \begin{bmatrix}\boldsymbol{\theta}^*_{\mathbf{x}_0} & \boldsymbol{\theta}^*_\Psi & \boldsymbol{\theta}^*_h & \boldsymbol{\theta}^*_\Sigma & \boldsymbol{\theta}^*_\Gamma\end{bmatrix}^* \in \mathbb{R}^{d_\theta}$ correspond to the initial condition, dynamics, observation function, process noise covariance $\boldsymbol{\Sigma} \in \mathbb{R}^{d_x \times d_x}$, and measurement noise covariance $\boldsymbol{\Gamma} \in \mathbb{R}^{d_y \times d_y}$, respectively. Rather than include $\boldsymbol{\Sigma}$ and $\boldsymbol{\Gamma}$ as parameters, we write them as functions of $\boldsymbol{\theta}$ for generality. This is useful when encoding matrix structure or constraining parameter values.

Since uncertainty modeling is a key aspect of this formulation, it warrants a brief discussion. The process noise term $\boldsymbol{\xi}_k$ represents the model uncertainty. This term is intended to capture modeling error in Ψ that arises regardless of parameter values such as numerical inaccuracies, e.g., from a numerical integrator, or insufficient model expressiveness. The measurement noise term $\boldsymbol{\eta}_k$ represents the measurement uncertainty. This term is included to capture not only the modeling error in h but also the random sensor noise corrupting the observations. The choice to model both the process and measurement noise terms as Gaussians is informed by the Principle of Maximum Entropy [16]. The final source of uncertainty in system ID arises from the parameters. We describe how we account for the parameter uncertainty in Section 2.3.

Despite their similarities, the impacts of the model and measurement uncertainties on the system ID problem are fundamentally different. The measurement noise terms do not affect any other variables in the HMM, making them the simplest errors to account for in estimation. The process noise terms, on the other hand, are propagated through the system dynamics and enter into all downstream states. To account for model errors requires tracking how this uncertainty changes over time. Although this necessitates greater computational expense, the time-varying uncertainty provides additional information on the output estimate that can be incorporated into the objective function for improved estimation. We discuss how to leverage this information in Section 2.4.1.

2.3. Bayesian learning

Next, we describe a Bayesian approach to system ID. This is the same approach used in [1].

The goal of Bayesian system ID is to characterize the posterior probability distribution of the parameters $\boldsymbol{\theta}$ after collecting $n+1$ data points $\mathbf{y}_0, \dots, \mathbf{y}_n$. This posterior distribution is denoted by $p(\boldsymbol{\theta}|\mathcal{Y}_n)$, where p is a probability density function (pdf), and $\mathcal{Y}_n := (\mathbf{y}_0, \dots, \mathbf{y}_n)$ is the collection of data points. Bayes' rule represents this posterior distribution in a computable form

$$p(\boldsymbol{\theta}|\mathcal{Y}_n) = \frac{\mathcal{L}(\boldsymbol{\theta}; \mathcal{Y}_n) p(\boldsymbol{\theta})}{p(\mathcal{Y}_n)}, \tag{2}$$

where $\mathcal{L}(\boldsymbol{\theta}; \mathcal{Y}_n) := p(\mathcal{Y}_n|\boldsymbol{\theta})$ is the likelihood, $p(\boldsymbol{\theta})$ is the prior, and $p(\mathcal{Y}_n)$ is the evidence. The main computational challenge in evaluating the posterior is computing the likelihood. In this case, the uncertainty in the states further increases the computational difficulty by inducing the joint likelihood $p(\boldsymbol{\theta}, \mathcal{X}_n|\mathcal{Y}_n)$, where $\mathcal{X}_n := (\mathbf{x}_0, \dots, \mathbf{x}_n)$, rather than the target marginal likelihood $\mathcal{L}(\boldsymbol{\theta}; \mathcal{Y}_n)$. Obtaining this marginal likelihood requires the high-dimensional integration $\mathcal{L}(\boldsymbol{\theta}; \mathcal{Y}_n) = \int \mathcal{L}(\boldsymbol{\theta}, \mathcal{X}_n; \mathcal{Y}_n) \mathrm{d}\mathcal{X}_n$. Specifically, this is an integral over the d_x-dimensional state at each of the $n+1$ time instances. Typically, high-dimensional integration is intractable, but this integral can be computed efficiently though recursion. The recursive procedure begins by factorizing the marginal likelihood as

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{Y}_n) = p(\mathbf{y}_0 \mid \boldsymbol{\theta}) \prod_{k=1}^{n} p(\mathbf{y}_k \mid \mathcal{Y}_{k-1}, \boldsymbol{\theta}). \tag{3}$$

Then, Algorithm 1 provides a recursive approach [17] to evaluate each of the terms in the product.[1] If the system is linear-Gaussian, the marginal likelihood $\mathcal{L}(\boldsymbol{\theta}; \mathcal{Y}_k)$, prediction distribution $p(\mathbf{x}_{k+1}|\mathcal{Y}_k, \boldsymbol{\theta})$, and update distribution $p(\mathbf{x}_k|\mathcal{Y}_k, \boldsymbol{\theta})$ are all Gaussian, and the Kalman filter provides a closed-form solution. If the system is not linear-Gaussian, other Bayesian filters must be used. For example, in this and past works [1], the unscented Kalman filter [18] is used.

Algorithm 1 Recursive marginal likelihood evaluation [17]

Require: $p(\mathbf{x}_0|\boldsymbol{\theta})$, $\mathcal{Y}_n$
Ensure: $\mathcal{L}(\boldsymbol{\theta}; \mathcal{Y}_n)$
1: Initialize $p(\mathbf{x}_0|\mathcal{Y}_{-1}, \boldsymbol{\theta}) := p(\mathbf{x}_0|\boldsymbol{\theta})$ and $\mathcal{L}(\boldsymbol{\theta}; \mathcal{Y}_{-1}) := 1$
2: **for** $k = 0, \dots n$ **do**
3: Marginalize:

$$p(\mathbf{y}_k|\mathcal{Y}_{k-1}, \boldsymbol{\theta}) \leftarrow \int p(\mathbf{y}_k|\mathbf{x}_k, \boldsymbol{\theta}) p(\mathbf{x}_k|\mathcal{Y}_{k-1}, \boldsymbol{\theta}) \mathrm{d}\mathbf{x}_k$$

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{Y}_k) \leftarrow \mathcal{L}(\boldsymbol{\theta}; \mathcal{Y}_{k-1}) p(\mathbf{y}_k|\mathcal{Y}_{k-1}, \boldsymbol{\theta})$$

4: **if** $k < n$ **then**
5: Update: $p(\mathbf{x}_k|\mathcal{Y}_k, \boldsymbol{\theta}) \leftarrow \frac{p(\mathbf{y}_k|\mathbf{x}_k, \boldsymbol{\theta})}{p(\mathbf{y}_k|\mathcal{Y}_{k-1}, \boldsymbol{\theta})} p(\mathbf{x}_k|\mathcal{Y}_{k-1}, \boldsymbol{\theta})$
6: Predict: $p(\mathbf{x}_{k+1}|\mathcal{Y}_k, \boldsymbol{\theta}) \leftarrow \int p(\mathbf{x}_{k+1}|\mathbf{x}_k, \boldsymbol{\theta}) p(\mathbf{x}_k|\mathcal{Y}_k, \boldsymbol{\theta}) \mathrm{d}\mathbf{x}_k$
7: **end if**
8: **end for**

In general, the Bayesian posterior is non-Gaussian and not analytically tractable. Consequently, we use a Markov chain Monte Carlo (MCMC) algorithm to sample the posterior. The tuning and convergence details for this approach are given Section 4.

2.4. Linear time-invariant systems

For certain systems, the marginal likelihood is analytically tractable. Here we show the approach in the context of linear time-invariant (LTI) models defined as

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{A}(\boldsymbol{\theta})\mathbf{x}_k + \mathbf{B}(\boldsymbol{\theta})\mathbf{u}_k + \boldsymbol{\xi}_k, \quad \boldsymbol{\xi}_k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}(\boldsymbol{\theta})), \\ \mathbf{y}_k &= \mathbf{H}(\boldsymbol{\theta})\mathbf{x}_k + \mathbf{D}(\boldsymbol{\theta})\mathbf{u}_k + \boldsymbol{\eta}_k, \quad \boldsymbol{\eta}_k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Gamma}(\boldsymbol{\theta})). \end{aligned} \tag{4}$$

Note, if $\mathbf{x}_0$ is either given or Gaussian-distributed, then the equations in Algorithm 1 have closed-form solutions. We now consider two different, but technically equivalent, approaches for evaluating the closed-form marginal likelihood.

2.4.1. State–space approach

The first approach uses the state–space model with a Kalman filter to evaluate the marginal likelihood. Following [17], let $\mathbf{m}_k(\boldsymbol{\theta})$ and $\mathbf{P}_k(\boldsymbol{\theta})$ denote the mean and covariance of the Gaussian distribution $p(\mathbf{x}_k \mid \mathcal{Y}_{k-1}, \boldsymbol{\theta})$, i.e., $p(\mathbf{x}_k \mid \mathcal{Y}_{k-1}, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{m}_k(\boldsymbol{\theta}), \mathbf{P}_k(\boldsymbol{\theta}))$, at time t_k. The value of the mean $\mathbf{m}_k$ and covariance $\mathbf{P}_k$ can be found via a Kalman filter. Then, each term in Eq. (3) becomes $p(\mathbf{y}_k \mid \mathcal{Y}_{k-1}, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\mu}_k(\boldsymbol{\theta}), \mathbf{S}_k(\boldsymbol{\theta}))$, where $\boldsymbol{\mu}_k(\boldsymbol{\theta}) = \mathbf{H}(\boldsymbol{\theta})\mathbf{m}_k(\boldsymbol{\theta}) + \mathbf{D}(\boldsymbol{\theta})\mathbf{u}_k$ and $\mathbf{S}_k(\boldsymbol{\theta}) = \mathbf{H}(\boldsymbol{\theta})\mathbf{P}_k(\boldsymbol{\theta})\mathbf{H}(\boldsymbol{\theta})^* + \boldsymbol{\Gamma}(\boldsymbol{\theta})$.

[1] To avoid numerical underflow, we evaluate Algorithm 1 for the log likelihood.

Defining the residual $\mathbf{r}_k(\theta) = \mathbf{y}_k - \boldsymbol{\mu}_k(\theta)$, the log marginal likelihood from line 3 of Algorithm 1 becomes

$$\log \mathcal{L}(\theta; \mathcal{Y}_n) \propto -\sum_{k=0}^{n} \left(\|\mathbf{r}_k(\theta)\|^2_{\mathbf{S}_k(\theta)} + \log\det\left(\mathbf{S}_k(\theta)\right) + d_y \log(2\pi) \right), \tag{5}$$

where $\det(\cdot)$ is the determinant [17, Th. 12.3].

The form of Eq. (5) resembles a least squares (LS) metric plus a regularization term $\log\det\left(\mathbf{S}_k\right)$. The inclusion of this term differs from the typical approach in which regularization is introduced through a prior distribution or heuristic penalty placed on the parameters (e.g., the L_2 norm used in ridge regression). Here, the regularization term has arisen in the likelihood directly from the probabilistic model of the dynamical system and does not require any assumptions on the parameters. The effect of this additional term is a penalty on systems where the estimated output has a large covariance. A large $\mathbf{S}_k$ can arise when the output is sensitive to perturbations in the state, which is undesirable as it is commonly a sign of overfitting. The regularization term $\log\det\left(\mathbf{S}_k\right)$ encodes this dispreference automatically in the likelihood.

2.4.2. Input–output Markov parameter approach

The other common approach to LTI system ID is to first estimate the Markov parameters and then extract a realization of the system matrices $(\mathbf{A}, \mathbf{B}, \mathbf{H}, \mathbf{D})$ using the ERA. Here we discuss how the Markov parameters can be obtained from data.

The Markov parameters are obtained by rewriting the linear system (4) in a form that removes the states through recursive substitution into the observation equations. Let the Markov parameters be $\mathbf{G}_0 = \mathbf{D}$ and $\mathbf{G}_k = \mathbf{H}\mathbf{A}^{k-1}\mathbf{B}$ for $k = 1, 2, \ldots$, and define a new random variable $\boldsymbol{\nu}_k = \sum_{i=1}^{k} \mathbf{H}\mathbf{A}^{i-1}\boldsymbol{\xi}_{k-i}$. Then we obtain

$$\mathbf{y}_k = \mathbf{H}\mathbf{A}^k\mathbf{x}_0 + \sum_{i=0}^{k} \mathbf{G}_i\mathbf{u}_{k-i} + \boldsymbol{\nu}_k, \quad \boldsymbol{\nu}_k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Lambda}_k), \tag{6}$$

where $\boldsymbol{\Lambda}_k = \sum_{i=1}^{k} \mathbf{H}\mathbf{A}^{i-1}\boldsymbol{\Sigma}(\mathbf{H}\mathbf{A}^{i-1})^* + \boldsymbol{\Gamma}$ if $k > 0$, and $\boldsymbol{\Lambda}_k = \boldsymbol{\Gamma}$ if $k = 0$. The $\boldsymbol{\nu}_k$ are not independent due to their sharing of the process noise $\boldsymbol{\xi}_k$. The covariance $\boldsymbol{\Lambda}_{j,k} := \mathbb{C}\mathrm{ov}[\boldsymbol{\nu}_j, \boldsymbol{\nu}_k]$ is $\boldsymbol{\Lambda}_{j,k} = \sum_{i=1}^{j} \mathbf{H}\mathbf{A}^{i-1}\boldsymbol{\Sigma}(\mathbf{H}\mathbf{A}^{k-j+i-1})^*$, for $0 < j < k$. If $j = 0$ and $j \neq k$, then $\boldsymbol{\Lambda}_{j,k} = \mathbf{0}$. Lastly, if $k < j$, then $\boldsymbol{\Lambda}_{j,k} = \boldsymbol{\Lambda}_{k,j}^*$.

The task then is to learn the Markov parameters for use within the ERA. One can use Bayesian inference again to learn a posterior over the Markov parameters. Assuming $\mathbf{x}_0 = \mathbf{0}$, the likelihood model implied by Eq. (6) is

$$p(\mathcal{Y}_n \mid \mathbf{G}_{0:n}, \mathbf{U}_{0:n}) = \mathcal{N}\left(\mathbf{G}_{0:n}\mathbf{U}_{0:n}, \boldsymbol{\Lambda}\right), \tag{7}$$

where $\mathbf{G}_{0:n} = \begin{bmatrix}\mathbf{G}_0 & \mathbf{G}_1 & \cdots & \mathbf{G}_n\end{bmatrix}$ and

$$\boldsymbol{\Lambda} = \begin{bmatrix} \boldsymbol{\Lambda}_0 & \boldsymbol{\Lambda}_{0,1} & \cdots & \boldsymbol{\Lambda}_{0,n} \\ & \boldsymbol{\Lambda}_1 & \cdots & \vdots \\ & & \ddots & \vdots \\ \mathrm{Sym} & & & \boldsymbol{\Lambda}_n \end{bmatrix}, \quad \mathbf{U}_{0:n} = \begin{bmatrix} \mathbf{u}_0 & \mathbf{u}_1 & \cdots & \mathbf{u}_n \\ \mathbf{0} & \mathbf{u}_0 & \cdots & \mathbf{u}_{n-1} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{u}_0 \end{bmatrix}.$$

The log of this likelihood (7) is equivalent to the state–space log likelihood (5) in the following sense: if we evaluate (7) with matrices $\mathbf{G}_{0:n}$ and $\boldsymbol{\Lambda}$ determined by a set of state–space matrices, the result equals the evaluation of (5) using that same set of state–space matrices.

For comparisons in Section 3.1, it is useful to consider a maximum likelihood estimate (MLE) obtained as:

$$\hat{\mathbf{G}}_{0:n} = \operatorname*{argmin}_{\mathbf{G}_{0:n}} \left\| \mathrm{vec}\left(\mathbf{Y}_{0:n} - \mathbf{G}_{0:n}\mathbf{U}_{0:n}\right) \right\|^2_{\boldsymbol{\Lambda}}, \tag{8}$$

where $\mathrm{vec}(\cdot)$ denotes the vectorization of a matrix, and $\mathbf{Y}_{0:n} = \begin{bmatrix}\mathbf{y}_0 & \mathbf{y}_1 & \cdots & \mathbf{y}_n\end{bmatrix}$. The generalized LS solution is $\mathrm{vec}(\hat{\mathbf{G}}_{0:n}) = (\mathbf{V}^*\boldsymbol{\Lambda}^{-1}\mathbf{V})^\dagger\mathbf{V}^*\boldsymbol{\Lambda}^{-1}\mathrm{vec}(\mathbf{Y}_{0:n})$, where † denotes the pseudo-inverse, and $\mathbf{V}^* := \mathbf{U}_{0:n} \otimes \mathbf{I}_{d_y}$, where $\otimes$ is the Kronecker product.

There are two main issues with the MLE approach: (1) the covariance matrix $\boldsymbol{\Lambda}$ depends on the unknown state–space matrices and is therefore itself unknown, and (2) the Markov parameters are dramatically overparameterized since they are direct functions of the system matrices $(\mathbf{A}, \mathbf{B}, \mathbf{H}, \mathbf{D})$. The usual solutions for the first issue include either removing the weighting and using the standard L_2 norm or estimating a realization of the state–space matrices directly. The overparameterization issue arises when $nd_yd_u > d_x^2 + d_xd_u + d_xd_y + d_yd_u$, as is typical. As a result, the number of data points (nd_y) is typically smaller than the number of unknowns (nd_yd_u), — except in the case where $d_u = 1$ — and the optimum is not unique. Moreover, any optimum found with this approach, including when $d_u = 1$, will necessarily overfit the data when there is noise.

This issue motivates approaches that either use multiple trajectories/rollouts with differing inputs[2] to increase the number of data points, or break a single trajectory into multiple ones to decrease the effective number of Markov parameters. Nevertheless, we will show that these existing works use implicit simplifying assumptions and are still at risk of underdetermination for certain input signals. The Bayesian state–space approach, on the other hand, makes all assumptions explicit and is viable regardless of the type of control inputs. These details are further described in Section 3.1.

3. Theoretical foundations and analysis

In this section, we show that Markov parameter estimation and MS objectives arise as simplifications of the HMM formulation.

3.1. LTI systems

To begin our analysis, we consider learning state–space realizations of stochastic LTI systems. First, we analyze how existing algorithms approach the problem of unknown covariance $\boldsymbol{\Lambda}$. Second, we describe how single and multiple rollout resolve the issue of overparameterized Markov parameters that leads to the aforementioned underdetermined system. Finally, we demonstrate how learning the system matrices rather than the Markov parameters leads to faster convergence.

3.1.1. Markov parameter estimation

Many existing approaches avoid knowledge of $\boldsymbol{\Lambda}$ by minimizing an LS objective with an unweighted L_2 norm. Here, we explain this approach as assuming conditionally independent data — given parameters, inputs, and initial conditions — within the setup provided in Section 2.4.2. This assumption sets $\boldsymbol{\Lambda}_{j,k} = \mathbf{0}$ for $j \neq k$. The resulting estimator is provided below.

Proposition 1. *Assume that $\mathbf{x}_0 = \mathbf{0}$, a sample path realization $(\mathbf{u}_k, \mathbf{y}_k)$ is available, and the outputs $\mathbf{y}_k$ are conditionally independent given $\mathbf{G}_{0:n}$ and $\mathbf{U}_{0:n}$, $\forall k = 0, 1, \ldots, n$. Then, the MLE of an LTI system's Markov parameters is*

$$\hat{\mathbf{G}}_{0:n} = \operatorname*{argmin}_{\{\mathbf{G}_i\}_{i=0}^n} \sum_{k=0}^{n} \left\| \mathbf{y}_k - \sum_{i=0}^{k} \mathbf{G}_i\mathbf{u}_{k-i} \right\|^2_{\boldsymbol{\Lambda}_k}. \tag{9}$$

Proof. Conditional independence implies $p(\mathcal{Y}_n|\mathbf{G}_{0:n}, \mathbf{U}_{0:n}) = \prod_{k=0}^{n} p(\mathbf{y}_k|\mathbf{G}_{0:n}, \mathbf{U}_{0:n})$. With $\mathbf{x}_0 = \mathbf{0}$, the marginal distributions follow from the input–output relation (6) as $p(\mathbf{y}_k|\mathbf{G}_{0:n}, \mathbf{U}_{0:n}) = \mathcal{N}(\sum_{i=0}^{k} \mathbf{G}_i\mathbf{u}_{k-i}, \boldsymbol{\Lambda}_k)$. Then, taking the negative log yields the MLE of the Markov parameters (9). □

The independence assumption, however, is not sufficient to convert Eq. (8) to the LS objectives commonly used in the literature. The additional assumption that $\boldsymbol{\Lambda}_k \propto \mathbf{I}$ is needed to convert the weighted norm to a scalar multiple of the L_2 norm. This assumption holds

[2] The inputs must differ by more than a scalar multiplier to avoid underdetermination.

trivially if $d_y = 1$. Otherwise, there is no reasonable assumption to enable $\boldsymbol{\Lambda}_k \propto \mathbf{I}$. Nevertheless, if one considers an approximate objective where this is assumed to be so, one obtains

$$\hat{\mathbf{G}}_{0:n} = \underset{\{\mathbf{G}_i\}_{i=0}^{n}}{\operatorname{argmin}} \sum_{k=0}^{n} \left\| \mathbf{y}_k - \sum_{i=0}^{k} \mathbf{G}_i \mathbf{u}_{k-i} \right\|_2^2. \tag{10}$$

This approximate objective no longer requires knowledge of the system matrices and is used as the basis for a number of approaches [19–21] for both single and multiple rollout data. This work considers learning from single trajectories, so we focus only on single rollout.

For the single rollout procedure, the data are divided into K overlapping subtrajectories of length $\bar{n}$ such that $n = \bar{n} + K - 2$. To address underdetermination, one must also require $\bar{n} < \frac{n+1}{d_u}$. After dividing the single trajectory into multiple trajectories, the final output of each subtrajectory follows the same form as Eq. (6),

$$\mathbf{y}_k = \mathbf{H}\mathbf{A}^{\bar{n}-1}\mathbf{x}_{k-\bar{n}+1} + \sum_{i=0}^{\bar{n}-1} \mathbf{G}_i \mathbf{u}_{k-i} + \boldsymbol{\nu}_k, \tag{11}$$

for $k = \bar{n}, \dots, n$. Assuming that the inputs are zero-mean, the expected value of each $\mathbf{x}_{k-\bar{n}+1}$ is zero with respect to the inputs and noise variables. This is sometimes used as justification to eliminate $\mathbf{x}_{k-\bar{n}+1}$ from the estimation problem [22], and we therefore also adopt this ansatz. Adding the approximation that $\boldsymbol{\Lambda}_k \propto \mathbf{I}$ yields the following optimization problem

$$\hat{\mathbf{G}}_{0:\bar{n}-1} = \underset{\{\mathbf{G}_i\}_{i=0}^{\bar{n}-1}}{\operatorname{argmin}} \sum_{k=\bar{n}}^{n} \left\| \mathbf{y}_k - \sum_{i=0}^{\bar{n}-1} \mathbf{G}_i \mathbf{u}_{\bar{n}-i} \right\|_2^2. \tag{12}$$

The number of unknown Markov parameters is now only $\bar{n}$ rather than $n+1$, mitigating the problem of underdetermination. The LS solution is $\hat{\mathbf{G}}_{0:\bar{n}-1} = \mathbf{Y}_{\bar{n}:n}\bar{\mathbf{U}}^{\dagger}_{\bar{n}:n}$, where

$$\bar{\mathbf{U}}_{\bar{n}:n} = \begin{bmatrix} \mathbf{u}_{\bar{n}} & \mathbf{u}_{\bar{n}+1} & \cdots & \mathbf{u}_n \\ \vdots & \vdots & \cdots & \vdots \\ \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_K \end{bmatrix}.$$

Eq. (12) is equivalent to the slightly different form provided by [22].[3] Although the system of equations now has more equations than unknowns for proper choice of $\bar{n}$, the system can still suffer from underdetermination if the input signal is not *persistently exciting* [23]. Additionally, each estimated data point requires exactly $\bar{n}$ inputs. Consequently, the outer sum skips the first $\bar{n}$ data points since the inputs $\mathbf{u}_0, \mathbf{u}_{-1}, \dots$ are typically assumed unknown.

This L_2 optimization problem (12) is equivalent to the weighted L_2 optimization problem (9) under additional assumptions stated in Proposition 2.

Proposition 2. *Let the assumptions of Proposition 1 be met, and assume that* $\sum_{i=\bar{n}}^{k} \mathbf{G}_i \mathbf{u}_{k-i} = \mathbf{0}$, $\mathbf{A}^k \boldsymbol{\Sigma} (\mathbf{A}^k)^* = \mathbf{0}$ *for* $k \geq \bar{n}$ *and* $\boldsymbol{\Lambda}_{\bar{n}} \propto \mathbf{I}$. *If the first* $\bar{n}$ *outputs are discarded, then the MLE in Eq. (9) is equivalent to the estimator in Eq. (12).*

Proof. If $\sum_{i=\bar{n}}^{k} \mathbf{G}_i \mathbf{u}_{k-i} = \mathbf{0}$, then the sum in Eq. (9) is $\sum_{i=0}^{k} \mathbf{G}_i \mathbf{u}_{k-i} = \sum_{i=0}^{\bar{n}-1} \mathbf{G}_i \mathbf{u}_{k-i}$. Lastly if $\mathbf{A}^k \boldsymbol{\Sigma} (\mathbf{A}^k)^* = \mathbf{0}$ for $k \geq \bar{n}$, then $\boldsymbol{\Lambda}_k = \sum_{i=1}^{\bar{n}} \mathbf{H}\mathbf{A}^{i-1} \boldsymbol{\Sigma} (\mathbf{H}\mathbf{A}^{i-1})^* + \boldsymbol{\Gamma} = \boldsymbol{\Lambda}_{\bar{n}}$, for $k \geq \bar{n}$. By assumption, $\boldsymbol{\Lambda}_{\bar{n}} \propto \mathbf{I}$, so the weighted norm is equivalent to the standard L_2 norm. □

The assumptions $\sum_{i=\bar{n}}^{k} \mathbf{G}_i \mathbf{u}_{k-i} = \mathbf{0}$ and $\mathbf{A}^k \boldsymbol{\Sigma} (\mathbf{A}^k)^* = \mathbf{0}$ for $k \geq \bar{n}$ can be satisfied if the system has finite impulse response. Alternatively, these two assumptions can be achieved asymptotically under the much weaker assumption that $\rho(\mathbf{A}) < 1$, where $\rho(\cdot)$ denotes the spectral radius of a matrix. Such a result is given in Proposition 3.

[3] The unlabeled equation following Eq. 5 in [22].

Proposition 3. *Let* $\rho(\mathbf{A}) < 1$, *the inputs* $\mathbf{u}_k$ *be independent realizations of a real-valued random variable, and* $\boldsymbol{\Lambda}_{\bar{n}} \propto \mathbf{I}$. *As* $\bar{n} \to \infty$, *the negative log likelihood of Eq. (9) approaches the single rollout LS objective of Eq. (12) with probability 1. Moreover, it converges at least linearly.*

The proof is in Appendix A. For systems where $\boldsymbol{\Lambda}_k \propto \mathbf{I}$ approximately holds for $k > \bar{n}$, this result implies that even if $\boldsymbol{\Lambda}_k$ varies over time, a good approximation can be achieved for reasonably small $\bar{n}$. However, for systems where $\rho(\mathbf{A}) \geq 1$, the conditions of this proposition no longer hold, e.g., periodic systems have $\rho(\mathbf{A}) = 1$.

3.1.2. Numerical comparison

We now compare three approaches using the same numerical experiment from [22]. The first approach is the LS approach (12) used in single rollout in [22]. In the second approach, we assume that $\boldsymbol{\Lambda}_k$ is given, e.g., by an oracle, so we minimize the same objective as Eq. (12), but with a different weighted norm

$$\hat{\mathbf{G}}_{0:\bar{n}-1} = \underset{\{\mathbf{G}_i\}_{i=0}^{\bar{n}-1}}{\operatorname{argmin}} \sum_{k=\bar{n}}^{n} \left\| \mathbf{y}_k - \sum_{i=0}^{\bar{n}-1} \mathbf{G}_i \mathbf{u}_{k-i} \right\|_{\boldsymbol{\Lambda}_k}^2. \tag{13}$$

This objective is henceforth referred to as generalized least squares (GLS). Finally, we compare with the maximum *a-posteriori* (MAP) estimate of the Bayesian approach described in Section 2.4. The state–space approach is used numerically, though it is theoretically equivalent to the input–output approach.

This experiment was performed as follows. A random state–space system was generated by independently sampling the entries of $\mathbf{H}$ and $\mathbf{D}$ from $\mathcal{N}(0, 1/d_y)$ and of $\mathbf{B}$ from $\mathcal{N}(0, 1/d_x)$. The dimensions of the system were $d_y = 2$, $d_x = 5$, and $d_u = 3$, and the noise covariances were $\boldsymbol{\Sigma} = \sigma_\xi^2 \mathbf{I}$ and $\boldsymbol{\Gamma} = \sigma_\eta^2 \mathbf{I}$. To highlight the difference between LS and GLS, $\mathbf{A}$ was set as the identity matrix to ensure that $\rho(\mathbf{A}) = 1$ and consequently that the covariance $\boldsymbol{\Lambda}_k$ would vary significantly over time. To avoid having priors give the Bayesian algorithm an edge, improper uniform priors were placed on the state–space matrices, and only weakly informative priors of half-$\mathcal{N}(0, 1)$ were placed on the parameters σ_ξ and σ_η to enforce positivity and improve convergence. Then, data were generated by simulating the system with inputs sampled from a standard normal, i.e., $\mathbf{u}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. For this experiment, a subtrajectory length of $\bar{n} = 18$ was considered for a total of $d_y d_u \bar{n} = 108$ parameters. The Markov parameters were estimated using the first K subtrajectories of the simulated trajectory where $K = d_u \bar{n}, \dots, 2000$. Optimizing over the posterior is significantly more expensive than solving a linear LS problem, so for computational feasibility, this optimization was only performed at $K = d_u \bar{n}, d_u \bar{n} + 100, \dots, 2000$. Since the LS methods do not use the first $\bar{n}$ data points, these data were also removed from MAP estimation for consistency.

To assess the accuracy of each estimate, the spectral norm of the estimation error $\|\hat{\mathbf{G}}_{0:\bar{n}-1} - \mathbf{G}_{0:\bar{n}-1}\|_2$ was evaluated. This experiment was repeated 50 times, and the average error norm plotted against the number of data used in each estimate is shown in Figs. 1(a)–1(c) with a shaded region representing plus-minus one standard deviation. The figure also compares the LS, GLS, and MAP estimates at various noise levels $\sigma_{\xi,\eta} = 1/4, 1/2, 1$. The weighting used by GLS yielded lower error mean and variance at all noise levels compared to the LS estimate. The MAP estimate produced the lowest mean error and error variance of all by a significant margin. The same experiment was repeated with larger dimensions of $d_y = 8$, $d_x = 10$, and $d_u = 5$ for a total 720 parameters, and the results are shown in Figs. 1(d)–1(f). Again, the ranking of the performance of the estimates is the same, but in the larger system, the improvement achieved by the MAP estimate is even greater. We also observe very little degradation of the MAP estimate when the system dimensions increase other than in convergence rate, which can be attributed to the greater number of parameters. The results of these experiments illustrate the performance costs incurred by adding simplifying assumptions into the objective.

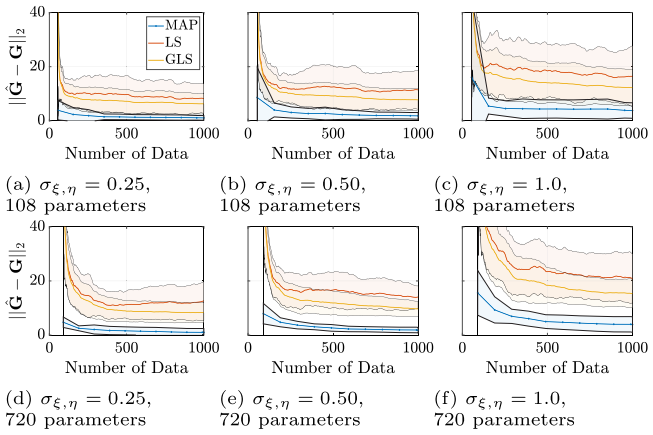


Fig. 1. A comparison of the spectral norm of the Markov parameters estimation error for $\bar{n} = 18$ using the LS (12), GLS (13), and MAP estimates at varying noise levels. The lines represent mean error and the shaded regions are plus-minus one standard deviation.

3.2. Nonlinear systems

We next turn to nonlinear system ID, where the dynamics are parameterized via nonlinear mappings such as neural networks or nonlinear partial differential equations (PDEs). In nonlinear system ID, there are certain behaviors that the models and/or underlying systems can exhibit that make many LS objectives unsuitable for estimation. For example, many nonlinear models can have their states approach infinity in finite time depending on the model parameters. Therefore, if the objective requires the model to be simulated over a relatively long period of time, the optimizer will likely run into this diverging behavior often. When the states diverge, the objective function value and gradients are undefined, which makes optimization especially difficult. Another example is chaos. In a chaotic system, arbitrarily small perturbations to the state grow exponentially over time. Thus, even small errors in the model's simulated state can quickly lead to large errors, making it difficult to discern good models. Mechanically, these issues lead to objectives that are multi-modal and difficult to optimize. To address these issues, the MS objective [12] introduces simulation length as a design parameter as a means to avoid the excessive error growth common in nonlinear systems.

3.2.1. Discussion on the common least squares objective

Before introducing the MS objective, we first provide a discussion on two fundamental objectives that will motivate the design of the MS objective. An objective function $\mathcal{J}$ in system ID is typically defined as follows:

$$\mathcal{J}(\boldsymbol{\theta}) = \sum_{k=0}^{n} \|\mathbf{y}_k - \hat{\mathbf{y}}_k(\boldsymbol{\theta})\|_2^2, \tag{14}$$

where $\mathbf{y}$ and $\hat{\mathbf{y}}$ are the observed and estimated outputs, respectively, and a regularization/physics-enforcing term is sometimes added [24,25]. Regularization aside, the differences among objective functions typically come from how the estimated output $\hat{\mathbf{y}}$ is evaluated. For expositional purposes, let us define a function $f(\mathbf{x}_i, \mathbf{u}_{i:j}, t_j; \boldsymbol{\theta})$ parameterized by $\boldsymbol{\theta}$ that maps an initial state $\mathbf{x}_i$ to the output at time t_j with inputs from time t_i to time t_j denoted as $\mathbf{u}_{i:j}$, where $i \leq j$. One fundamental objective uses a single initial condition $\mathbf{x}_0$ and evaluates $\hat{\mathbf{y}}$ as

$$\hat{\mathbf{y}}_k = f(\mathbf{x}_0, \mathbf{u}_{0:k}, t_k; \boldsymbol{\theta}), \tag{15}$$

for all $k \geq 0$. Since the system is simulated without any corrections to the trajectory, we refer to this objective as the *deterministic LS*. This objective is used to train a number of machine learning models [26–28]. The other fundamental objective estimates the output as

$$\hat{\mathbf{y}}_k = f(\mathbf{y}_{k-1}, \mathbf{u}_{k-1:k}, t_k; \boldsymbol{\theta}), \tag{16}$$

using the most recent data point as the initial condition and simulating the system only until the time at which the next data point is available. We refer to this objective as the *propagator LS* since in this case, f propagates one data point to the next. This objective is used within popular system ID algorithms [13,29] and for training certain neural networks [30]. We reinforce the difference between Eqs. (15) and (16) is that Eq. (15) uses $\mathbf{x}_0$ as the initial condition and returns the output after k timesteps whereas Eq. (16) uses $\mathbf{y}_{k-1}$ as the initial condition and returns the output after only one timestep.

There have been several comparisons between the deterministic and propagator LS objectives [1,31–33] that have shown that the deterministic LS yields better estimates when measurement noise is included and process noise is omitted, and the propagator LS yields better estimates when process noise is included and measurement noise is omitted. In addition, the deterministic LS is much more difficult and computationally intensive to optimize since it involves compositions of the dynamics propagator, which allows errors to accumulate and leads to complicated, non-convex surfaces.

3.2.2. Multiple shooting objective

Here, the MS objective is introduced. We then show that the deterministic (15) and propagator (16) LS objectives can be seen as special cases of MS. In MS, the output trajectory is divided into L disjoint subtrajectories with initial times $\{t_{\ell_i}\}_{i=1}^{L}$ such that the ith subtrajectory has length $\Delta\ell_i := \ell_{i+1} - \ell_i$. Then the output at time t_k contained within the ith subtrajectory is estimated as $\hat{\mathbf{y}}_k = f(\mathbf{x}_{\ell_i}, \mathbf{u}_{\ell_i:k}, t_k; \boldsymbol{\theta})$. Such an objective function requires the estimation of the set of subtrajectory initial conditions $\mathcal{Z}_L := \{\mathbf{x}_{\ell_i}\}_{i=1}^{L}$, which can be done by adding the initial conditions as parameters [33], training an encoder [34,35], or, if the system is fully observed, simply using the data $\mathbf{x}_{\ell_i} = \mathbf{y}_{\ell_i}$ [36]. In effect, this method introduces additional parameters (the initial conditions) as the cost for an improved estimate.

The MS objective function is defined as

$$\mathcal{J}(\boldsymbol{\theta}) = \sum_{i=1}^{L} \sum_{k=\ell_i}^{\ell_{i+1}-1} \|\mathbf{y}_k - \hat{\mathbf{y}}_k\|_2^2, \tag{17}$$

where $\ell_{L+1} := n + 1$. Oftentimes, a constant length of $T = \Delta\ell_i$ for all $i = 1, \ldots, L$ is used for simplicity. In the case $T = n$, the objective is equivalent to the deterministic LS, and if $T = 1$, the objective is equivalent to the propagator LS. Therefore, when $1 < T < n$, MS can be seen as a type of combination of these two objectives. In the original paper [8], the objective additionally had the constraint that $\mathbf{x}_{\ell_{i+1}} = \Psi^{\Delta\ell_i}(\mathbf{x}_i, \boldsymbol{\theta}_\Psi)$, where $\Psi^{\Delta\ell_i}$ denotes $\Delta\ell_i$ compositions of the Ψ operator, but this constraint is sometimes removed to simplify optimization. We will distinguish between the objectives with and without the constraints by referring to them as the constrained and unconstrained MS objectives, respectively.

3.2.3. Relation to probabilistic approach

From a probabilistic perspective, the MS objective can be viewed as a joint parameter-state estimation problem. Rather than estimating the state at every timestep, however, only the subset of subtrajectory initial conditions $\mathcal{Z}_L \subseteq \mathcal{X}_n$ are estimated. The posterior of such a problem can be factorized with Bayes' rule as

$$p(\mathcal{Z}_L, \boldsymbol{\theta} | \mathcal{Y}_n) \propto \mathcal{L}(\boldsymbol{\theta}, \mathcal{Z}_L; \mathcal{Y}_n) p(\mathcal{Z}_L, \boldsymbol{\theta}). \tag{18}$$

Factorizing further, the likelihood can be decomposed as $\mathcal{L}(\boldsymbol{\theta}, \mathcal{Z}_L; \mathcal{Y}_n) = \prod_{k=\ell_i}^{\ell_{i+1}-1} p(\mathbf{y}_k | \mathbf{x}_{\ell_i}, \boldsymbol{\theta})$ and the prior also as $p(\mathcal{Z}_L, \boldsymbol{\theta}) = p(\boldsymbol{\theta}) \prod_{i=1}^{L} p(\mathbf{x}_{\ell_i} | \mathbf{x}_{\ell_{i-1}}, \boldsymbol{\theta})$, with $p(\mathbf{x}_{\ell_1} | \mathbf{x}_{\ell_0}, \boldsymbol{\theta}) := p(\mathbf{x}_{\ell_1} | \boldsymbol{\theta})$. Each term in the likelihood $p(\mathbf{y}_k | \mathbf{x}_{\ell_i}, \boldsymbol{\theta})$ can still be evaluated with Algorithm 1 using data from only a single trajectory.

The most significant difference is that the prior has the added terms $p(\mathbf{x}_{\ell_i} | \mathbf{x}_{\ell_{i-1}}, \boldsymbol{\theta})$, which can be evaluated as

$$p(\mathbf{x}_{\ell_i} | \mathbf{x}_{\ell_{i-1}}, \boldsymbol{\theta}) = \int \prod_{k=\ell_{i-1}+1}^{\ell_i} p(\mathbf{x}_k | \mathbf{x}_{k-1}) \mathrm{d}\mathbf{x}_{\ell_i - 1} \mathrm{d}\mathbf{x}_{\ell_i - 2} \ldots \mathrm{d}\mathbf{x}_{\ell_{i-1}+1}. \tag{19}$$

This equation shows that $p(\mathbf{x}_{\ell_i}|\mathbf{x}_{\ell_{i-1}},\boldsymbol{\theta})$ can be seen as representing the probability of $\mathbf{x}_{\ell_i}$ averaged over all trajectories that start at $\mathbf{x}_{\ell_{i-1}}$ with dynamics determined by $\boldsymbol{\theta}$. Alternatively, this term can be viewed as a soft constraint enforcing the estimated initial conditions to be connected by a single trajectory with initial condition $\mathbf{x}_{\ell_1}$. Under certain conditions, this constraint is equivalent to the MS constraints. Furthermore, estimators based on the posterior (18) are equivalent to estimators using the (un)constrained MS objective (17) under certain assumptions. This result is stated in Proposition 4.

Proposition 4. *Assume an improper uniform prior distribution on the parameters $\boldsymbol{\theta}$ and that $\boldsymbol{\Sigma} = \mathbf{0}$. Then, the negative log marginal likelihood of the joint parameter-state estimation problem (18) is equivalent to the unconstrained MS objective (17). Moreover, the negative log posterior is equivalent to the constrained MS objective.*

Proof. According to Theorem 2 in [1], each term in the marginal likelihood $\prod_{k=\ell_i}^{\ell_{i+1}-1} p(\mathbf{y}_k|\mathbf{x}_{\ell_i},\boldsymbol{\theta})$ is equivalent to a deterministic LS objective when $\boldsymbol{\Sigma} = \mathbf{0}$. Then, taking the negative log of this product gives the unconstrained MS objective. When the prior over the states is added, each term $p(\mathbf{x}_{\ell_i}|\mathbf{x}_{\ell_{i-1}},\boldsymbol{\theta})$ approaches the Dirac delta function $\delta_{\boldsymbol{\psi}^{\Delta\ell_i}(\mathbf{x}_{\ell_{i-1}},\boldsymbol{\theta}_\psi)}(\mathbf{x}_{\ell_i})$ as $\boldsymbol{\Sigma} \to \mathbf{0}$. These delta functions are equivalent to the constraints in the constrained MS objective. The prior over the parameters is constant, so taking the negative log of the posterior yields the constrained MS objective. □

3.2.4. Comparison of smoothing effects

Now that it is understood that the MS objective is equivalent to an objective of a joint parameter-state estimation problem, we demonstrate empirically that the additional expense of inferring the subtrajectory initial conditions is not computationally necessary. More specifically, we demonstrate that the proposed marginal likelihood improves the optimization surface in a similar fashion to the MS objective, and we provide a brief discussion on the advantages of the proposed approach.

To show the similarity of the smoothing effects in MS and the marginal likelihood, the logistic map example from [33] is considered. The logistic map is defined as

$$y_{k+1} = \theta y_k(1 - y_k). \tag{20}$$

This system exhibits chaotic behavior when θ is within the range [3.57, 4]. For this example, θ was set to 3.78, an initial condition of $y_0 = 0.5$ was used, and 200 noiseless data points were collected. To show how the objective surfaces vary with θ, all other parameters must be fixed. For the MS objective, the initial conditions were set to the true values, and for the marginal likelihood, we set $\boldsymbol{\Gamma} = 10^{-16}$ to maintain positive definiteness. Then, the objectives were compared at different time horizons T and variance ratios $\boldsymbol{\Sigma}/\boldsymbol{\Gamma}$. The ratio $\boldsymbol{\Sigma}/\boldsymbol{\Gamma}$ is used because in this problem, the shape of the objective did not appear to change for a fixed ratio, regardless of the $\boldsymbol{\Sigma}$ and $\boldsymbol{\Gamma}$ values. Note that the validity of using this ratio is only possible since the full state is observed, meaning that $\boldsymbol{\Sigma}$ and $\boldsymbol{\Gamma}$ are represented in the same coordinate frame.

Both surfaces were normalized to equal 1.0 at $\theta = 2$, and the results are shown in Fig. 2. There is no exact mapping between T and $\boldsymbol{\Sigma}/\boldsymbol{\Gamma}$, so the values of $\boldsymbol{\Sigma}/\boldsymbol{\Gamma}$ were chosen such that the smoothness of the marginal likelihood roughly matched that of the MS objective by visual comparison. Fig. 2(a) shows values of T and $\boldsymbol{\Sigma}/\boldsymbol{\Gamma}$ where MS is equivalent to and the marginal likelihood approximates the deterministic LS. Due to the chaotic nature of this system, the deterministic LS objective is filled with local minima that make optimization extremely difficult. As T is decreased and $\boldsymbol{\Sigma}$ is increased, both surfaces show increasing smoothness, demonstrating the similar effect these variables have on their respective objectives. An important difference to note between T and $\boldsymbol{\Sigma}$ is that T is a discrete scalar variable, whereas $\boldsymbol{\Sigma}$ is a positive definite matrix of continuous values. Therefore, $\boldsymbol{\Sigma}$ gives the user greater flexibility when tuning the marginal likelihood, including the ability to use different variance values for different components of the state.

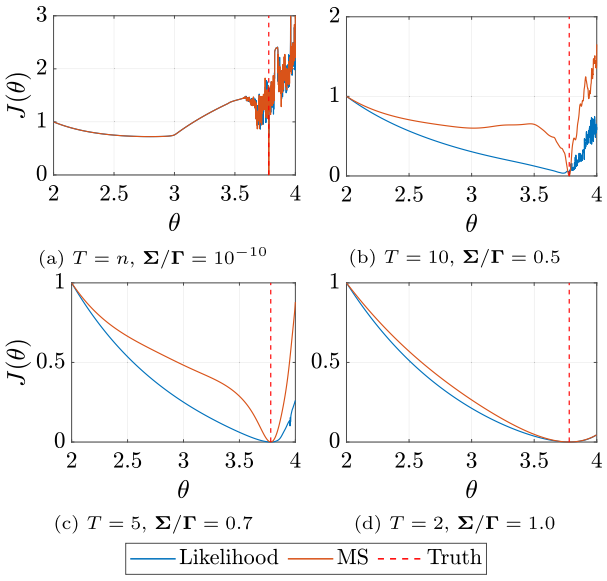


Fig. 2. Comparison of the log marginal likelihood and MS objective (17) as $\boldsymbol{\Sigma}$ and T vary. When $T = n$, as in Fig. 2(a), MS is equivalent to deterministic LS (15).

4. Numerical experiments

In this section, four numerical experiments are conducted to compare the performance of the posterior to the LS+ERA (12), deterministic LS (15), and MS (17) objective functions. The first experiment considers a linear pendulum with forcing for various data timesteps and noise levels. This experiment shows that for every timestep and noise level considered, the Bayesian approach can, on average, improve the estimate provided by the LS+ERA algorithm. The second example uses the Wiener–Hammerstein system ID benchmark [37] and shows that the Bayesian approach displays greater robustness than MS when the number of training data is reduced and noise is present in the data. The third example is the forced Duffing oscillator and shows that the deterministic LS and MS metrics cannot always identify good models for chaotic systems, but the Bayesian approach is still applicable. The final example considers the prediction of a PDE quantity of interest and shows that deterministic LS struggles when noise is introduced and there are more model parameters than data, whereas the posterior can account for both the measurement noise and the expressiveness of the model.

4.1. Algorithmic considerations

In this section, we discuss several implementation considerations necessary for reproducibility. These include the tuning of the Bayesian method, strategies for sampling non-identifiable parameters, a methodology for evaluating model estimate quality, and a statement of the software used for experimentation.

4.1.1. Tuning

For each numerical experiment, we follow the same general procedure when applying the Bayesian approach. First, we set the initial state covariance matrix to be a small positive definite matrix $\mathbf{P}_0 = \mathbf{I} \times 10^{-8}$ and parameterize the unknown noise covariances as diagonal matrices. Then, we numerically minimize the negative log posterior using an interior point algorithm [38] with analytical gradients supplied to produce an estimate of the MAP point. The initial point of this optimization is problem-dependent and will be stated in each experiment. To sample, we use a delayed rejection adaptive Metropolis (DRAM) [39] within Gibbs procedure to sequentially sample the parameter groups $\{\boldsymbol{\theta}_{\mathbf{x}_0}\}$,

$\{\boldsymbol{\theta}_{\boldsymbol{\Psi}}\}$, and $\{\boldsymbol{\theta}_{\boldsymbol{\Sigma}}, \boldsymbol{\theta}_{\boldsymbol{\Gamma}}\}$. The sampler is initialized at the estimated MAP with an initial proposal covariance proportional to the identity matrix. The scaling of this matrix, the number of sampling iterations, and the burn-in period are all problem-dependent and stated in the examples. Adaptation of the proposal covariance starts after 200 iterations, and the second-tier proposal covariance is equal to the first-tier proposal covariance scaled by 0.01. Lastly, we use these samples to generate probabilistic forecasts. To reduce the correlation between samples, we subsample the chain of samples at regular intervals before generating forecasts.

4.1.2. Parameter non-identifiability and MCMC sampling

One of the goals of the Bayesian approach is to quantify the uncertainty in the output behavior of the system of interest. Toward this goal, we seek to draw MCMC samples from the output space of the model dynamics. For parameterized functions, this consists of drawing a set of parameter samples with corresponding output samples that have converged to the posterior. When there exists a one-to-one mapping between parameter and output values, the convergence in output space is equivalent to the convergence in parameter space. If the function is overparameterized, however, the parameter values of a function are non-unique, and convergence in parameter space is no longer a necessity [40].

To achieve convergence in the output space, it is important for the sampler to be able to freely traverse the parameter space without getting stuck in local minima. This can be a challenge when sampling non-identifiable parameters, which generate complicated correlations in the posterior. Non-identifiability arises in the system ID framework from two main sources. The first source pertains to the coordinate frame of the state space — a state–space dynamics model is unique only up to a change of coordinates. In an effort to mitigate this source of non-identifiability, we seek to constrain the model coordinate frame by fixing the observation parameters $\boldsymbol{\theta}_h$ (also $\mathbf{B}$ in LTI models) at the MAP and only sampling the remaining parameters. Fixing parameters runs the risk of neglecting uncertainty, but we are primarily concerned with the uncertainty in the output behavior, not in the parameters. This constraint should theoretically not restrict the behavior of the model dynamics nor the uncertainty in the output, so we consider this an acceptable tradeoff.

For better mixing in the parameter space, we empirically found that coupling the approach of fixing $\boldsymbol{\theta}_h$ with a DRAM within Gibbs procedure worked well. As an illustration of this finding, Fig. 3 shows samples drawn from the $\mathbf{A}$ matrix of the linear system in Section 4.2 using both the proposed approach and the approach of running DRAM over all parameters at once. Figs. 3(a) and 3(b) show the samples drawn when using standard DRAM. With this approach, 10^7 samples were drawn, the first 10^6 discarded as burn-in, and every 1,000th remaining sample was plotted. Figs. 3(c) and 3(d) show samples drawn with the DRAM within Gibbs approach with matrices $\mathbf{B}$ and $\mathbf{H}$ fixed at the estimated MAP. For this method, 10^6 samples were drawn, 10^5 were discarded as burn-in, and every 1,000th remaining sample was plotted. Despite the former approach drawing 10 times as many samples, we observe that the latter approach covers more of the posterior. Moreover, the mixing of the chain in the DRAM within Gibbs approach appears much better.

Note that the interesting relationship between the parameters in Fig. 3(c) reveals that θ_3 and θ_4 remain non-identifiable. This observation is the result of the second source of non-identifiability: overparameterization. Having more parameters than degrees of freedom in the system implies parameter non-uniqueness. Without additional knowledge on the system or more intensive modeling efforts, this type of non-identifiability cannot easily be removed. However, modern research suggests that it may not be an issue if we are only interested in capturing the output uncertainty. The authors in [40] achieved convergence in predicted outputs despite poor mixing in parameter space when using MCMC to sample the weights of a deep neural network.

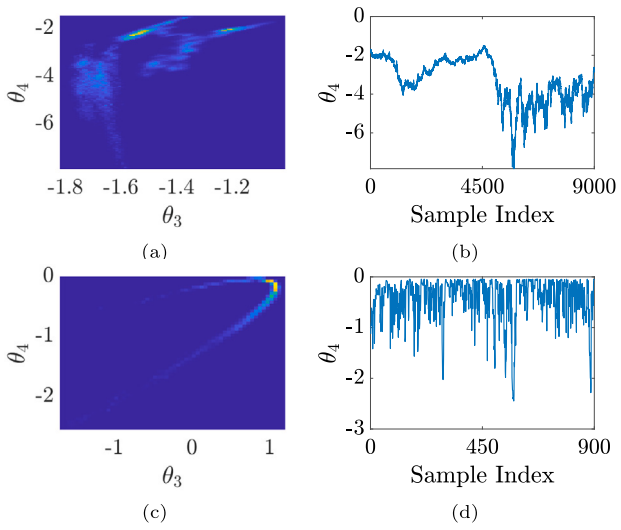


Fig. 3. 2D marginal distributions and chains from MCMC sampling System (22). Figs. 3(a) and 3(b) show samples drawn using DRAM without alteration, and Figs. 3(c) and 3(d) show samples drawn using a DRAM within Gibbs procedure with the parameters in matrices $\mathbf{B}$ and $\mathbf{H}$ fixed at the estimated MAP point.

Additionally, [41,42] independently showed that local minima in the objective function of a neural network are connected by simple curves of nearly constant loss value, allowing for a sampler to easily traverse across multiple modes. Aligned with these results, we observe good mixing for predictions in the numerical experiments in this work, as shown in Appendix B.

4.1.3. Testing methodology

In this section, we discuss approaches for generating testing data and output estimates. Generating testing data using a different initial condition than the training data is non-trivial since each method learns its own coordinate frame and initial condition. Therefore, in each experiment we simulate the system from a single initial condition and use disjoint subsets of the resulting output trajectory for training and testing. The training subset always begins at the initial condition, but the testing subset can begin any time after the training subset ends. In most cases, the testing subset is equal in length to the training subset and begins at the first timestep after the training data period ends. To generate output estimates from each method, we simulate each model estimate starting at the beginning of the training period and terminating at the end of the testing period.

In the Bayesian approach, we are interested in the distribution of output estimates generated by the parameter posterior. To draw samples of this output distribution, we can simply simulate the model using parameter posterior samples. In this approach, the full parameter sample must be used for simulation, including $\boldsymbol{\theta}_{\mathbf{x}_0}$. If a point estimate is desired, we can either simulate the model with a point chosen from the parameter posterior or select a point from the output posterior. This choice is problem-dependent and will be noted in each experiment.

Another choice when generating probabilistic output estimates is how to simulate the model. We have identified two conceivable approaches for output estimation that we use throughout the experiments.

Stochastic simulation. The first method will be referred to as 'stochastic simulation.' There are two phases of this approach. During the beginning period of simulation in which data are available, a Bayesian filter is run for estimation. Once the data period ends, forecasting evolves according to Eq. (1a) with random realizations of the process noise added at each timestep. This approach is meant to help visualize how the model uncertainty and data affect predictions.

Deterministic simulation. The other method for prediction will be referred to as 'deterministic simulation.' In this method, the model dynamics are simply compositions of the propagator Ψ with neither filtering nor process noise realizations. This approach can be used by any system ID method, so the predictions from this approach will be the ones used for evaluating performance metrics. Note that using this approach with MS implies that the estimated initial conditions in the set $\mathcal{Z}_L$ are not used in simulation except for the initial condition at time t_0.

4.1.4. Software

The following experiments were run using MATLAB R2020a, and the code for these experiments can be found at https://github.com/ngalioto/BayesID. In all experiments, optimization was performed with the `fmincon` function with default hyperparameters and integration with the `ode45` function.

4.2. Linear pendulum with control

The first example considers a linear pendulum with unit length and mass with damping and random inputs. This example builds upon that presented in [43] by comparing to the single rollout LS objective (12) rather than the exactly determined objective (9). In addition, sampling is added to make probabilistic forecasts that quantify uncertainty using deterministic simulation.

The system dynamics are given as

$$\begin{aligned} \mathbf{x}_{k+1} &= \operatorname{expm}\left(\begin{bmatrix} 0 & 1 \\ -9.81 & -1 \end{bmatrix} \Delta t\right) \mathbf{x}_k + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_k, \\ \mathrm{y}_k &= \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{x}_k + \eta_k; \qquad \mathbf{x}_0 = \mathbf{0}, \end{aligned} \tag{21}$$

where expm is the matrix exponential and the inputs are Gaussian-distributed as $u_k \sim \mathcal{N}(0, \Delta t)$. The damping term is included to ensure that the $\mathbf{A}$ matrix is asymptotically stable at all Δt considered. This damping term ensures $\rho(\mathbf{A}) < 1$, so the theoretical requirements in [22] for the LS+ERA are met.

4.2.1. Data generation and training

Data were collected from the system (21) over a 20 s training period at various timesteps and noise levels. For this experiment, timesteps of $\Delta t = 0.10, 0.15, \ldots, 0.50$ and noise ratios of $\sigma = 0.000, 0.025, \ldots, 0.200$ were considered. Here, the noise ratio is defined as $\sigma := \sigma_{\boldsymbol{\eta}} / \max(\mathbf{x}[1])$, where $\sigma_{\boldsymbol{\eta}}$ is the standard deviation of the measurement noise. For each noise-timestep pair, 100 realizations of data were generated, and each method trained on every realization separately, yielding 100 estimated models per pair. Note that since the inputs are random, the system behavior is also random and the noise ratio of each dataset therefore varies, even within a given noise-timestep pair.

The model parameterization is

$$\begin{aligned} \mathbf{x}_0 = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}; \mathbf{x}_{k+1} &= \begin{bmatrix} \theta_3 & \theta_5 \\ \theta_4 & \theta_6 \end{bmatrix} \mathbf{x}_k + \begin{bmatrix} \theta_7 \\ \theta_8 \end{bmatrix} u_k + \boldsymbol{\xi}_k, \\ \mathrm{y}_k &= \begin{bmatrix} \theta_9 & \theta_{10} \end{bmatrix} \mathbf{x}_k + \eta_k, \\ \boldsymbol{\xi}_k \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \theta_{11} & 0 \\ 0 & \theta_{12} \end{bmatrix}\right), &\quad \eta_k \sim \mathcal{N}(0, \theta_{13}). \end{aligned} \tag{22}$$

The priors half-$\mathcal{N}(0, 10^{-6})$ and half-$\mathcal{N}(0, 1)$ were placed on $\boldsymbol{\theta}_{\boldsymbol{\Sigma}}$ and $\boldsymbol{\theta}_{\boldsymbol{\Gamma}}$, respectively, and an improper uniform prior was placed on the remaining 10 parameters. The initial points for optimization were $\boldsymbol{\theta}_{\mathbf{x}_0} = \mathbf{0}$, $\boldsymbol{\theta}_{\Psi} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\boldsymbol{\theta}_{\boldsymbol{\Sigma}} = \mathbf{1} \times 10^{-6}$, and $\boldsymbol{\theta}_{\boldsymbol{\Gamma}} = \sigma_{\boldsymbol{\eta}}^2 + 10^{-8}$, where 10^{-8} is added to ensure positive definiteness when $\sigma_{\boldsymbol{\eta}} = 0$. On every dataset, optimization was run upwards of six times, each with a different initial $\boldsymbol{\theta}_{\Psi}$ sample, and the result with the lowest negative log posterior was used as the MAP.

To get a sense for the computational cost of each method, the training times on a single dataset are shown in Table 1. The selected dataset has a noise ratio of $\sigma = 0.2$ and a timestep of $\Delta t = 0.1$. Since optimization in the LS method involves computing only a linear LS solve, there are no iterations or evaluations of the objective function.

Table 1
Optimization times on a single dataset in the linear pendulum with control example. The dataset comes from the (noise, timestep) pair of (0.20,0.1). 'Bayes' represents optimization over the negative log posterior, and 'LS' represents solving the linear LS problem (12) for Markov parameter estimation. The average is the total time divided by the number of function evaluations.

	Iterations	Function evaluations	Time (ms)	Average (ms/func.)
Bayes	257	4,020	3,350	0.833
LS	N/A	N/A	0.183	N/A

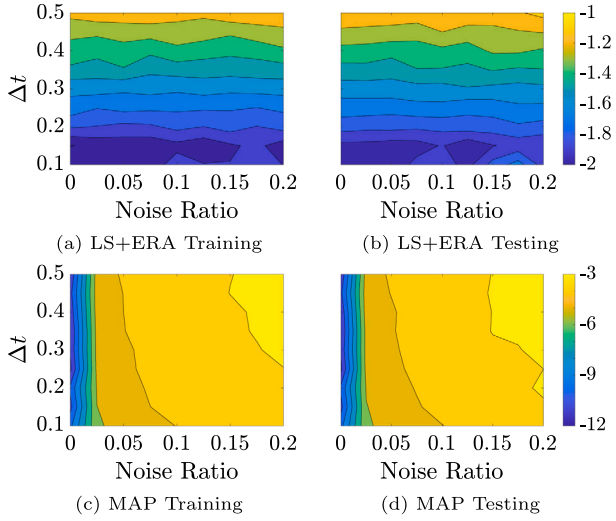


Fig. 4. Contour plots of the $\log_{10}$(MSE) of the LS+ERA and MAP estimates on the linear pendulum system (21). Figs. 4(a) and 4(c) are the training MSE and Figs. 4(b) and 4(d) are the testing MSE.

4.2.2. MSE contours over noise ratio and timestep

The LS+ERA and MAP estimates were compared with respect to the $\log_{10}$ of the average mean squared error (MSE) at each noise-timestep pair. For a given pair of values of σ and Δt, let $i = 1, \ldots, 100$ index the data realizations. Then, the MSE on the ith dataset is defined as $\frac{1}{n}\sum_{k=1}^{n}(x_k[1] - \hat{y}_k(\boldsymbol{\theta}_i))^2$, where $\boldsymbol{\theta}_i$ represents the parameter estimate on the given dataset. Additionally, the MSE on a testing period of 20 s beyond the training data was also calculated over time indices $k = n+1, \ldots, 2n$. There were a handful of outliers in the LS+ERA MSE that significantly skewed the average value, so only the lowest 99 MSE values were used to compute the average MSE of the LS+ERA estimate. The average MSE of the MAP estimate retains all 100 MSE values.

Contour plots of the $\log_{10}$ average MSE of the LS+ERA and MAP estimates are given in Fig. 4. We observe that the LS+ERA performance degrades most significantly as the timestep increases as a consequence of having fewer data available for estimation. The MAP estimate, on the other hand, appears to degrade more as the noise ratio is decreased, but its degradation due to increasing timestep is of similar magnitude. The slower degradation along the timestep axis suggests that the Bayesian approach has low data requirements, especially when the data are low-noise. Based on the colorbars of each set of plots, the MAP estimate gives at least an order of magnitude of improvement over the LS+ERA estimate.

4.2.3. Selected noise-timestep pairs

To get a better understanding of how each method performs when the data are noisy/sparse, two datasets from different (noise, timestep) pairs were chosen on which to examine the estimated output from each method. The selected pairs are the high noise, low sparsity case (0.20, 0.1), and the low noise, high sparsity case (0.00, 0.5). These two pairs were chosen so that the effect of high noise and high sparsity could

be studied separately. Within each pair, the data realization chosen was the dataset for which the LS+ERA method had the lowest training MSE so that its performance was fairly represented. The estimated outputs of the two estimates are shown in Figs. 5(a) and 5(b) for the high noise and high sparsity cases, respectively. Furthermore, we examine the impulse response to check whether the model only learned how to produce sinuoids at a given frequency or if the estimated system actually approximates a realization of the state–space matrices as desired. The impulse response for the models in the high noise and high sparsity cases are shown in Figs. 5(c) and 5(d), respectively.

To represent the posterior predictive distribution, 10^6 samples were drawn using the MCMC procedure described earlier in Section 4, the first 10^5 were discarded as burn-in, and 100 samples were selected at regular intervals to be simulated and plotted. The initial proposal covariance was $\mathbf{I} \times 10^{-5}$ for every parameter group. The blue 'mean' line indicates the mean of these posterior predictive samples.

In the high noise, low sparsity case (Figs. 5(a) and 5(c)), the mean estimate and LS+ERA estimate both appear to fit the truth fairly well, despite the noisiness of the data. The MSE of the LS+ERA estimate over the full 40 s is 2.86×10^{-4} and the MSE of the mean estimate is 4.74×10^{-4}. This was the only dataset on which the LS+ERA MSE was less than that of the MAP estimate. In fact, the next lowest LS+ERA training MSE was 4.68×10^{-4}, which is larger than the average MAP training MSE of 4.47×10^{-4}. Furthermore, the standard deviations of the LS+ERA training and testing MSE over all 100 datasets are 1.72×10^{-2} and 2.58×10^{-2} respectively, while the training and testing standard deviations of the MAP are 3.65×10^{-4} and 5.56×10^{-4}, respectively. The fact that the standard deviation of the LS+ERA MSE is about 100 times greater than that of the MAP MSE indicates that there are far worse LS+ERA estimates than the one presented here, but the MAP estimates likely all resemble the one shown in the figure.

In the impulse response, the LS+ERA estimate is also closer to the truth than the mean estimate, but the posterior is wide and encompasses the truth. Therefore, the Bayesian method is 'aware' of the error and gives a reasonable quantification of the estimate uncertainty. In the low noise, high sparsity case (Figs. 5(b) and 5(d)), the posterior is so narrow that it visually appears as a single line, indicating low uncertainty. In both the output and impulse response plots, the mean is directly on top of the truth, and the LS+ERA estimate has large discrepancies between its output and the truth. In contrast to the Bayesian estimate, the LS+ERA method has no way to identify this larger error/uncertainty.

4.2.4. Remarks on timestep

In these experiments, the length of the testing period was kept constant at 20 s, and the timestep was varied. This had the effect of changing the number of training data for each timestep. An alternative approach for investigating the effects of Δt would be to keep the number of training data constant such that the length of the training period changes for each timestep. Assuming that the timestep is still small enough to satisfy the Nyquist criterion, larger timesteps would be more advantageous for estimation in this case. The reason for this pertains to the correlations between data from a single trajectory. Data that are nearby in time have greater correlation, making the information held by each data point more "redundant". As a result, data with larger timesteps carry more information on the system compared to the same number of data sampled at smaller timesteps.

Note, however, that for the LS Markov parameter estimation approach (12), the number of data is much more important than the data timestep. Proposition 2 showed that in the LS method, each data point is treated as being generated from an independent trajectory of length $\bar{n}$, implying that the correlation between data points and the length of the training period are not taken into account.

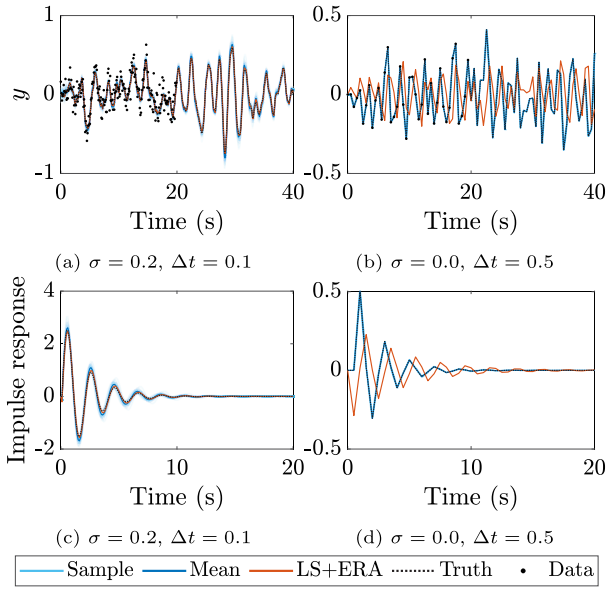


Fig. 5. The Bayesian estimate is compared to the LS+ERA with $\bar{n} = 18$. Figs. 5(a) and 5(b) show the LS+ERA estimate, deterministic simulations of 100 posterior samples, and the mean of the sample outputs. Figs. 5(c) and 5(d) show the impulse response of each of these estimates. Figs. 5(a) and 5(c) show the high noise, low sparsity case. Figs. 5(b) and 5(d) are the low noise, high sparsity case.

4.2.5. Conclusions

This example demonstrated the robustness of Bayesian system ID at varying timesteps and noise levels compared to an LS-based LTI system ID method. Since the LS method treats the data as being conditionally independent, the quality of its estimation strongly depends on the number of training data available. The Bayesian method, on the other hand, makes no such assumption, and therefore its estimate quality does not vary as much as the timestep changes. Instead, the estimate quality degrades gradually as the noise is increased. Moreover, the probabilistic approach of the Bayesian method allows for assessing the estimate accuracy through uncertainty quantification without the need for additional data.

4.3. Wiener–Hammerstein benchmark

Next, experimental data collected from a nonlinear system are considered. For this example, we use data from the Wiener–Hammerstein benchmark [37], which is a standard dataset that has been used to compare the performances of different nonlinear system ID methods. The benchmark dataset is composed of 188,000 low-noise input–output data points, with a suggested training/testing split of 100,000/88,000. The best performance to date on this benchmark to the authors' knowledge comes from [35], which uses the MS objective (17). In this experiment, the method of [35] will be compared to the Bayesian method.

4.3.1. Data generation and training

Because the dataset has such a high number of data points with low measurement noise, the advantages of the Bayesian approach are not nearly as evident. Therefore, we seek to assess the performance of the state-of-the-art method when the dataset size is reduced and noise is added. To this end, only the first 1000 data points of the original 100,000 point training set were used for training. Furthermore, zero-mean Gaussian noise with standard deviation $\sigma = 0.0178$ was added to these training data. This standard deviation is equal to 1% of $(\mathbf{y}_{max} - \mathbf{y}_{min})$. The same testing set as [35] was used for comparison, which begins 99,000 timesteps after the shortened training period.

Table 2
Optimization times on the Wiener–Hammerstein example using the negative log posterior (Bayes) and the state-of-the-art method (MS).

	Iterations	Function evaluations	Time (h:min:s)	Average (s/func.)
Bayes	10,000	15,961	20:09:13	4.5
MS	100,000	100,000	23:01:57	0.83

The nonlinear model follows the form of Eq. (1) with latent space dimension $d_x = 6$, where the dynamics operator Ψ and observation operator h are now parameterized as neural networks. Following the approach of [35], each neural network has a single hidden layer with 15 nodes and tanh activation functions. Additionally, a linear transformation from the input of the network directly to the output is included such that the network form is

$$\mathbf{z}_{out} = \mathbf{A}_1(\boldsymbol{\theta})\tanh(\mathbf{A}_2(\boldsymbol{\theta})\mathbf{z}_{in} + \mathbf{b}_2(\boldsymbol{\theta})) + \mathbf{A}_3(\boldsymbol{\theta})\mathbf{z}_{in} + \mathbf{b}_3(\boldsymbol{\theta}), \tag{23}$$

where $\mathbf{z}_{in} = \begin{bmatrix}\mathbf{x}_k^* & u_k\end{bmatrix}^*$. For the dynamics network Ψ, $\mathbf{z}_{out} = \mathbf{x}_{k+1}$, and for the observation network h, $\mathbf{z}_{out} = \mathbf{y}_k$. The combined number of parameters in Ψ and h is 401. The one difference between this model and that of [35] is that rather than learning an encoder function to estimate the current state, the initial condition is estimated directly. The priors used for this model were log–$\mathcal{N}(\log 10^{-5}, 10^2)$ on $\boldsymbol{\theta}_{\boldsymbol{\Sigma}}$, log–$\mathcal{N}(\log \sigma^2, 1)$ on $\boldsymbol{\theta}_{\boldsymbol{\Gamma}}$, and $\mathcal{N}(0, 5)$ on the remaining parameters.

Before training, the input and output data were both normalized to have zero means and standard deviations of one. The comparison method was trained with Adam batch optimization using the available code from the repository linked by [35]: https://github.com/GerbenBeintema/SS-encoder-WH-Silver. The batch size was reduced from 1024 to 256 to handle the smaller dataset, but the number of epochs was kept at 100,000. The time horizon was also kept at $T = 80$ since it was chosen according to the time-scale of the system, which does not change. The Bayesian method was trained for 10,000 iterations with starting points of $\{\boldsymbol{\theta}_{\mathbf{x}_0}, \boldsymbol{\theta}_{\Psi}\} \sim \mathcal{N}(\mathbf{0}, \mathbf{I} \times 10^{-1})$, $\boldsymbol{\theta}_{\boldsymbol{\Sigma}} = \mathbf{1} \times 10^{-5}$, and $\boldsymbol{\theta}_{\boldsymbol{\Gamma}} = 1$. The training time of each method is shown in Table 2. Then, 10^5 samples were drawn from the posterior, and 2×10^4 were discarded as burn-in. The initial proposal covariance matrix was $\mathbf{I} \times 10^{-5}$ for each parameter group.

4.3.2. Results

Figs. 6(a) and 6(b) show the estimated output of the MS and Bayesian estimates in the time domain during the training period and during the last 1000 iterations of the testing period, respectively. The posterior predictive distribution is represented by 100 samples drawn at regular intervals from the collected samples and simulated deterministically. 'Mean' refers to the mean of these 100 posterior predictive samples. In Fig. 6(a), the estimates look nearly identical. In Fig. 6(b), some noisiness has appeared in the MS estimate indicative of overfitting, but the Bayesian estimate remains smooth due to its inherent regularization. Figs. 6(c) and 6(d) show the differences in the estimated outputs and the unaltered benchmark data ("Original") during the testing period in the time and frequency domains. The time-domain errors are converted to the frequency domain using the discrete Fourier transform.

The MSE values of the MS and mean estimates are given in Table 3. Although the MS objective leads to smaller MSE relative to the training data, the posterior predictive mean displays smaller MSE relative to the unaltered benchmark data over both the training and testing periods. In fact, the posterior predictive mean MSE is over 8.7 times lower than that of MS. These results demonstrate that the Bayesian method is better-suited for noisy and small datasets compared to the state-of-the-art machine learning approach.

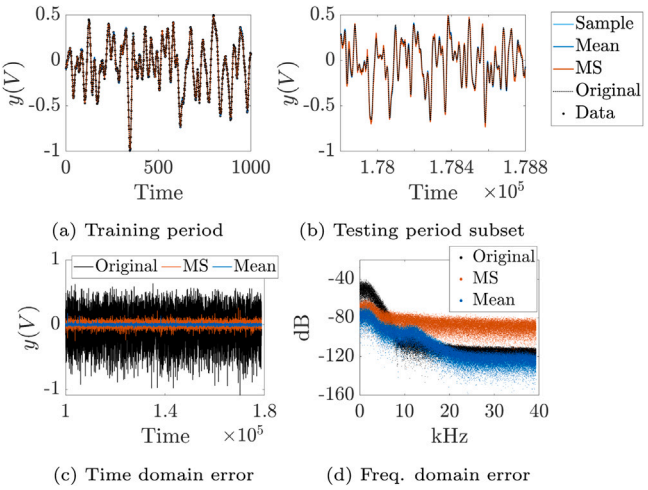


Fig. 6. Estimation results on the Wiener–Hammerstein example. The trajectory estimates of the MS and Bayesian methods are shown over the duration of the training data in Fig. 6(a) and over the duration of the last 1000 testing data in Fig. 6(b). The errors between the unaltered testing data (original) and the estimates are shown in the time domain in Fig. 6(c) and in the frequency domain in Fig. 6(d).

Table 3
MSE values of the posterior predictive mean (Bayes) and MS estimate on the Wiener–Hammerstein training data and unaltered benchmark data (original) during the training/testing periods.

	Train MSE (data)	Train MSE (original)	Test MSE (original)
Bayes	3.6965×10^{-4}	7.7813×10^{-5}	1.2546×10^{-4}
MS	1.2863×10^{-4}	1.9698×10^{-4}	1.0948×10^{-3}

4.3.3. Conclusions

In this example, we compared the Bayesian method to a method that had achieved state-of-the-art performance on a system ID benchmark problem. We demonstrated that on a small and noisy dataset, this comparison method becomes no longer state-of-the-art. This example serves to illustrate how machine learning methods often rely on large datasets to achieve good estimation and how more rigorous modeling of uncertainty can help extend these methods to datasets with fewer and noisier data.

4.4. Forced duffing oscillator

Next, the utility of the proposed Bayesian method for learning chaotic behavior will be demonstrated. For this example, the Duffing oscillator is considered. This system was also considered in one of the authors' previous works [43], but the solution was periodic and a linear model was used for estimation. This work, on the other hand, considers a chaotic solution and therefore uses a nonlinear model parameterization.

The governing equation of the Duffing oscillator is

$$\begin{bmatrix}\dot{x} \\ \ddot{x}\end{bmatrix} = \begin{bmatrix}0 & 1 \\ \alpha & \delta\end{bmatrix}\begin{bmatrix}x \\ \dot{x}\end{bmatrix} + \beta\begin{bmatrix}0 \\ x^3\end{bmatrix} + \begin{bmatrix}0 \\ 1\end{bmatrix}\gamma\cos(\omega t). \tag{24}$$

Depending on the value of the parameters, the solution of this system can be periodic or chaotic. In this example, the parameter values are $\alpha = 1$, $\delta = -0.3$, $\beta = -1$, $\omega = 1.2$, and $\gamma = 0.65$ following an example in [44] that yields chaotic behavior. The comparison methods are the MS (17) and deterministic LS (15) objectives.

4.4.1. Data generation and training

To generate the data for this problem, an initial condition of $(x, \dot{x}) = (0, 0)$ is used, and the system is simulated for 600 s before data collection to eliminate any initial transient behavior. After this initial period,

Table 4
Optimization times on the forced Duffing oscillator example for optimizing the negative log posterior (Bayes), MS objective, and deterministic LS objective (LS). The MAP was used to initialize the dynamics parameters in the optimization of the MS objective.

	Iterations	Function evaluations	Time (min:s)	Average (s/func.)
Bayes	1,000	2,758	27:16	0.59
MS	186	21,506	5:24	0.015
LS	718	93,531	23:55	0.015

Table 5
MSE values of the MAP, MS, and LS estimates over the training period from the forced Duffing oscillator. The LS estimate has the lowest MSE, but Fig. 7 shows that its behavior least resembles the true system. This shows how the LS objective is not always appropriate for system ID.

	MAP	MS	LS
Training MSE	1.6205	1.8165	0.7419

the position x of the system is measured every $\Delta t = 0.25$ s for 300 s for a total of 1200 data points, each with additive noise drawn from $\mathcal{N}(0, 10^{-6})$.

The dynamics model is the neural network architecture (23) from the previous example with latent space dimension $d_x = 2$, and the observation operator $\mathbf{H} = \begin{bmatrix}1 & 0\end{bmatrix}$ and measurement noise covariance Γ are assumed to be known. The priors are half–$\mathcal{N}(0, 10^{-4})$ on $\boldsymbol{\theta}_{\boldsymbol{\Sigma}}$, and $\mathcal{N}(0, 5)$ on the remaining parameters. The initial optimization points are $\{\boldsymbol{\theta}_{\mathbf{x}_0}, \boldsymbol{\theta}_{\boldsymbol{\Psi}}\} \sim \mathcal{N}(\mathbf{0}, \mathbf{I} \times 10^{-2})$ and $\boldsymbol{\theta}_{\boldsymbol{\Sigma}} \sim$ half–$\mathcal{N}(0, 10^{-4})$. The dynamics parameters in the MS objective optimization were initialized at the estimated MAP values. The training times are given in Table 4. For sampling, 10^6 samples are drawn and half are discarded as burn-in. The initial proposal covariance is $\mathbf{I} \times 10^{-8}$ for each parameter group. The MS objective uses a time horizon of $T = 200$. Smaller values of T in the range [30, 80] were tried but were found to give worse estimates.

4.4.2. Results

The estimates produced by the Bayesian posterior, MS, and deterministic LS are shown in Fig. 7. Figs. 7(a) and 7(b) show 25 posterior samples and the estimated MAP point simulated stochastically and deterministically, respectively. Fig. 7(c) shows the LS and MS estimates. The truth is plotted alongside the estimates in each figure for comparison. The LS estimate is clearly the worst in these figures out of the three, but the MSE of the LS estimate is actually lower than that of the MAP estimate, as shown in Table 5. This shows that for certain system ID problems, especially ones including chaotic behavior, the LS objective induces a nonsensical ranking within the model space.

Although it is difficult to identify any sort of structure in the time domain of a chaotic system, the Duffing oscillator possesses an invariant set known as an attractor in phase space. Therefore, one way to assess how similar a model is to the underlying system is by comparing the phase space of the two systems. Figs. 7(d), 7(e), and 7(f) show the phase space of the MAP model, MS model, and truth system, respectively, over 600 s. The MS and MAP models have similar shapes to the truth attractor, and both have foci near $\pm(1, 1)$ around which their outputs rotate. The MS model's attractor, however, becomes larger around its $-(1, 1)$ focus compared to both its $+(1, 1)$ focus and the truth attractor. In contrast, the structure of the MAP model visibly appears consistent with the truth attractor.

Figs. 7(g)–7(i) show the distributions of the outputs over this 600 s period. For a quantifiable comparison, we compute the first four moments of each distribution and the absolute error of each moment compared to the true moment in Table 6. The moment errors of the MAP output are on the order of 10^{-2} or smaller, whereas those of the MS output are on the order of 10^{-1}. These results suggest that the MAP estimate is a more accurate representation of the truth despite its high MSE.

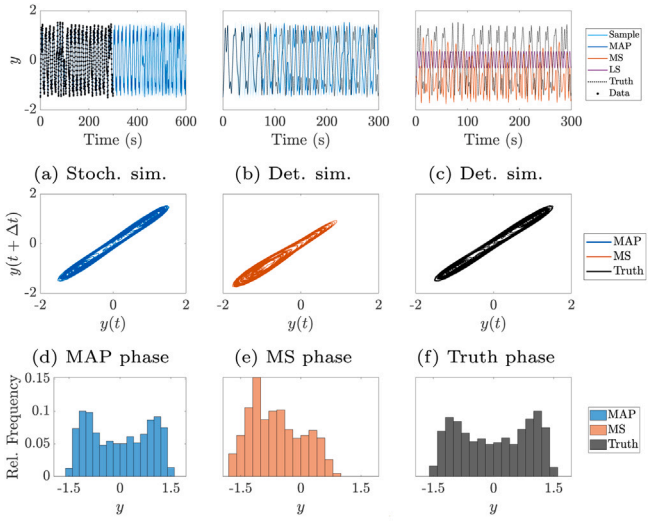


Fig. 7. Estimation results on the Duffing example. Figs. 7(a) and 7(b) compare the posterior predictive distribution and MAP to the truth, where the posterior samples and MAP are generated using stochastic and deterministic simulations, respectively, and Fig. 7(c) shows deterministic simulations of the MS and LS estimates. Figs. 7(d)–7(f) show the phase space of the MAP, MS, and truth over 600 s, and Figs. 7(g)–7(i) show the histograms of these phase space simulations.

Table 6
The first four moments of the true, MAP, and MS output distributions on the Duffing example. Error refers to the absolute difference between the estimated and true moments.

	Moment			
	1st	2nd	3rd	4th
True	0.0198	0.7871	−0.0399	1.5989
MAP	−0.0153	0.7895	0.0296	1.5807
MS	−0.6056	0.4270	0.3380	2.0541
MAP Error	0.0351	0.0024	0.0695	0.0182
MS Error	0.6255	0.3601	0.3779	0.4552

4.4.3. Conclusions

The MS objective is most commonly used to address the issue of error accumulation resulting from model errors and uncertainty. In a chaotic system, model errors grow exponentially, amplifying the need for proper handling of model uncertainty. In this example we compared the Bayesian method to an objective that handles model uncertainty through an *ad hoc* approach (MS) and an objective that does not account at all for model uncertainty (deterministic LS). We observed that although MS could yield a much better estimate than deterministic LS, the Bayesian method produced an estimate that most closely resembled the true system behavior. This example demonstrates the need for addressing model uncertainty when estimating chaotic systems. Furthermore, it shows that rigorous modeling of uncertainty leads to better results compared to the *ad hoc* MS approach.

In addition to highlighting the need for modeling model uncertainty, this example also showed how the LS metric is not always suitable for ranking model quality. The deterministic LS estimate produced the lowest MSE out of the three estimates, but the estimate behavior clearly did not resemble that of the true system. The inadequacy of deterministic LS on this example illustrates the need for other objective functions, such as the proposed Bayesian objective.

4.5. Allen–Cahn equation with forcing

In certain applications involving PDEs, one is interested not in the full-field solution, but only in certain statistics of the full field [45,46].

Table 7
Optimization times on the Allen–Cahn with forcing example for optimizing the negative log posterior (Bayes) and deterministic LS objective (LS). The estimated MAP is used as the initial point for optimizing the LS objective.

	Iterations	Function evaluations	Time (min:s)	Average (s/func.)
Bayes	1,000	4,319	27:35	0.38
LS	18	6,835	00:10	0.0015

In this experiment, the goal is to learn a dynamical model of a PDE quantity of interest (QoI) that can be used for forecasting. Continuing with a focus on non-autonomous systems, we consider an example of the Allen–Cahn equation with forcing that was used in [47]. The system uses Neumann boundary conditions, and its dynamics are given as

$$\frac{\partial}{\partial t} w(\xi, t) = \sigma \frac{\partial^2 w}{\partial \xi^2} + w(1 - w^2) + \chi_\delta(\xi) u(t), \tag{25}$$

where w is the flow, $\xi \in [-1, 1]$ is the spatial coordinate, $t \in [0, \infty)$ is the time coordinate, u is the control input, and χ is an indicator function that takes the value one when $\xi \in \delta = [-0.5, 0.2]$ and zero otherwise. The control inputs are sampled as $u(t_k) \sim \mathcal{N}(0, 10^{-2})$ for $k = 0, \ldots, n$, and a zero-order hold is assumed for intermediate time values. To assess the effects of including process noise in the model formulation, we use the deterministic LS objective (15) as a comparison method.

4.5.1. Data generation and training

To generate the data for this system, a spatial mesh with 256 cells and a time discretization with $\Delta t = 0.1$ s are used. Then at each timestep t_k, Eq. (25) is solved for $w(\xi_i, t_k)$ at each vertex ξ_i using the `solve` function in FEniCS [48]. The output of this system is the approximated second moment of the flow

$$y_k = \frac{1}{257} \sum_{i=0}^{256} w^2(\xi_i, t_k) + \eta_k, \tag{26}$$

where $\eta_k \sim \mathcal{N}(0, 0.04)$ represents sensor noise. The system is simulated using an initial condition of $w(\xi_i, 0) = 0$, $\forall i = 0, \ldots, 256$. Since the first 20 s contain a transient period where the system moves toward a stable equilibrium at ± 1, training data collection begins at $t = 20$ s. After this initial period, data are collected for 10 s at $\Delta t = 0.1$ s intervals for a total 101 data points. The next 10 s following data collection are used as a testing period.

The dynamics model used in both the Bayesian method and deterministic LS is the neural network (23) used in previous examples with $d_x = 8$. The neural network has 350 parameters and there are only 101 data points, so the system is underdetermined. The observation operator is fixed as $\mathbf{H} = \begin{bmatrix} 1 & \mathbf{0}_{1\times 7} \end{bmatrix}$. The priors are half–$\mathcal{N}(0, 10^{-6})$ on $\boldsymbol{\theta}_{\boldsymbol{\Sigma}}$, half–$\mathcal{N}(0, 1)$ on $\boldsymbol{\theta}_{\boldsymbol{\Gamma}}$, and $\mathcal{N}(0, 0.25)$ on the remaining parameters. The data were normalized before training to have zero mean and standard deviation of one. The initial optimization points for $\boldsymbol{\theta}_{\mathbf{x}_0}$ and $\boldsymbol{\theta}_{\boldsymbol{\psi}}$ were sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I} \times 10^{-2})$, and those for $\boldsymbol{\theta}_{\boldsymbol{\Sigma}}$ and $\boldsymbol{\theta}_{\boldsymbol{\Gamma}}$ were sampled from their priors. Optimization of the LS objective was initialized at the estimated MAP to aid optimization. Table 7 shows the training time of each method. In this experiment, 10^5 samples were drawn from the posterior, and half were discarded as burn-in. The initial proposal covariances were $\mathbf{I}\times 10^{-8}$, $\mathbf{I}\times 10^{-6}$, and $\mathbf{I}\times 10^{-8}$ for $\boldsymbol{\theta}_{\mathbf{x}_0}$, $\boldsymbol{\theta}_{\boldsymbol{\psi}}$, and $\{\boldsymbol{\theta}_{\boldsymbol{\Sigma}}, \boldsymbol{\theta}_{\boldsymbol{\Gamma}}\}$, respectively.

4.5.2. Results

Figs. 8(a) and 8(b) show 100 samples simulated stochastically and deterministically, respectively, and the 'mean' represents the mean of these sample trajectories. The LS estimate matches the data closely during the training period, but performs poorly beyond this period due to overfitting. In contrast, the mean provides a good estimate of the truth throughout the 20 s time period.

The root mean squared error (RMSE) values of the mean and LS estimates on the training data and on the noiseless QoI values during the training and testing periods are given in Table 8. The LS estimate has a training data RMSE two orders of magnitude smaller than that of the mean estimate, but the noiseless training QoI RMSE of the LS estimate is actually worse than that of the mean estimate, indicating overfitting. Moreover, the noiseless testing QoI RMSE of the LS estimate is an order of magnitude worse than the mean estimate. Also note that the RMSE of the mean estimate on the noiseless training and testing QoIs are very similar, indicating good generalizability of the estimate. Recall that the prior used on the dynamics parameters was only weakly informative, so the improved generalizability of the mean estimate over the LS estimate comes nearly entirely from the inclusion of process noise in the likelihood. The LS objective, on the other hand, does not account for model uncertainty and implicitly assumes that the model that most closely fits the data is the best, making it prone to overfit when the model form is very expressive, as is the case here.

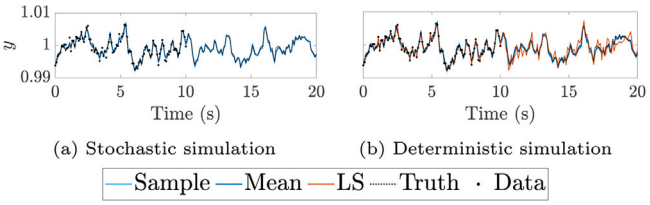


Fig. 8. The estimates of the Allen–Cahn QoI over the 10 s training period and the subsequent 10 s testing period.

Table 8
RMSE values of the posterior predictive mean (Bayes) and deterministic LS estimate (LS) on the training data and on the noiseless QoI values during the training/testing periods of the Allen–Cahn example.

	Training data RMSE	Training QoI RMSE	Testing QoI RMSE
Bayes	7.10×10^{-4}	3.74×10^{-4}	3.77×10^{-4}
LS	3.43×10^{-6}	8.18×10^{-4}	1.46×10^{-3}

4.5.3. Conclusions

This example compared the Bayesian method to the deterministic LS on an underdetermined PDE QoI estimation problem. Since the Bayesian method is asymptotically equivalent to deterministic LS as the process noise term approaches zero [1], this example was meant to isolate the effects of including process noise on estimation, especially the theoretical regularization benefits. We observed that the deterministic LS estimate fit the training data very closely but generalized poorly beyond the training period. In contrast, the Bayesian estimate incurred less than half the RMSE on the noiseless training signal as the LS estimate and yielded nearly equal RMSE on the testing period as it did on the training period. The results from this example illustrate that the inherent regularization that arises from including process noise in the dynamics is effective at resisting overfitting and improving generalization without sacrificing estimation accuracy.

5. Discussion

The numerical experiments showed that modeling uncertainty with the proposed approach leads to better generalization and more robust estimation under small and noisy training datasets. In this section, we broaden the context of our analysis of the Bayesian system ID methodology through discussion of its limitations and potential future directions.

5.1. Limitations

As with any method, the gains in performance of the Bayesian approach come at a cost. The two primary drawbacks of the proposed Bayesian system ID methodology relative to other methods is its computational cost and the difficulty of tuning the variance parameters.

In [1], the computational complexity of evaluating the marginal likelihood was analyzed. This analysis showed the complexity to be on the order of $\mathcal{O}(n(d_x^3 + d_y^3))$ for linear system ID and $\mathcal{O}(n(d_x^3 + d_y^3 + d_x(C_\Psi + C_h)))$ for nonlinear system ID, where C_Ψ and C_h represent the computational complexity of Ψ and h. In contrast, methods with simpler uncertainty modeling tend to be considerably cheaper. The LS Markov parameter estimation method admits a closed-form solution that can be computed with complexity of $\mathcal{O}(d_u^3 + n(d_u^2 + d_u d_y))$. The evaluation of many nonlinear system ID objectives, including MS and deterministic LS, only require evaluating a deterministic simulation of the model, which has a cost on the order $\mathcal{O}(n(C_\Psi + C_h))$. Since these other methods do not compute covariance matrices, they are able to scale much better to higher dimensions.

In addition to increasing computational cost, tracking covariance matrices also leads to greater memory complexity. Running a Bayesian filter necessitates evaluation of state and output covariances, resulting in a memory complexity of $\mathcal{O}(d_x^2 + d_y^2)$. MS and deterministic LS, on the other hand, track only a point estimate, which requires memory on the order of $\mathcal{O}(d_x + d_y)$. Due to its computational and memory complexities, the Bayesian method can be prohibitively expensive for high-dimensional systems.

Note, however, that the parameter dimension d_θ does not affect the orders of the computational and memory complexities of evaluating the marginal likelihood. Therefore, the Bayesian method has the potential to scale well to even more expressive model forms such as deep neural networks with thousands, if not millions, of parameters. The only additional cost as d_θ increases is the cost of gradient evaluation, but every other system ID method incurs this cost as well. Moreover, the development of differentiable programming and reverse-mode differentiation have made gradient evaluation economical for large parameter dimensions.

In addition to computational considerations, it is also important to consider how performance quality scales with parameter dimension. As d_θ increases, the posterior landscape becomes filled with more local minima and more complex correlations between parameters that pose challenges for optimization. A similar effect occurs as n increases since compositions of Ψ lead to increasingly nonlinear relationships between the parameters and outputs. As noted in the discussion in Section 4.1.2 and observed in Appendix B, sampling the output space with large d_θ may not be as problematic as it initially appears. Optimization, however, still remains a challenge. A possible solution is to use the optimization result of one of the computationally cheaper system ID methods as a warm-start for optimization.

Another possible solution for optimization is to leverage the smoothness-inducing effects of the process noise covariance. This leads into the other difficulty of the Bayesian method: tuning the variance parameters. The process noise covariance represents the model error, so its value strongly influences the quality of the learned model. Ideally, the process noise covariance is as small as possible to minimize output error on the training data. As was shown in Fig. 2, however, the smaller the process noise is, the less smooth the log likelihood becomes, making optimization and sampling challenging. For best results, the process noise covariance should start large and slowly shrink as the optimizer makes progress toward better model estimates. To encourage this trend, half-normal priors with high density near zero are placed on the variance parameters. These priors penalize models with high model error, but if they are too tight around zero, they can artificially shrink the process noise before the optimizer has moved into a region of lower model error. When this happens, the optimizer will have more difficulty moving toward better estimates because the lower process noise leads to more local minima. Developing a systematic method of shrinking the process noise covariance while keeping the posterior smooth in the neighborhood of the optimizer would have the potential to drastically improve the optimization efficiency over the negative log posterior.

5.2. Extensions to other systems

Next we consider how to extend the Bayesian method to a wider range of systems. Here we will specifically consider high-dimensional systems and stochastic systems.

As noted in the previous section, the evaluation of the marginal likelihood scales cubically with the state and observation dimensions. Therefore, to learn high-dimensional systems, the marginal likelihood must be evaluated in a reduced-dimensional space. There are various areas of research that would be useful for this endeavor, notably reduced-order modeling and manifold learning. One possible direction is to parameterize a full-order model (FOM), use reduced-order modeling to create a reduced-order model (ROM) of this FOM, evaluate the marginal likelihood with the ROM, then adjust the parameters of the FOM based on the result. Another direction would be to parameterize the ROM and then simultaneously estimate the ROM and a mapping from the reduced-dimensional space back to the full-dimensional space. For these approaches, a main challenge would be modeling how uncertainty is transformed when passing between the low- and high-dimensional spaces.

Another class of systems that the Bayesian method could potentially be applied to is stochastic systems. Since the HMM formulation (1) already uses a stochastic dynamics model, this seems like a natural extension. The main challenge here would be disambiguating between the aleatoric randomness of the stochastic dynamics and the epistemic uncertainty of the model form. If they both enter into the dynamics in the same form, e.g., stationary additive noise, they may be indistinguishable.

6. Conclusion

Theoretical and experimental comparisons were made between the Bayesian system ID method of [1] and two other system ID algorithms: the Markov parameter estimation approach of [22] and the multiple shooting objective. It was shown that the objectives of these two approaches are special cases of the posterior derived from the HMM formulation. Specifically, the Markov parameter estimation approach implicitly assumes the data are conditionally independent, and the multiple shooting approach performs joint parameter-state estimation under the assumption of zero model error. It was then shown on a number of numerical experiments that these underlying assumptions incur performance costs compared to the negative log posterior, especially when the data are noisy, sparse, and/or from a chaotic system. For example, the posterior predictive mean yielded 8.7 times lower MSE compared to multiple shooting in one example.

CRediT authorship contribution statement

Nicholas Galioto: Conceptualization, Investigation, Software, Formal analysis, Writing – original draft. **Alex Arkady Gorodetsky:** Conceptualization, Methodology, Writing – review & editing, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Links to all data and code are provided within the text of this manuscript.

Acknowledgment

This work was funded by the AFOSR Computational Mathematics Program, USA (P.M Fariba Fahroo).

Appendix A. Proof of Proposition 3

Our goal is to show that the assumptions introduced in Proposition 2 that lead to equivalency hold asymptotically. That is, we want to show $\lim_{\bar{n}\to\infty}\sum_{i=\bar{n}}^{k}\mathbf{G}_i\mathbf{u}_{k-i} = \mathbf{0}$ and $\lim_{\bar{n}\to\infty}\mathbf{A}^k\boldsymbol{\Sigma}(\mathbf{A}^k)^* = \mathbf{0}$ for $k \geq \bar{n}$. Let $N = k - \bar{n} + 1$ be the number of terms in the sum. We assume N is bounded such that it cannot grow arbitrarily large.

To show $\lim_{\bar{n}\to\infty}\sum_{i=\bar{n}}^{k}\mathbf{G}_i\mathbf{u}_{k-i} = \mathbf{0}$, it suffices to show $\lim_{\bar{n}\to\infty}\sum_{i=\bar{n}}^{k}|\mathbf{G}_i[j,:]\mathbf{u}_{k-i}| = 0$ for any $j \in \{1,\dots,d_y\}$. By assumption, the inputs $\mathbf{u}_i[j] \in \mathbb{R}$ are independent realizations of the random variable $\mathbf{u}$ for $i = 0, 1, \dots$ and $j = 1,\dots,d_u$. Recall that for any real-valued random variable z, $\lim_{a\to\infty}\int_{-a}^{a}p(z)\mathrm{d}z = 1$, where the integral represents the probability that $|z| < a$. This implies that for any $0 < \varepsilon < 1$, $\exists\mathbf{a} \in \mathbb{R}^{d_u}$ such that $|\mathbf{u}[j]| < \mathbf{a}[j]$ for $j = 1,\dots,d_u$ with probability (w.p.) $1-\varepsilon$. Then, upper and lower bounds can be established as

$$0 \leq \sum_{i=\bar{n}}^{k}\left|\mathbf{G}_i[j,:]\mathbf{u}_{k-i}\right| < \sum_{i=\bar{n}}^{k}\left|\mathbf{G}_i[j,:]\right|\mathbf{a}, \tag{A.1}$$

w.p. at least $(1-\varepsilon)^N$. Next we prove that the upper bound goes to zero as $\bar{n}\to\infty$ regardless of $\mathbf{a}$ by showing that $\lim_{\bar{n}\to\infty}\sum_{i=\bar{n}}^{k}\mathbf{G}_i[j,:] = \mathbf{0}$. Recall that when $\rho(\mathbf{A}) < 1$, the LTI system is exponentially stable. Specifically, there exist constants $c > 0$ and $\lambda \in (0,1)$ such that $\|\mathbf{A}^k\mathbf{x}_0\|_2 \leq c\lambda^k\|\mathbf{x}_0\|_2$, $\forall\mathbf{x}_0 \in \mathbb{R}^{d_x}$. We can therefore bound the norm of the columns of $\mathbf{G}_i$ as follows:

$$\begin{aligned}\|\mathbf{G}_i[:,j]\|_2 &= \|\mathbf{H}\mathbf{A}^{i-1}\mathbf{B}[:,j]\|_2\\ &\leq \|\mathbf{H}\|_2\|\mathbf{A}^{i-1}\mathbf{B}[:,j]\|_2\\ &\leq \|\mathbf{H}\|_2 c\lambda^{i-1}\|\mathbf{B}[:,j]\|_2.\end{aligned} \tag{A.2}$$

Noting that the quantity $c\|\mathbf{H}\|_2\|\mathbf{B}[:,j]\|_2$ is constant, we see that $\|\mathbf{G}_i\|_2$ is bounded above by an exponentially decaying function of timestep i. This leads us to the following bound on the norm of the sum:

$$\begin{aligned}\left\|\sum_{i=\bar{n}}^{k}\mathbf{G}_i[:,j]\right\|_2 &\leq \sum_{i=\bar{n}}^{k}\|\mathbf{G}_i[:,j]\|_2\\ &\leq \sum_{i=\bar{n}}^{k}c\lambda^{i-1}\|\mathbf{H}\|_2\|\mathbf{B}[:,j]\|_2\\ &\leq Nc\lambda^{\bar{n}-1}\|\mathbf{H}\|_2\|\mathbf{B}[:,j]\|_2.\end{aligned} \tag{A.3}$$

Since $0 < \lambda < 1$, $\lim_{\bar{n}\to\infty}\|\sum_{i=\bar{n}}^{k}\mathbf{G}_i[:,j]\|_2 = 0$, and consequently $\lim_{\bar{n}\to\infty}\sum_{i=\bar{n}}^{k}\mathbf{G}_i[:,j] = \mathbf{0}$ as well, each w.p. at least $(1-\varepsilon)^N$. The variable ε is arbitrary, and as a result $\lim_{\bar{n}\to\infty}\sum_{i=\bar{n}}^{k}\mathbf{G}_i\mathbf{u}_{k-i} \to \mathbf{0}$ w.p. 1. Moreover, the rate of convergence of the upper bound can be found as follows:

$$\lim_{\bar{n}\to\infty}\frac{Nc\lambda^{\bar{n}}\|\mathbf{H}\|_2\|\mathbf{B}[:,j]\|_2}{Nc\lambda^{\bar{n}-1}\|\mathbf{H}\|_2\|\mathbf{B}[:,j]\|_2} = \lim_{\bar{n}\to\infty}\frac{\lambda^{\bar{n}}}{\lambda^{\bar{n}-1}} = \lambda. \tag{A.4}$$

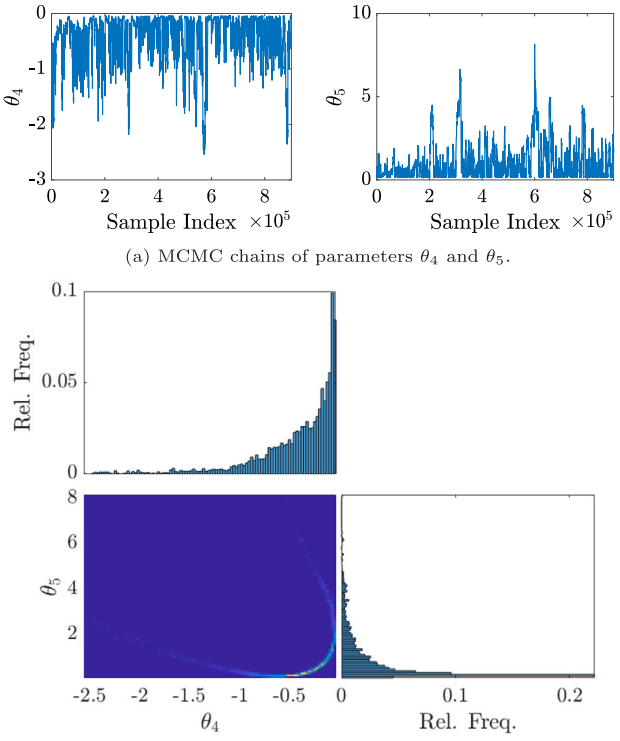
Therefore, the upper bound converges to zero linearly with rate λ, and the sum $\sum_{i=\bar{n}}^{k}\mathbf{G}_i\mathbf{u}_{k-i}$ must converge at least as fast. When $\mathbf{A}$ is diagonalizable, λ can be chosen to be $\rho(\mathbf{A})$ such that the convergence rate is bounded above by the maximum eigenvalue of $\mathbf{A}$.

The proof for the covariance term is similar. To show $\lim_{\bar{n}\to\infty}\mathbf{A}^{\bar{n}}\boldsymbol{\Sigma}(\mathbf{A}^{\bar{n}})^* = \mathbf{0}$, we begin by decomposing the covariance matrix $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}^{\frac{1}{2}}(\boldsymbol{\Sigma}^{\frac{1}{2}})^*$. Then, the norm $\mathbf{A}^{\bar{n}}\boldsymbol{\Sigma}(\mathbf{A}^{\bar{n}})^*$ is bounded above as follows:

$$\|\mathbf{A}^{\bar{n}}\boldsymbol{\Sigma}(\mathbf{A}^{\bar{n}})^*\|_2 \leq \left\|\mathbf{A}^{\bar{n}}\boldsymbol{\Sigma}^{\frac{1}{2}}\right\|_2\left\|\left(\mathbf{A}^{\bar{n}}\boldsymbol{\Sigma}^{\frac{1}{2}}\right)^*\right\|_2 = \left\|\mathbf{A}^{\bar{n}}\boldsymbol{\Sigma}^{\frac{1}{2}}\right\|_2^2. \tag{A.5}$$

It suffices to show $\lim_{\bar{n}\to\infty}\mathbf{A}^{\bar{n}}\boldsymbol{\Sigma}^{\frac{1}{2}}[:,j] = \mathbf{0}$ for columns $j \in \{1,\dots,d_x\}$. To begin, we derive an upper bound:

$$\left\|\mathbf{A}^{\bar{n}}\boldsymbol{\Sigma}^{\frac{1}{2}}[:,j]\right\|_2^2 \leq \left(c\lambda^{\bar{n}}\left\|\boldsymbol{\Sigma}^{\frac{1}{2}}[:,j]\right\|_2\right)^2. \tag{A.6}$$



(a) MCMC chains of parameters θ_4 and θ_5.

(b) Marginal distributions of θ_4, θ_5, and their joint.

Fig. B.9. Parameter posteriors in the linear pendulum example.

The fact that $\lim_{\bar{n}\to\infty}c^2\lambda^{2\bar{n}}\|\boldsymbol{\Sigma}^{\frac{1}{2}}[:,j]\|_2^2 = 0$ implies that $\lim_{\bar{n}\to\infty}\mathbf{A}^{\bar{n}}\boldsymbol{\Sigma}^{\frac{1}{2}}[:,j] = \mathbf{0}$. Lastly, the rate of convergence of the upper bound is as follows:

$$\lim_{\bar{n}\to\infty}\frac{c^2\lambda^{2\bar{n}}\left\|\boldsymbol{\Sigma}^{\frac{1}{2}}[:,j]\right\|_2^2}{c^2\lambda^{2(\bar{n}-1)}\left\|\boldsymbol{\Sigma}^{\frac{1}{2}}[:,j]\right\|_2^2} = \lim_{\bar{n}\to\infty}\frac{\lambda^{2\bar{n}}}{\lambda^{2\bar{n}-2}} = \lambda^2. \tag{A.7}$$

Then $\|\mathbf{A}^{\bar{n}}\boldsymbol{\Sigma}^{\frac{1}{2}}\|_2^2$, and consequently $\|\mathbf{A}^{\bar{n}}\boldsymbol{\Sigma}(\mathbf{A}^{\bar{n}})^*\|_2$, converges at least as fast. Therefore, we have shown that $\sum_{i=\bar{n}}^{k}\mathbf{G}_i\mathbf{u}_{k-i}$ and $\mathbf{A}^{\bar{n}}\boldsymbol{\Sigma}(\mathbf{A}^{\bar{n}})^*$ both converge to zero as $\bar{n}\to\infty$ with rate no greater than λ.

Appendix B. MCMC posterior distributions

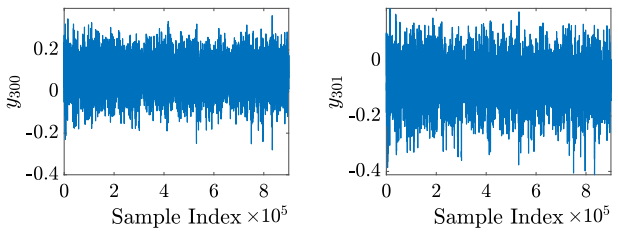
In the discussion in Section 4.1.2, we noted that it is possible to have poor mixing in the parameter space while still achieving good mixing in the output space. In this appendix, we show examples of parameter and output MCMC samples from the examples in Section 4.

For every example, we choose two parameters from $\boldsymbol{\theta}_\psi$ and two output variables. We plot the Markov chain of each of these variables to show the quality of the mixing, and we also plot 1D and 2D histograms to show the MCMC estimates of the marginal posterior distributions and reveal any correlations between the variables.

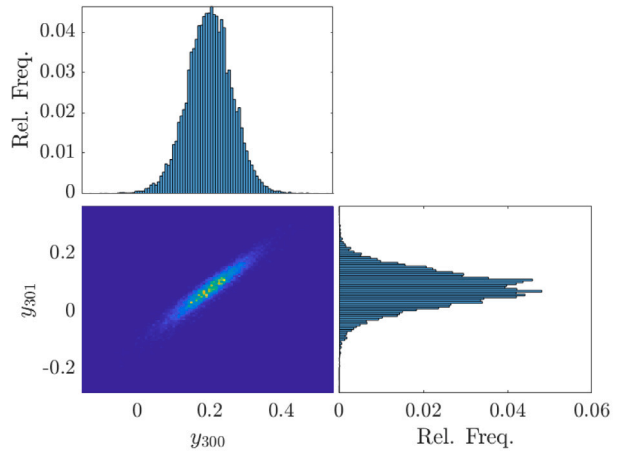
B.1. Linear pendulum with control

We consider the high noise, low sparsity case with a (noise, timestep) pair of (0.20, 0.1) from Figs. 5(a) and 5(c). We plot the samples of parameters θ_4 and θ_5 from the $\mathbf{A}$ matrix in Fig. B.9 and the samples of outputs y_{300} and y_{301} corresponding to times $t_{300} = 30.0$ s and $t_{301} = 30.1$ s in Fig. B.10. Although only 100 samples were used in Fig. 5, we include all 900,000 here.

In this example, the posterior is well-sampled in both the parameter and output spaces. By achieving convergence in the parameter space,

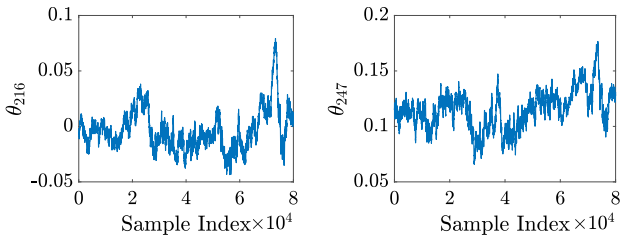


(a) MCMC chains of outputs y_{300} and y_{301}.

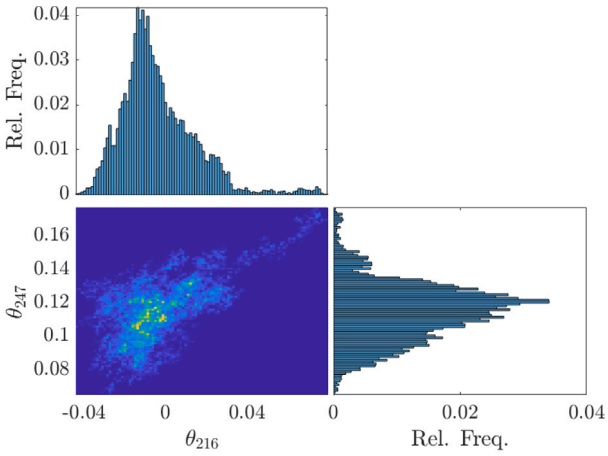


(b) Marginal distributions of y_{300}, y_{301}, and their joint.

Fig. B.10. Output posteriors in the linear pendulum example.

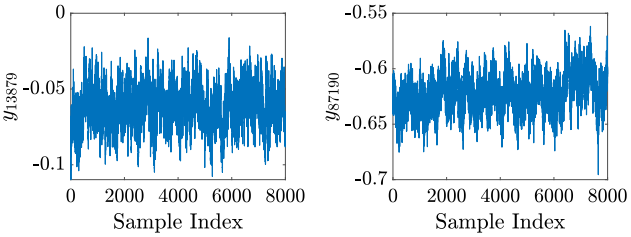


(a) MCMC chains of parameters θ_{216} and θ_{247}.

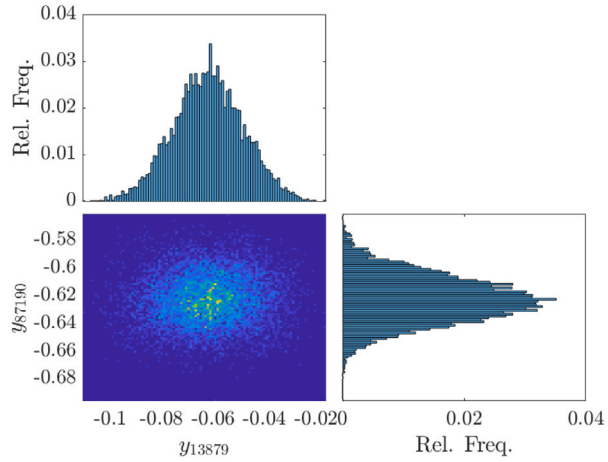


(b) Marginal distributions of θ_{216}, θ_{247}, and their joint.

Fig. B.11. Parameter posteriors in the Wiener–Hammerstein example.



(a) MCMC chains of outputs y_{13879} and y_{87190}.



(b) Marginal distributions of y_{13879}, y_{87190}, and their joint.

Fig. B.12. Output posteriors in the Wiener–Hammerstein example.

we are able to observe non-identifiability in the parameters. This non-identifiability is evident from the curve seen in the 2D marginal of Fig. B.9. Additionally, we observe the correlation in the outputs in Fig. B.10 that arises due to the fact that the outputs are nearby in time. The Pearson correlation coefficient between these two outputs is 0.9531, indicating high correlation. Recall that the LS method (12) neglects this correlation during estimation.

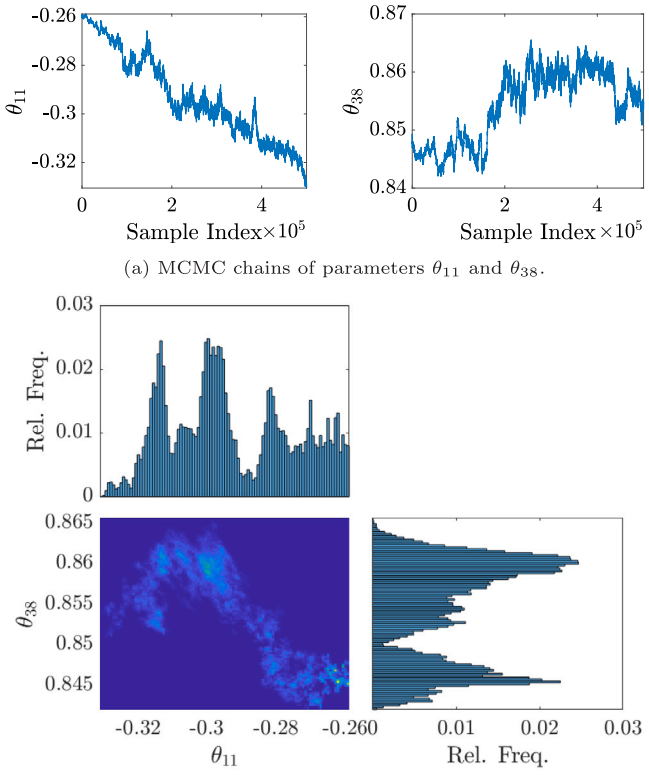
B.2. Wiener–Hammerstein benchmark

For this example, we randomly choose two parameters from $\boldsymbol{\theta}_{\psi}$ and two output time indices. The first parameter is θ_{216}, and the second parameter is θ_{247}. These parameters come from the $\mathbf{b}_2$ and $\mathbf{A}_3$ components of the neural network (23), respectively. The MCMC samples of these parameters are shown in Fig. B.11. The randomly-selected output variables are y_{13879} and y_{87190}. Both of these outputs lie in the time period between the training and testing periods. The samples of these outputs are shown in Fig. B.12. In these plots, all 80,000 parameter samples are included, but, due to memory and computational constraints, only every 10th output sample is plotted for a total of 8000 samples.

The parameter chains in Fig. B.11 are indicative of poor mixing. Despite this observation, the output chains in Fig. B.12 appear well-mixed. These observations are aligned with the findings in [40] that good mixing can be achieved in the output space without good mixing in the parameter space.

B.3. Forced Duffing oscillator

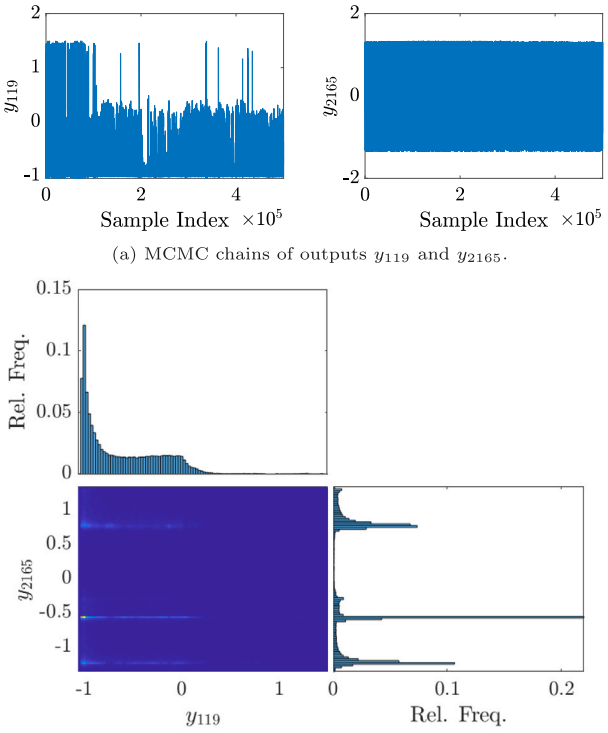
Again, we randomly select parameters from $\boldsymbol{\theta}_{\psi}$ and the outputs. The parameters are θ_{11} from $\mathbf{A}_1$ and θ_{38} from $\mathbf{A}_2$. The outputs are y_{119} and y_{2165} corresponding to times $t_{119} = 29.75$ s (within the training period) and $t_{2165} = 541.25$ s (within the testing period). The full 500,000 samples of these parameters and outputs are shown in Figs. B.13 and B.14. The parameter chains in Fig. B.13 take relatively few large jumps and resemble random walks as a result. Despite this behavior, we



(a) MCMC chains of parameters θ_{11} and θ_{38}.

(b) Marginal distributions of θ_{11}, θ_{38}, and their joint.

Fig. B.13. Parameter posteriors in the Duffing example.



(a) MCMC chains of outputs y_{119} and y_{2165}.

(b) Marginal distributions of y_{119}, y_{2165}, and their joint.

Fig. B.14. Output posteriors in the Duffing example.

observe in Fig. B.14 that the sampler frequently jumps between modes in the output space.

Interestingly, the output distribution in Fig. B.14 is multi-modal with three distinct modes. To get a better understanding of this behavior, we find a posterior sample near each mode and plot each one in the time domain, in phase space, and as a histogram in Fig. B.15. The modes are labeled such that their y_{2166} values are ordered from least to greatest. Table B.9 shows the value of y_{2166} of each sample as well as the value of y_{2166} produced by the parameter MAP for context. This table shows that the MAP point most closely belongs to the first mode. Note that the output from the parameter MAP point does not necessarily coincide with the output MAP point.

To assess how the output distribution of each mode sample compares to the true output distribution, the first four moments of each output sample and their absolute errors compared to the true moments are provided in Table B.10. This table shows that the output distribution produced by the mode 1 sample best matches the true distribution. In fact, this sample matches better than the MAP moments from Table 6. Table B.10 also shows that the output distribution produced by the mode 3 sample is the worst match, though it is still better than the MS estimate from Table 6.

For a final piece of context, we compare the unnormalized log posterior values of each mode sample to that of the MAP in Table B.11. We observe that the ranking of the log posterior values does not directly correspond to how well each sample matches the output statistics of the system. Rather, the posterior identifies probable models based on the available data, and these models tend to statistically resemble the underlying system, though not necessarily.

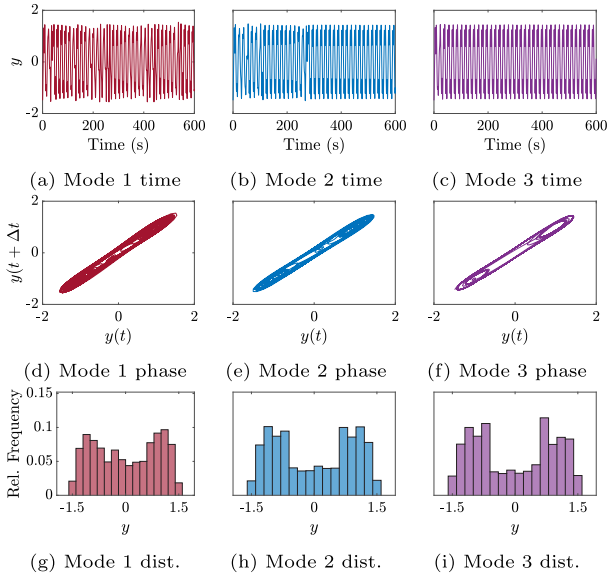


Fig. B.15. Output samples from each mode in the Duffing example. Figs. B.15(a)–B.15(c) show deterministic simulations of each mode sample in the time domain, Figs. B.15(d)–B.15(f) show these simulations in phase space, and Figs. B.15(g)–B.15(i) show the histogram of each simulated output. The truth is plotted similarly in Fig. 7.

B.4. Allen–Cahn equation with forcing

In the final example, the randomly-selected parameters are θ_{190} from $\mathbf{A}_2$ and θ_{300} from $\mathbf{A}_3$. The outputs are y_{87} and y_{167} corresponding to times $t_{87} = 8.7$ s (within the training period) and $t_{167} = 16.7$ s (within the testing period). All 50,000 parameter and output samples are shown in Figs. B.16 and B.17. Similar to Appendix B.2, the output chains are well-mixed and the output marginals are unimodal despite poor mixing in the parameter chains.

Table B.9
The value of y_{2166} produced by the parameter MAP and mode samples from the Duffing example. The MAP point belongs to the first mode.

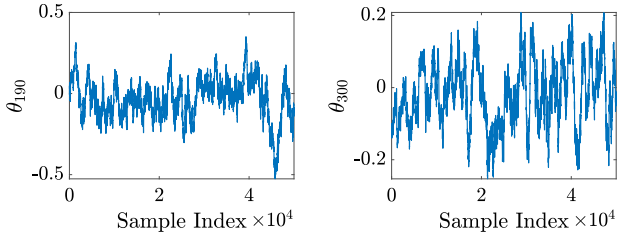
	MAP	Mode 1	Mode 2	Mode 3
y_{2166}	−1.1302	−1.2393	−0.5606	0.7683

Table B.10
The first four moments of the output distributions produced by each of the mode samples from Fig. B.14 and their absolute errors relative to the true moments.

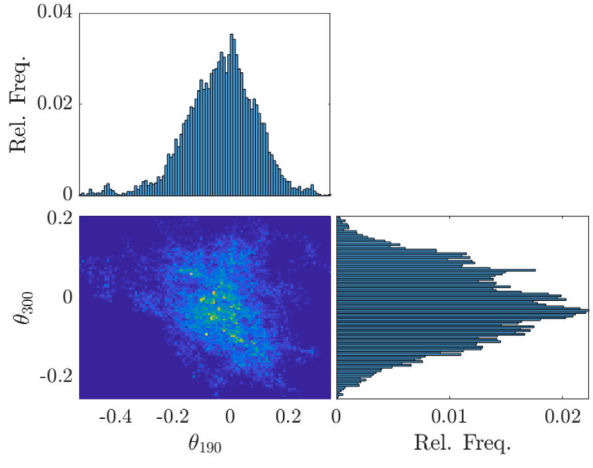
	Moment			
	1st	2nd	3rd	4th
True	0.0198	0.7871	−0.0399	1.5989
Mode 1	0.0206	0.8005	−0.0411	1.5937
Mode 2	0.0081	0.8529	−0.0166	1.4813
Mode 3	−0.0066	0.8708	0.0105	1.4746
Mode 1 Error	0.0007	0.0134	0.0012	0.0052
Mode 2 Error	0.0117	0.0658	0.0233	0.1176
Mode 3 Error	0.0264	0.0837	0.0504	0.1243

Table B.11
The log of unnormalized posterior values of the parameter MAP and mode samples in the Duffing example. Note that within any mode, the log posterior values can vary greatly. This table is not intended to rank the modes, only to give additional context to the mode samples.

	MAP	Mode 1	Mode 2	Mode 3
$\log p(\theta\|\mathcal{Y}_n)$ $\times 10^3$	7.5128	1.9559	3.3398	1.4298

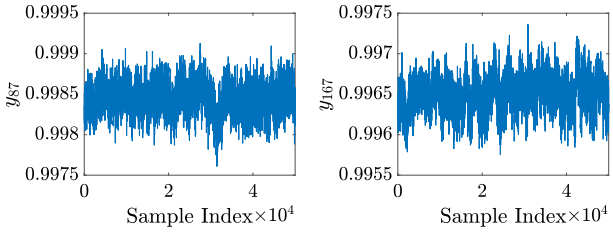


(a) MCMC chains of parameters θ_{190} and θ_{300}.

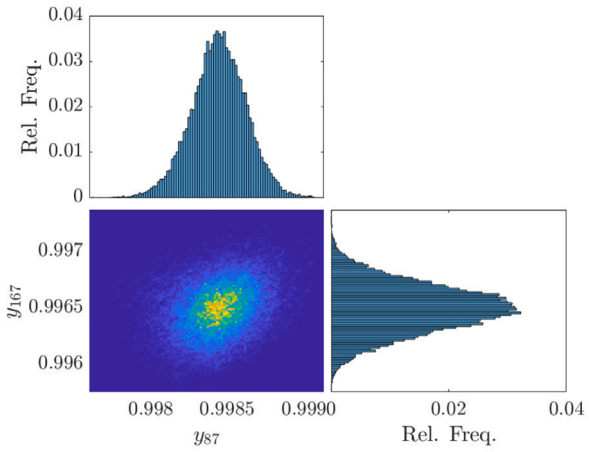


(b) Marginal distributions of θ_{190}, θ_{300}, and their joint.

Fig. B.16. Parameter posteriors in the Allen–Cahn example.



(a) MCMC chains of outputs y_{87} and y_{167}.



(b) Marginal distributions of y_{87}, y_{167}, and their joint.

Fig. B.17. Output posteriors in the Allen–Cahn example.

References

[1] N. Galioto, A.A. Gorodetsky, Bayesian system ID: Optimal management of parameter, model, and measurement uncertainty, Nonlinear Dynam. 102 (2020) 241–267.

[2] S. Klus, F. Nüske, S. Peitz, J.-H. Niemann, C. Clementi, C. Schütte, Data-driven approximation of the Koopman generator: Model reduction, system identification, and control, Physica D 406 (2020) 132416.

[3] K.Y. Ng, M.M. Gui, COVID-19: Development of a robust mathematical model and simulation package with consideration for ageing population and time delay for control action and resusceptibility, Physica D 411 (2020) 132599.

[4] X. Jia, J. Willard, A. Karpatne, J. Read, J. Zwart, M. Steinbach, V. Kumar, Physics guided RNNs for modeling dynamical systems: A case study in simulating lake temperature profiles, in: Proceedings of the 2019 SIAM International Conference on Data Mining, SIAM, 2019, pp. 558–566.

[5] J.L. Proctor, S.L. Brunton, J.N. Kutz, Dynamic mode decomposition with control, SIAM J. Appl. Dyn. Syst. 15 (1) (2016) 142–161.

[6] S.L. Brunton, J.L. Proctor, J.N. Kutz, Sparse identification of nonlinear dynamics with control (SINDYc), IFAC-PapersOnLine 49 (18) (2016) 710–715, 10th IFAC Symposium on Nonlinear Control Systems NOLCOS 2016.

[7] J.-N. Juang, R.S. Pappa, An eigensystem realization algorithm for modal parameter identification and model reduction, J. Guid. Control Dyn. 8 (5) (1985) 620–627.

[8] H.G. Bock, K.-J. Plitt, A multiple shooting algorithm for direct solution of optimal control problems, IFAC Proc. Vol. 17 (2) (1984) 1603–1608.

[9] L. Piroddi, W. Spinelli, An identification algorithm for polynomial NARX models based on simulation error minimization, Internat. J. Control 76 (17) (2003) 1767–1781.

[10] S. Kirkpatrick, C.D. Gelatt Jr., M.P. Vecchi, Optimization by simulated annealing, Science 220 (4598) (1983) 671–680.

[11] P.L. Green, Bayesian system identification of a nonlinear dynamical system using a novel variant of simulated annealing, Mech. Syst. Signal Process. 52 (2015) 133–146.

[12] H.G. Bock, Numerical treatment of inverse problems in chemical reaction kinetics, in: Modelling of Chemical Reaction Systems, Springer, 1981, pp. 102–125.

[13] J.H. Tu, C.W. Rowley, D.M. Luchtenburg, S.L. Brunton, J.N. Kutz, On dynamic mode decomposition: Theory and applications, J. Comput. Dyn. 1 (2) (2014).

[14] V. Peterka, Bayesian approach to system identification, in: Trends and Progress in System Identification, Elsevier, 1981, pp. 239–304.

[15] R.J. Elliott, L. Aggoun, J.B. Moore, Hidden Markov Models: Estimation and Control, vol. 29, Springer Science & Business Media, 2008.

[16] E.T. Jaynes, Probability Theory: The Logic of Science, Cambridge University Press, 2003.
[17] S. Särkkä, Bayesian Filtering and Smoothing, vol. 3, Cambridge University Press, 2013.
[18] S.J. Julier, J.K. Uhlmann, New extension of the Kalman filter to nonlinear systems, in: Signal Processing, Sensor Fusion, and Target Recognition VI, vol. 3068, International Society for Optics and Photonics, 1997, pp. 182–193.
[19] A. Tsiamis, G.J. Pappas, Finite sample analysis of stochastic system identification, in: 2019 IEEE 58th Conference on Decision and Control, CDC, IEEE, 2019, pp. 3648–3654.
[20] T. Sarkar, A. Rakhlin, M.A. Dahleh, Finite time LTI system identification, J. Mach. Learn. Res. 22 (26) (2021) 1–61.
[21] Y. Zheng, N. Li, Non-asymptotic identification of linear dynamical systems using multiple trajectories, IEEE Control Syst. Lett. 5 (5) (2020) 1693–1698.
[22] S. Oymak, N. Ozay, Non-asymptotic identification of LTI systems from a single trajectory, in: 2019 American Control Conference, ACC, IEEE, 2019, pp. 5655–5661.
[23] Y. Zhu, Multivariable System Identification for Process Control, Elsevier, 2001.
[24] G.R. Yoo, H. Owhadi, Deep regularization and direct training of the inner layers of neural networks with kernel flows, Physica D 426 (2021) 132952.
[25] G.A. Gottwald, S. Reich, Supervised learning from noisy observations: Combining machine-learning techniques with data assimilation, Physica D 423 (2021) 132911.
[26] R.T. Chen, Y. Rubanova, J. Bettencourt, D.K. Duvenaud, Neural ordinary differential equations, Adv. Neural Inf. Process. Syst. 31 (2018).
[27] Z. Long, Y. Lu, X. Ma, B. Dong, PDE-net: Learning PDEs from data, in: International Conference on Machine Learning, PMLR, 2018, pp. 3208–3216.
[28] C. Rackauckas, Y. Ma, J. Martensen, C. Warner, K. Zubov, R. Supekar, D. Skinner, A. Ramadhan, A. Edelman, Universal differential equations for scientific machine learning, 2020, arXiv preprint arXiv:2001.04385.
[29] S.L. Brunton, J.L. Proctor, J.N. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proc. Natl. Acad. Sci. 113 (15) (2016) 3932–3937.
[30] S. Greydanus, M. Dzamba, J. Yosinski, Hamiltonian neural networks, Adv. Neural Inf. Process. Syst. 32 (2019).
[31] P. De Valpine, A. Hastings, Fitting population models incorporating process noise and observation error, Ecol. Monograph 72 (1) (2002) 57–76.
[32] L.A. Aguirre, B.H. Barbosa, A.P. Braga, Prediction and simulation errors in parameter estimation for nonlinear systems, Mech. Syst. Signal Process. 24 (8) (2010) 2855–2867.
[33] A.H. Ribeiro, K. Tiels, J. Umenberger, T.B. Schön, L.A. Aguirre, On the smoothness of nonlinear system identification, Automatica 121 (2020) 109158.
[34] D. Masti, A. Bemporad, Learning nonlinear state–space models using autoencoders, Automatica 129 (2021) 109666.
[35] G. Beintema, R. Toth, M. Schoukens, Nonlinear state-space identification using deep encoder networks, in: Learning for Dynamics and Control, PMLR, 2021, pp. 241–250.
[36] Y.D. Zhong, B. Dey, A. Chakraborty, Symplectic ODE-Net: Learning Hamiltonian dynamics with control, in: International Conference on Learning Representations, 2020.
[37] J. Schoukens, J. Suykens, L. Ljung, Wiener-Hammerstein benchmark, in: Proc. of the 15th IFAC Symposium on System Identification, SYSID 2009, 2009.
[38] R.H. Byrd, J.C. Gilbert, J. Nocedal, A trust region method based on interior point techniques for nonlinear programming, Math. Programming 89 (2000) 149–185.
[39] H. Haario, M. Laine, A. Mira, E. Saksman, DRAM: Efficient adaptive MCMC, Stat. Comput. 16 (4) (2006) 339–354.
[40] P. Izmailov, S. Vikram, M.D. Hoffman, A.G.G. Wilson, What are Bayesian neural network posteriors really like? in: International Conference on Machine Learning, PMLR, 2021, pp. 4629–4640.
[41] F. Draxler, K. Veschgini, M. Salmhofer, F. Hamprecht, Essentially no barriers in neural network energy landscape, in: J. Dy, A. Krause (Eds.), Proceedings of the 35th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, vol. 80, PMLR, 2018, pp. 1309–1318.
[42] T. Garipov, P. Izmailov, D. Podoprikhin, D.P. Vetrov, A.G. Wilson, Loss surfaces, mode connectivity, and fast ensembling of DNNs, in: S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Eds.), in: Advances in Neural Information Processing Systems, vol. 31, Curran Associates, Inc., 2018.
[43] N. Galioto, A.A. Gorodetsky, A new objective for identification of partially observed linear time-invariant dynamical systems from input-output data, in: Learning for Dynamics and Control, PMLR, 2021, pp. 1180–1191.
[44] D. Jordan, P. Smith, Nonlinear Ordinary Differential Equations: An Introduction for Scientists and Engineers, vol. 10, Oxford University Press on Demand, 2007.
[45] G. Migliorati, F. Nobile, E. von Schwerin, R. Tempone, Approximation of quantities of interest in stochastic PDEs by the random discrete L^2 projection on polynomial spaces, SIAM J. Sci. Comput. 35 (3) (2013) A1440–A1460.
[46] J.E. Castrillon-Candas, F. Nobile, R.F. Tempone, Analytic regularity and collocation approximation for elliptic PDEs with random domain deformations, Comput. Math. Appl. 71 (6) (2016) 1173–1197.
[47] S. Dolgov, D. Kalise, K.K. Kunisch, Tensor decomposition methods for high-dimensional Hamilton–Jacobi–Bellman equations, SIAM J. Sci. Comput. 43 (3) (2021) A1625–A1650.
[48] A. Logg, K.-A. Mardal, G. Wells, Automated Solution of Differential Equations by the Finite Element Method: The FEniCS Book, vol. 84, Springer Science & Business Media, 2012.