Estimating Varying Parameters in Dynamical Systems: A Modular Framework Using Switch Detection, Optimization, and Sparse Regression

A Preprint

Jamiree Harrison*
Department of Mechanical Engineering
University of California, Santa Barbara
Santa Barbara, California
jamiree@ucsb.edu

Enoch Yeung
Department of Mechanical Engineering
University of California, Santa Barbara
Santa Barbara, California
eyeung@ucsb.edu

December 17, 2024

Abstract

The estimation of static parameters in dynamical systems and control theory has been well studied, and much headway has been made in sub-classes of problems dealing with the estimation of varying parameters in particular types of systems. Suppose for the general case, that we have data from a system which has parameters that depend on some independent variable such as time or space. Further, suppose that we know the form of the system's dynamics or model *a priori*, but we wish to identify functions that describe the parameter-varying elements of the model as these elements change with respect to time or some other system variable. Before tackling the entire class of problems, we first consider the case where parameters are discretely switching piece-wise constant functions. For this sub-class of problems, we devise an algorithmic framework for the detection of these discrete parameter switches and the fitting of a piece-wise constant parameter-varying model to the data using optimization-based parameter estimation. One of the benefits of our framework is that it is modular in that the switch detection, numerical integration, and optimization sub-steps can be specified to meet user requirements. Secondly, we demonstrate our framework's capabilities across several examples including a time-varying promoter-gene expression model, the time-varying genetic toggle switch, a parameter-switching manifold, a model with a mixture of fixed and switching parameters, a model in which parameters are not switching uniformly, the heat equation with a time-varying coefficient of diffusion, and the advection-diffusion equation with a continuously varying parameter. For these demonstrations, we use binary segmentation for switch detection and employ the Nelder-Mead and Powell methods for optimization. Third, we characterize the robustness to measurement noise of algorithms in our framework. Finally, to handle the broad class of problems mentioned above, we show how dictionary-based sparse regression with trigonometric and polynomial dictionary functions can be used in conjunction with our algorithm to obtain continuously varying parameter functions.

Keywords parameter estimation, parameter-varying systems, curve fitting, optimization, switch detection, regression

1 Introduction

Parameter estimation is a fundamental inverse problem which finds applications in control theory, statistics, systems biology, and physics, to name just a few. Estimating a system's parameters often precedes the prediction of that system's future states, the understanding of that system's dynamics, and the structure of the given system. Obtaining a system's parameters in certain instances is a key step in the accurate control of that system. To exemplify the estimation of a system's parameters for its accurate control, the work of Ryan and Ichikawa [1] shows how this can be done in the context of a linear distributed system with static parameters that get updated estimates. Furthermore, parameter estimation has been shown to provide benefits in addressing model discrepancies in control systems. For example, Burns et al. [2] address the accurate control of systems with delays, model errors, and measurement errors by employing hierarchical modeling to estimate uncertain disturbances, offering an alternative to Bayesian analysis.

Parameter estimation plays a critical role in diverse fields, particularly in the expansive domain of quantitative biology, which serves as the primary motivation for this paper. As several examples in this work focus on biological models, we provide a literature review of how parameter estimation has been applied in quantitative biology. Notably, none of the studies reviewed address the specific problem we aim to solve: the estimation of parameters that vary with the evolution of a given system.

Mitra and Hlavacek offer a comprehensive review of parameterization and uncertainty quantification in mathematical models of immunoreceptor signaling and other biological processes. They highlight techniques such as gradient-based estimation, profile likelihood, bootstrapping, and Bayesian inference, emphasizing their applications in immune system modeling [3]. Penas et al. introduced saCeSS, a novel parallel optimization method for parameter estimation in large-scale kinetic models. This approach demonstrated robust performance, significantly reduced computation times, and user-friendly self-tuning, with potential applications in systems biology and whole-cell dynamic modeling [4].

Zhan and Yeung proposed two innovative parameter estimation methods that combine spline theory with Linear and Nonlinear Programming. These methods offer robust, efficient, and flexible solutions for identifying parameters in systems biology models without the need for ODE solvers [5]. Valderrama-Bahamóndez and Fröhlich compared Markov Chain Monte Carlo (MCMC) techniques for estimating rate parameters in non-linear ODE systems used in systems biology. They found that parallel adaptive MCMC, combined with informative Bayesian priors, outperformed other methods for high-dimensional, multi-modal parameter distributions [6]. Giampiccolo et al. introduced a hybrid Neural ODE framework for parameter estimation and identifiability assessment in biological models with partial structural knowledge. Their approach integrates hyperparameter tuning with *a posteriori* identifiability analysis to address challenges posed by neural networks, demonstrating effectiveness on noisy, real-world-inspired benchmarks [7].

For additional examples in quantitative biology, Ashyraliyev et al. provide a mini-review exploring the applications of mathematical models in biology. They focus on parameter inference, model identifiability, and commonly used methods for parameter space exploration [8]. Beyond quantitative biology, parameter estimation has been successfully applied in fields like astrophysics. For instance, Douspis et al. [9] and Bailer-Jones [10] have utilized parameter estimation across various models and datasets, demonstrating its versatility and effectiveness.

The problem we aim to solve in this paper is at the interface dynamical systems theory and parameter estimation. Dynamical systems in which the parameters vary as the system evolves are known as parameter-varying systems, and these parameters can change continuously or switch discretely. Systems that are modeled with the prior assumption of having static parameters could often be better characterized, better controlled, and more accurately captured via parameter-varying models, and our paper serves as a novel avenue in reaching these goals. Accurately estimating the varying parameters for general classes of systems remains a nontrivial task. In this paper, our aim is to address this challenge.

1.1 Definitions

We first need to define what constitutes a discrete parameter switch in a parameter-varying system, as well as clarify what it means for a parameter to vary continuously.

Definition 1. ***Continuously Varying Parameter***

*We say that $p(t) : \mathbb{R} \to \mathbb{R}$ is a **continuously varying parameter** if $p(t)$ is continuous and not a constant function.*

Definition 2. ***Discrete Parameter Switch***

Let n_p be defined as the number of parameters in a given system. Suppose that we have piece-wise parameters $p(t) : \mathbb{R} \to \mathbb{R}^{n_p}$, given by

$$p(t) = \begin{cases} p_1(t) & \textit{if} \quad t \in [t_0, t_1] \\ p_2(t) & \textit{if} \quad t \in (t_1, t_2] \\ & \vdots \\ p_n(t) & \textit{if} \quad t \in (t_{n-1}, t_n] \end{cases} \tag{1}$$

*where $n \in \mathbb{N}$ and $p_i(t)$ are continuously varying . If $p_k(t_k) \neq p_{k+1}(t_k)$ for a given $k \in \{1, ..., n-1\}$, then we say that a **discrete parameter switch** has occured at t_k. The number of discrete parameter switches is the **discrete switch number** N_{s_d}. These **discrete switch locations** which we will denote by t_{k_d} are jump discontinuities of $p(t)$.*

We say that a ***continuous parameter switch*** *occurs at* ***continuous switch locations*** *denoted by t_{k_c} at values of t where $p(t)$ is continuous but not differentiable. Explicitly, a continuous switch location occurs if $p_k(t) \neq p_{k+1}(t)$ for some $t \in [t_{k-1}, t_{k+1}] \setminus t_k$, but $p_k(t_k) = p_{k+1}(t_k)$. The number of continuous parameter switches is the* ***continuous switch number*** *N_{s_c}. The* ***total switch number*** *is defined as $N_s := N_{s_c} + N_{s_d}$.*

Remark 1. *Suppose that we have a dynamical system $\frac{dX}{dt} = f(X, p(t))$, and suppose that f is continuous with respect to t if $p(t)$ is continuous (without discrete switches).*

If we have $\frac{dX}{dt} = f(X, p(t))$ such that $p(t)$ represents piece-wise parameters with discrete parameter switches at t_k (and thus, jump discontinuities) as described in Definition (2), then f has jump discontinuities at t_k and $X(t)$ is non-differentiable at t_k.

The above remark expresses that the discrete parameter switches of a parameter-varying system manifest in a given trajectory, $X(t)$, as non-differentiable points in t. In a data-driven empirical setting where we are given discrete data points, X_{data}, we cannot analytically determine where these parameter switch locations are, so we leverage the data-driven switch detection methods that are available in the literature (see Section 2.3) and we extend their implementations to detect switching parameters of dynamical systems.

1.2 General Problem Formulation (continuous-time systems)

Assume that we have data denoted $X_{data} \in \mathbb{R}^{m \times N}$ from a parameter-varying system denoted

$$\frac{dX}{dt} = f(X, p(t)) \tag{2}$$

with $m \in \mathbb{N}$ states, $N \in \mathbb{N}$ data points, vector field $f : \mathbb{R}^m \to \mathbb{R}^m$, and $n_p \in \mathbb{N}$ parameters $p(t) : \mathbb{R} \to \mathbb{R}^{n_p}$.

Further, assume we know the form of the system's model, $f(X, p(t))$ but we do not know the varying parameters $p(t)$. Lastly, assume that the parameters, $p(t)$, are piece-wise functions.

Problem: Identify $p(t)$ given X_{data}, i.e. find the minimizing parameters such that

$$p(t) = \mathbf{argmin} \| X_{data} - X_{model}(p(t)) \|_2 \tag{3}$$

where $X_{model} \in \mathbb{R}^{m \times N}$ is given by $X_{model} = \int_{\text{numerical}} \frac{dX}{dt} dt$.

Remark 2. *Here, the specified numerical integration scheme (ODE solver) is presupposed in Assumption 2 such that it will grant sufficiently accurate X_{model} data in terms of $\|X_{model} - X(t_d)\|_2 < \epsilon$ for some desired $\epsilon > 0$ where $X(t_d)$ is the corresponding discretization of $X(t)$, the solution to the system. This problem formulation accounts for partial differential equations (PDEs) as shown in Section (2.2) and discrete-time parameter-varying systems are accounted for in the supplement.*

This problem formulation encompasses several sub-problems, each addressed with varying degrees of success in the literature. While these contributions demonstrate rigor and specificity in tackling their respective challenges, our framework addresses the general class of problems holistically, filling critical gaps. Additionally, we acknowledge that the specialized methods discussed in the literature may outperform our framework in certain domains and should be employed when appropriate. However, our approach effectively covers many use cases within the largely unexplored space of the broader problem class.

For example, Wang et al. proposed a method for switch detection and robust identification in slowly switched Hammerstein systems [11]. Zheng et al. developed a technique for identifying fast-switched linear-nonlinear alternating subsystems, particularly in high-speed train models [12]. Yu et al. [13] and Bianchi et al. [14] introduced randomized methods to address switched ARX and NARX systems, SISO systems, and other non-autonomous systems. Li et al. created methods for identifying nonlinear Markov jump systems [15]. These methods stand in contrast to the recent data-driven system identification and control developments utilizing Koopman operator theory frameworks [16, 17, 18].

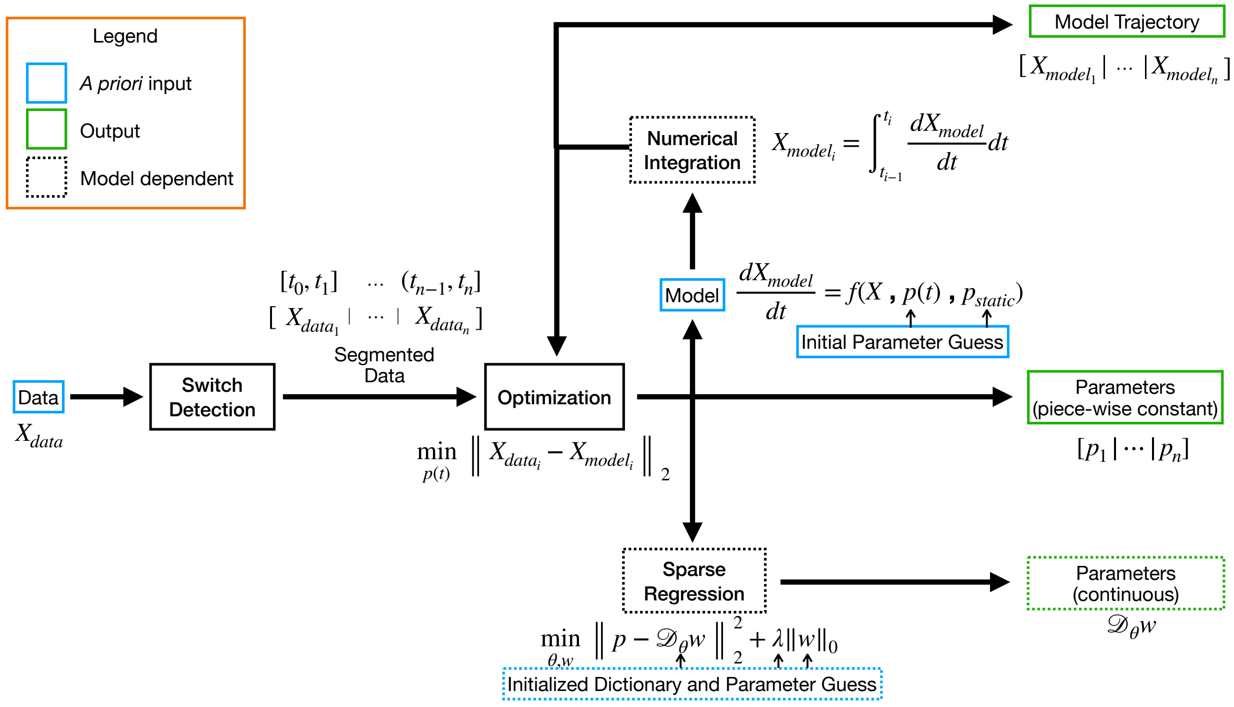
Kon et al. [19] presented an input-output linear parameter-varying (LPV) feedforward parameterization technique and a corresponding data-driven estimation method, leveraging neural networks to model the dependency of coefficients on scheduling signals. Xu et al. developed an algorithm for parameter estimation in time-varying systems with invariant matrix structures [20]. Similarly, Luiz and Reinaldo et al. tackled parameter estimation in linear, continuous, time-varying dynamical systems, assuming known coefficient matrices, measurable states, and bounded piecewise continuous parameters [21].

Johnson et al. [22] applied hybrid gradient descent to estimate unknown constant parameters in hybrid systems with flow and jump dynamics affine in the parameters. Qian et al. used Bayesian parameter estimation for discrete-time linear parameter-varying finite impulse response systems [23]. Altaie et al. [24] demonstrated how parameter estimation in parameter-varying systems can inform the design of robust controllers for multidimensional systems.

While accurately detecting a system's changing parameters has been achieved for certain classes of systems as shown above, our paper presents a generalized framework for estimating the parameters of systems with piece-wise constant parameters and continuously varying parameters. Further, we showcase implementations of algorithms within our framework that successfully handle several examples including a time-varying promoter-gene expression model, the time-varying genetic toggle switch, a parameter-switching manifold, a model with a mixture of fixed and switching parameters, a model in which parameters are not switching uniformly, and the advection-diffusion equation with a time-varying coefficient of advection. Our paper contributes to the body of literature in that it provides a framework to estimate the changing parameters of parameter varying dynamical systems using an approach that utilizes the combination of switch detection, optimization, and sparse dictionary regression.

2 Methodology for algorithmic framework

To give a broad overview of our framework, let us define $X_{data} \in \mathbb{R}^{m \times N}$ as the input data from our system of interest, and $X_{model} \in \mathbb{R}^{m \times N}$ as the predictive model data that approximates X_{data} upon completion of an algorithm within our framework. These are matrices with m being the number of states and N being the number of data points (or snapshots) such that a row of either X_{data} or X_{model} is a state trajectory. The overview of our framework is given in Figure 1 .



Legend
A priori input
Output
Model dependent
Data
X_{data}
Switch Detection
$[t_0, t_1]$ ··· $(t_{n-1}, t_n]$
$[\, X_{data_1} | \cdots | X_{data_n} \,]$
Segmented Data
Optimization
$\min_{p(t)} \| X_{data_i} - X_{model_i} \|_2$
Model
$\frac{dX_{model}}{dt} = f(X, p(t), p_{static})$
Initial Parameter Guess
Numerical Integration
$X_{model_i} = \int_{t_{i-1}}^{t_i} \frac{dX_{model}}{dt} dt$
Model Trajectory
$[X_{model_1} | \cdots | X_{model_n}]$
Parameters (piece-wise constant)
$[p_1 | \cdots | p_n]$
Sparse Regression
$\min_{\theta, w} \| p - \mathcal{D}_\theta w \|_2^2 + \lambda \|w\|_0$
Initialized Dictionary and Parameter Guess
Parameters (continuous)
$\mathcal{D}_\theta w$

Figure 1: Outline of algorithmic framework. The data (left) get segmented by a switch detection algorithm. The presupposed model (middle-right) is then fit to each data interval via minimization of the model trajectory error (middle-left). If the model is a differential equation, then the numerical integration sub-step (upper-middle) must take place in the optimization loop (middle), but numerical integration need not take place if the model is a parametric equation in the form of $X_{model} = f(X, p(t))$. Optimal parameters and model trajectories are then given as output (right). Sparse regression is used if the parameters of the model are pre-supposed to be continuously varying as opposed to being discretely switching piece-wise constant parameters.

The framework in Figure (1) starts with the initialized continuous-time model, then detects switches in the input data and then segments the data into n intervals. From here, model parameters $p(t)$ and p_{static} are identified to fit the model to the input data over each segmented time interval. This is done by numerically integrating the segmented dynamic model to obtain X_{model_i} and then using numerical optimization to minimize the distance (norm) between X_{data_i} and X_{model_i}

where $i \in \{1, ..., n\}$. Numerical integration is used for differential equation models in the form of $\frac{dX}{dt} = f(X, p(t))$ and is not used for parametric curve models in the form of $X_{model} = f(X, p(t))$. If the parameters were presupposed to be piece-wise constant, then our algorithm outputs them as such. If the parameters were presupposed to be continuously varying functions, then we assume a high number of switches to have occurred in the data and use dictionary-based sparse regression to fit continuous functions to the parameters. Finally, the identified model parameters are used to generate the model trajectory, X_{model} that accurately approximates X_{data}. Being that this trajectory is an approximation of the N sequential snapshots given in X_{data}, these are also known as N-step predictions.

From here it is important to note that this framework works for ODEs and PDEs that are written to have a first order derivative in time. In the context of numerical methods for ODE solving, this means that the ODE must be in standard form. In the next section of this paper, we go into more detail about how to handle PDEs using the method of lines. Far more details are given in Section (2.2) that elucidate the minutia of every step of the framework. Further, we showcase and provide code in the supplement for our given implementations, but it should not be understated that given the modular structure of the code, that a user can swap out the switch detection, optimization, and numerical integration schemes for preferred methods. Further, our framework is easily modulated to the estimate the varying parameters of a discrete time system (found in supplement).

Proposition 1. ***Convergence of Algorithms in framework.***

An algorithm specified under our framework will converge if the following sub-steps are convergent:

1. Switch detection / data segmentation

2. Numerical integration of parameterized model

3. Minimization of parameterized model error

Proof. Note that we have a finite number of processes as sub-steps in our algorithmic framework. Each sub-step is assumed to converge to either a locally or globally optimal solution depending on the method chosen. The proof is then trivial in that we have a finite sequence of convergent processes, so the sequence of processes itself must converge to either a locally or globablly optimal solution. □

Remark 3. *For locally optimal solutions to be granted under this framework, at least one of the sub-steps listed in Theorem 1 must be locally optimal. For globally optimal solutions to be granted under this framework (over finite domains), all sub-steps listed in Theorem 1 must be globally optimal. If global optimization algorithms are used over a finite domain of possible parameters, then globally optimal parameters are guaranteed over that domain, however, these globally optimal parameters may not be unique because other distinct minima of the specified non-convex objective function, $\|X_{model} - X_{data}\|_2$, may exist.*

2.1 Evaluative metrics

Now that we have established our algorithmic framework, we formalize the metrics to be used for the evaluation of such algorithms. The parameter error E_p is defined by the distance between a system's set of true parameters over the discrete data points, $p(t_d)$, and the approximate set of parameters of that system, $\hat{p}$, normalized by the number of parameters, n_p. The trajectory error E_t is defined by the distance between the input data matrix, X_{data}, and the data matrix generated by the given model with estimated parameters, X_{model}, normalized by the number of data points N. From context, it is clear that the trajectory error E_t is the objective function minimized in our framework. The switch number error E_{N_s} is the difference between the true number of switches, N_s, and those detected, $\hat{N}_s$. Note that if E_{N_s} is positive, then the number of switches detected is an underestimate and if E_{N_s} is negative, then the number of switches detected is an overestimate. In this way, the sign of E_{N_s} can be used to quantify if a given algorithm systematically under-reports or over-reports the correct number of switches in data. To measure the distance from the true switches, $\{t_1, ..., t_{n-1}\}$, to the detected switch estimates, $\{\hat{t}_1, ..., \hat{t}_{\hat{n}-1}\}$, we use the Hausdorff metric, H_s which measures the worst error among all of the detected switches and is explicitly given by

$$H_s := \max\{\max_k \min_l |t_k - \hat{t}_l|, \max_k \min_l |\hat{t}_k - t_l|\} \tag{4}$$

Remark 4. *For explicit clarity, the following would generally be unknown in a practical scenario before any algorithm is applied: the true parameters $p(t_d)$, the true number of switches N_s, and the locations of the switches t_k. The point of our framework and the goal of solving the previously stated problem is to infer what these unknowns are. We provide these metrics so that algorithms suited to our stated problem-class can have their performances bench-marked when applied to datasets and systems for which $p(t_d)$, N_s, and t_k are known.*

Error Type	Description	Formula
Parameter error	gap between true parameters and estimated parameters	$E_p := \frac{\|p(t_d)-\hat{p}\|_2}{n_p}$
Trajectory error	gap between input data and predicted trajectory from model	$E_t := \frac{\|X_{data}-X_{model}\|_2}{N}$
Switch number error	difference between true number of switches and those detected	$E_{N_s} := N_s - \hat{N}_s$
Switch distance	the worst error among all of the detected switches	H_s : (Equation 4)

Table 1: Metrics for evaluation of algorithm performance.

We now briefly explain our choice of the switch number error E_{N_s} and the Hausdorff metric H_s over other metrics commonly reviewed in the literature [25]. The precision and recall metrics, for instance, declare a change point as "detected" if at least one computed change point is within a "margin" of data points from it [25]. However, these metrics require the introduction of an additional hyperparameter, "margin," which is undesirable for our use case, as it already involves a significant number of user-defined parameters. Another alternative is the Rand index, which measures the similarity between two segmentations as the proportion of agreement between their partitions [26]. While this metric has merit, we chose the Hausdorff metric over the Rand index because we find the distance between true and detected switches to be a more intuitive and interpretable measure than a similarity score between the sets of true and detected switches. Ultimately, we recognize that the choice of metrics is inherently subjective. Nevertheless, we assert that E_{N_s}, in combination with H_s, provides an effective and practical means of evaluating errors in our specific context.

2.2 Model Instantiation for PDEs

Our framework is compatible with ODEs and parametric curves, however, it only is able fit varying-parameters to PDEs once the user has instantiated the a given PDE as a system of ODES. This can be done using the method of lines [27] which we illustrate on the 1D heat equation shown below.

$$\frac{\partial}{\partial t}u(t,x) = D\frac{\partial^2}{\partial x^2}u(t,x), \quad D > 0 \tag{5}$$

where $u(t,x)$ is the heat at a given spatio-temporal instance, D is the dissipation coefficient. To numerically solve this PDE, we would need initial conditions and boundary conditions, so let us define some for illustration purposes.

$$\text{Initial conditions:} \qquad u(0,x) = u_0(x) \tag{6}$$

$$\text{Boundary conditions:} \qquad u(t,x=0) = u(t,x=L) = 0 \quad . \tag{7}$$

Let us now choose a uniform grid spacing in x of integer $m > 0$ such that $m = \frac{1}{\Delta x}$ where Δx is the space between x_i and x_{i+1} for $i = 1, ..., m$. Now we must choose a method to approximate $\frac{\partial^2}{\partial x^2}u$. The second order central difference approximation gives us

$$\frac{\partial^2}{\partial x^2}u(t,x_i) \approx \frac{1}{\Delta x^2}(u(t,x_{i+1} - 2u(t,x_i + u(t,x_{i-1}) \tag{8}$$

such that our indexed system of ODEs is

$$\frac{d}{dt}u(t,x_i) \approx \frac{D}{\Delta x^2}[u(t,x_{i+1} - 2u(t,x_i) + u(t,x_{i-1}] \tag{9}$$

and written in matrix form as it would appear in the code of our algorithm is

$$\begin{bmatrix} \frac{d}{dt}u_1 \\ \frac{d}{dt}u_2 \\ \vdots \\ \vdots \\ \vdots \\ \frac{d}{dt}u_{m-1} \end{bmatrix} = \frac{D}{\Delta x^2} \begin{bmatrix} -2 & 1 & 0 & \dots & \dots & 0 \\ 1 & -2 & 1 & \ddots & & \vdots \\ 0 & 1 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 1 & 0 \\ \vdots & & \ddots & 1 & -2 & 1 \\ 0 & \dots & \dots & 0 & 1 & -2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ \vdots \\ \vdots \\ u_{m-1} \end{bmatrix}. \tag{10}$$

From here, we can use any standard ODE solver for this system of coupled ODEs. It may be possible to forego the use of the method of lines in this way and instead opt for the use of other PDE Solvers, but this is a significant challenge in that one would have to sacrifice the modular structure of the algorithmic framework which is suited for ODEs and parametric curves.

2.3 Detection of parameter switches in data

Switch detection (commonly referred to as change point detection) in noisy signals remains challenging, and the optimal method for a given dataset depends on the signal type, noise level, and number of expected changes. One popular method, Bayesian Online Change Point Detection (BOCPD), employs a Bayesian framework to sequentially update the probability of change points. It excels in handling noisy signals by incorporating noise into the model, making it useful for online detection in streaming data [28]. However, BOCPD can be computationally intensive and sensitive to prior assumptions. Pruned Exact Linear Time (PELT), another widely-used method, minimizes a cost function plus a penalty term to avoid over-fitting to noise, offering scalability to large datasets and flexibility in cost function choice. Its performance depends on a penalty term, making it suitable for large datasets with unknown change points [29]. For signals with high noise levels, Total Variation Regularization (TVR) effectively detects change points while simultaneously denoising, though it may smooth out small changes [30]. Cumulative Sum (CUSUM) detects mean shifts by monitoring the cumulative sum of deviations from a reference point, with robust variants for noisy data, though it is less effective for variance or distributional changes [31]. Dynamic Programming (Exact Segmentation) offers an exact solution by minimizing the cost function over all possible segmentations but is computationally expensive and requires the number of switches to be known *a priori*, making it more suitable for offline detection [32]. Hidden Markov Models (HMMs) are ideal for time series with hidden states and can model transitions as change points, though selecting the number of hidden states and computational cost are concerns [33]. Kernel-Based Methods, which map data to high-dimensional spaces, excel at detecting complex, non-linear changes but require parameter tuning and can be computationally intensive [34]. Lastly, Wavelet-Based Methods analyze signals at multiple scales, naturally denoising while detecting change points, making them suitable for noisy data with multiscale structures [35]. The choice of method depends on the type of change, noise level, and whether online or offline detection is needed. For low-to-moderate noise, PELT or BOCPD are recommended. For high noise, TVR or Wavelet-Based Methods work well, while for complex signals, Kernel Methods or HMMs are preferable.

In the field of switch detection, different cost functions are used to tailor the detection method to the specific characteristics of the data. L_1 and L_2 cost functions are basic measures used for minimizing absolute and squared residuals, respectively, where L_1 is more robust to outliers, while L_2 is sensitive to large deviations. The Normal cost from [29] assumes data follows a normal distribution and detects changes in mean and variance. More complex cost functions using Radial Basis Functions (RBFs) and sinusoids detect changes in data patterns by comparing similarity measures, with RBFs suitable for non-linear changes, and sinusoids focusing on angular shifts in data [36]. Cost functions for data with linear relationships and continuous linear relationships account for gradual linear changes over time [34]. Cost functions which rank data values are robust to non-Gaussian noise [37]. Maximum likelihood cost functions adapt to various statistical models for flexible change detection [38] . Finally, auto-regressive cost functions identify changes in time series data with auto correlation by capturing shifts in time dependencies [39]. Every cost function mentioned can be selected based on the specific structure and noise characteristics of the data.

For the implementation of switch detection in our framework for the examples in this paper, we used binary segmentation (`binseg` which is a recursive algorithm that was devised to detect switches in noisy data [40]. `Binseg` is sequential in nature, in that it first detects one change point, or switching point, in a given input data signal, then it splits the data at this change point, and this process is recrusively repeated on the two resulting sub-signals until no more change points are detected in all sub-signals [41].

Our implementation of binseg uses two hyperparameters where SD_{algo} below is binseg in our examples.

$$\sigma - \begin{cases} \text{standard deviation of noise,} & \text{if number of switches, } N_s \text{, is unknown,} \\ \text{number of switches } N_s, & \text{if number of switches, } N_s \text{, is known.} \end{cases} \tag{11}$$

$$s_g - \text{switch gap : minimum number of data points between detected switches.} \tag{12}$$

Algorithm 1: Switch Detection Over All States of a System's Model

```
Input: $X_{data} \in \mathbb{R}^{m \times N}$: data matrix
$t \in \mathbb{R}^N$: independent variable
$s_g \in \mathbb{N}$: switch gap threshold
$SD_{cost}$: switch detection cost function model
$SD_{algo}$: switch detection algorithm
$\sigma > 0$ or $\sigma = N_s \in \mathbb{Z}_{\geq 0}$: noise level or fixed number of switches
Output: $N_s \in \mathbb{Z}_{\geq 0}$: number of switches detected
$\hat{t}_k$: switch locations (indices of $t$ where $k \in \{1, ..., N_s\}$)
1  Function Switch_Detect($X_{data}$, $t$, $s_g$, $\sigma$, $SD_{cost}$, $SD_{algo}$):
2      $(m, N) \leftarrow$ shape($X_{data}$);
3      $\hat{t}_k \leftarrow \emptyset$;                      // Initialize true switch locations across states
4      for $j \leftarrow 1$ to $m$ do
5          $\hat{t}_k \leftarrow SD_{algo}(X_{data}[j,:], SD_{cost}, \sigma)$;      // Detected and sorted switches for state $j$
6          for $i \leftarrow 1$ to length($\hat{t}_k$) do
7              if $\hat{t}_k[i] > \hat{t}_k[i-1] + s_g$ then
8                  result $\leftarrow$ append($\hat{t}_k[i]$);          // Applies $s_g$ for state j
9          if result[end] $== N$ then
10             result $\leftarrow$ delete(result, end);       // Last entry of $t$ cannot be a switch location
11         $\hat{t}_k \leftarrow$ result;
12         for each entry $\in \hat{t}_k$ do
13             if entry $\notin \hat{t}_k$ then
14                 $\hat{t}_k \leftarrow$ append(entry) ;       // Removes redundant switches across states
15             $\hat{t}_k \leftarrow$ sort($\hat{t}_k$);
16     if length($\hat{t}_k$) $== 0$ then
17         $N_s \leftarrow 0$;
18         return $\hat{t}_k, N_s$ ;                      // If no switches detected, return
19     if length($\hat{t}_k$) $> 0$ then
20         for $i \leftarrow 1$ to length($\hat{t}_k$) do
21             if $\hat{t}_k[i] > \hat{t}_k[i-1] + s_g$ then
22                 result $\leftarrow$ append($\hat{t}_k[i]$) ;       // Applies $s_g$ across states
23         $\hat{t}_k \leftarrow$ result;
24     if $\sigma == N_s$ and $\sigma \sim= length(\hat{t}_k)$ then
25         throw error: "detected $N_s \sim=$ expected $N_s$"
26     $N_s \leftarrow$ length($\hat{t}_k$);
27     return $\hat{t}_k, N_s$;
```

Already existing as a feature of binseg, the user must input $\sigma > 0$ which is the standard deviation of the noise of the input signal. This noise level can be statistically estimated or inferred by the user as a pre-processing step. Alternatively, the user may also iteratively try different σ values until a desired model accuracy is achieved, or the user can smooth the data with a smoothing algorithm of one's choice such as the Savitzky-Golay filter [42], FIR methods [43], or

Whittaker-Henderson smoother [44]. The lower σ is, the more sensitive the switch detection is, meaning that more switches will typically be detected for a given signal. This is intuitive because `binseg` does not want to mistake noisy "jumps" for a switch in the signal. There is also the case where the user may already know how many switches there are in the given input signal, and does not need `binseg` to estimate the number of switches. If this is the case, the user may specify the number of switches to `binseg` as $\sigma = N_s$ and `binseg` will simply find the switch locations. Note that σ is the most important hyperparameter in determining the number of switches and their locations in given signals.

We denote the "switch gap" as $s_g \in \mathbb{N}$ which does not allow there to be two parameter switches to be within s_g discrete data points from each other. This was implemented to allow some necessary user discretion when applying switch detection methods. To expand, several methods such as `binseg` may detect switches at consecutive time points and at other intervals that are too small for multiple switches to have occurred for certain classes of problems. Simply setting $s_g = 1$ allows for consecutive data points be detected as switches.

For our implementation of `binseg`, all states (observed input signals) of an ODE or PDE are searched individually for the correct number of parameter switches as opposed to searching over all of the states at once. We do this by running `binseg` on each state's signal, collecting every switch that occurs, and removing switches that have already been counted. For example, suppose we have a two states of a system given by x_1 and x_2. If it is determined that state x_1 experienced 2 switches at t_1 and t_2 and state x_2 experienced the same 2 switches, but also 2 more completely different switches at t_3 and t_4, then we say the system has 4 switches in total, namely 2 shared (coupled) switches, and 2 unique (uncoupled) switches in the second state (For an example of this, see Equation (35). We show that this implementation guarantees the correct number of switches with accuracy in Proposition (2), thus laying the foundation for the use of switch detection algorithms on systems with discretely switching parameters.

Proposition 2. *The implementation of switch detection in Algorithm 1 detects the correct number of parameter switches, N_s, with accuracy ($E_{N_s} = 0$ & $H_s = 0$) for a dynamical system $\frac{dX}{dt} = f(X, p(t))$ where $X \in \mathbb{R}^m$ if the SD_{algo} selected grants accuracy for each state in X.*

Proof. Another way to phrase the proposition is that a system's parameter switches, t_k, will be equal to those detected, $\hat{t}_k$, if the specified SD_{algo} detects all parameter switches across all states in X. Let us then assume that the *a priori* selected SD_{algo} grants accuracy for each state in X. In other words, all parameter switches that occur in a given state are accurately detected and accounted for. Note that t_k is a set of unique indices in which particular snapshots of X_{data} indicate a switching parameter, and note that parameters can switch independently from state-to-state. If any of these parameter switches occur at the same locations across states, then our method removes these redundant parameter switches from the set of $\hat{t}_k$ so that each parameter switch is unique. So, if each state contains a set of parameter switches, then our algorithm returns of the union of the sets respectively obtained from each state. Thus, we have that our method exhaustively accumulates all unique switches across all states with accuracy. Hence, we have shown that $\hat{t}_k = t_k$. □

Remark 5. *If one was to naively use a switch detection algorithm directly on X_{data} without the nuances of our implementation, then there would be severe errors E_{N_s} & H_s in several cases such as the example given in Section (3.3) where the parameters switch at non-uniform intervals. Ultimately, since systems with multiple states can have parameters that switch distinctly and independently across those states, the necessity of our dynamical systems switch detection implementation is clear. Hence, this proposition provides the backing for the implementation of data-driven parameter switch detection in dynamical systems.*

2.4 Optimization schemes, known static parameters, parameter bounds, and constraints

Upon experimenting with several numerical schemes to minimize our desired objective function. Our best results came from robust, non-gradient based, deterministic optimization schemes, namely, the Nelder-Mead method [45] and the conjugate-based Powell's method [46]. Auto-differentiation based methods and stochastic methods could often give comparable results, but not as consistently as the deterministic methods aforementioned. The lack of consistency using auto-differentiation-based methods is most likely due to the combination of the general non-convex nature of objective functions and the stochastic elements of the methods. It may be possible to get improved performance with auto-differentiation-based methods or in general from adding regularization to our objective function, but this is yet to be explored. In [47] global optimization schemes are strongly recommended for handling the inherent non-convexity of these data-driven problems, especially in the context of biological systems. With this insight, it is important to note that the modular nature of our framework allows for the implementation of global optimization schemes on finite domains instead of the optimization schemes mentioned previously. Commonly used global optimization methods include basin hopping [48], brute force, differential evolution [49], dual annealing [50], SHGO [51], and the DIRECT algorithm [52]. In the context of parameter estimation and especially when using global optimization methods, the utilization of parameter bounds and constraints is crucial to mitigate the computational complexity of navigating these spaces of parameters. Parameter bounds and constraints can be used within our framework along with the distinctive feature of estimating a subset of parameters as constant. These features serve as ways to help tackle the non-convexity and computational efficiency of these data-driven optimization problems as they reduce the space of possible parameters when using global or local optimization schemes.

2.5 Sparse Regression for continuously varying parameters

Sparse regression and dictionary methods have been successfully used across several domains to accurately fit parametric curves to signals. Sparse signal reconstruction on fixed and adaptive supervised dictionary learning for transient stability assessment is provided by Dabou et al [53]. Devito et al. provide a dictionary optimization method for reconstruction of compressed ECG signals [54]. A framework for signal processing using dictionaries, atoms, and deep learning is given by Zhang and Van Der Baan [55]. Given the advances in using dictionaries for sparse regression, we use it for the purposes of identifying the structures of continuously changing parameters in parameter-varying systems.

Suppose that we wish to capture a continuously changing parameter $p(t) : \mathbb{R} \mapsto \mathbb{R}$ as opposed to a piecewise-constant parameter that changes a discrete number of times $p = \{p_1, ..., p_n\}$ where n is the number of segmented data intervals. We can estimate $p(t)$ using our framework as constructed with some specifications on switch detection and an additional step of sparse regression to identify the continuously changing parameters. To do this, we segment the data into $\frac{N}{6}$ intervals where N is the number of data points in the input signal states. This ensures that the parameters of a given system are not over-sampled or under-sampled. Once our parameters have been sampled, we use sparse regression to fit a function to each parameter where each function is a linear combination of a dictionary of functions. Let us denote this dictionary as $\mathcal{D}_\theta(t)$, where each column of this matrix is a parameterized function denoted as $f_i(\theta_i, t)$, where $i \in \{1, ..., r\}$, and r is the number of dictionary functions, and each θ_i is a set of parameters for f_i. The linear combinations of these dictionary functions shown below approximate each parameter, $p(t)$.

$$\mathcal{D}_\theta(t) = \begin{bmatrix} | & & | \\ f_1(\theta_1, t) & \cdots & f_r(\theta_r, t) \\ | & & | \end{bmatrix} \quad . \tag{13}$$

A given parameter $p(t)$ will be a weighted sum of the dictionary functions as shown below

$$p(t) = \sum_i^r w_i f_i(\theta_i, t) \tag{14}$$

where $w_i \in \mathbb{R}$ are the weights. We can then formalize finding $\mathcal{D}_\theta(t)$ such that $\mathcal{D}_\theta(t)w = p(t)$ with the optimization problem

$$\min_{\theta, w} \|p(t) - \mathcal{D}_\theta(t))w\|_2^2 \quad . \tag{15}$$

However, in practice, we cannot simply use the above formulation and expect to get good results. We cannot assume a well chosen dictionary, convexity of the objective function, uniqueness of a solution, or reasonable dimensionality. While these outlined challaenged are largely unresolved, we can use sparse regression to promote solutions $\mathcal{D}_\theta(t)w$ that approximate $p(t)$ with minimal terms. In an applied computational setting, let us denote p as a finite vector approximation of the continuous time-varying parameter $p(t)$. Then in order solve the sparse regression problem, we must instantiate a dictionary of finitely vectorized functions, denoted as $\mathcal{D}_\theta \approx \mathcal{D}_\theta(t)$ that when linearly combined can approximate p. The formulation of the problem is then

$$\min_{\theta, w} \|p - \mathcal{D}_\theta w\|_2 + \lambda \|w\|_0 \tag{16}$$

where $\lambda \in \mathbb{R}$ is the sparse regularization parameter, and $w \in \mathbb{R}^r$ is the vector of weights. The objective function seen in equation (16) allows us to achieve an approximation of the parameter $p \approx D_\theta w$ with only a sparse subset of our dictionary, granted to us by the $\lambda \|w\|_0$ term. Note that that 1-norm for the term $\lambda \|w\|_1$ may grant better results when using gradient-based optimization methods and in other cases.

Algorithm Outline for Dynamical Systems with Sparse Regression

1. **Initialize: model, parameter guesses, and** $X_{data} \in \mathbb{R}^{m\times N}$

$$\frac{dX_{model}}{dt} = f(X\,,\, p(t)\,,\, p_{static})$$

2. **Segment data into** $n := N/6$ **intervals using switch detection**

$$\begin{array}{cccccccccccc} & & [t_0, t_1] & & (t_1, t_2] & & \cdots & & (t_{i-1}, t_i] & & & (t_{n-1}, t_n] \\ X_{data} = & [& X_{data_1} & | & X_{data_2} & | & \ldots & | & X_{data_i} & | & \ldots & X_{data_n} \;] \\ p = & [& p_1 & | & p_2 & | & \ldots & | & p_i & | & \ldots & p_n \;] \end{array}$$

3. **Minimize objective function to obtain parameters for** $[t_0, t_1)$

$\longrightarrow$ **Numerically integrate:** $X_{model_1} = \displaystyle\int_{t_0}^{t_1} \frac{dX_{model_1}}{dt} dt$

$\longrightarrow$ **Optimize:** $\{p_1\,, p_{static}\} = \textbf{argmin} \; \left\| X_{data_1} - X_{model_1} \right\|_2$

4. **Iterate for the** $i = 2,...,n$ **segmented data intervals assuming** p_{static} **is known**

↻ **Optimize:** $\{p_i\} = \textbf{argmin} \; \left\| X_{data_i} - X_{model_i} \right\|_2$

5. **Sparse regression on** p

$\longrightarrow$ **Optimize:** $\{\theta\,, w\} = \textbf{argmin} \; \left\| p - \mathscr{D}_\theta w \right\|_2^2 + \lambda \|w\|_0$

Figure 2: Outline of algorithm with sparse regression for the identification of continuously varying parameters.

3 Results and performance

We first analyze our framework's performance on a nonlinear, parameter-varying system which has served as inspiration for the development of our algorithm. The model that for the Parameter-Varying Toggle Switch (PVTS) is given below and has been analyzed in [56].

$$\begin{pmatrix} \dot{x} \\ \\ \dot{y} \end{pmatrix} = \begin{pmatrix} \frac{\alpha_1(t)}{1+(y/k_y)^{n_y}} - \delta_1(t)x \\ \\ \frac{\alpha_2(t)}{1+(x/k_x)^{n_x}} - \delta_2(t)y \end{pmatrix} \tag{17}$$

$$\alpha_1 = \alpha_2 = \begin{cases} 1.0 & \text{if } t \in [0, 12] \\ 8.0 & \text{if } t \in (12, 24] \end{cases} \tag{18}$$

$$\delta_1 = \delta_2 = \begin{cases} 0.3 & \text{if } t \in [0, 12] \\ 0.6 & \text{if } t \in (12, 24] \end{cases} \tag{19}$$

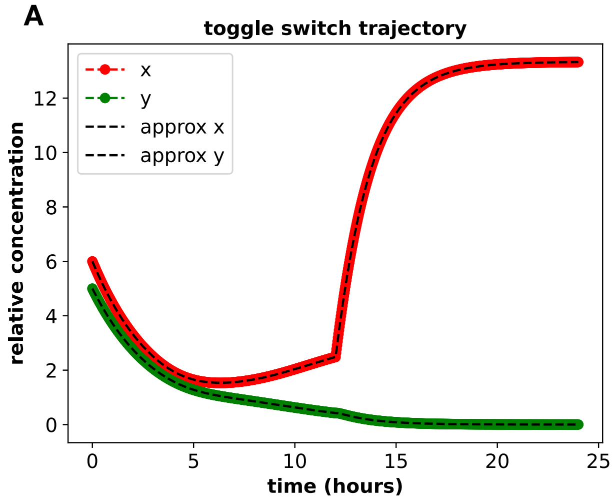
$$k_x = k_y = 1.0 \tag{20}$$

$$n_x = n_y = 3.35 \tag{21}$$

where x and y represent the concentrations of repressor one and repressor two respectively; $\alpha_1(t)$ and $\alpha_2(t)$ denote the effective rates of synthesis of repressor one and two. $\delta_1(t)$ and $\delta_2(t)$ are the self decay rates of repressor one, repressor two; n_x and n_y represent the respective cooperativities of repression for promoter one and promoter 2; the Michaelis constants k_x and k_y are respective binding affinities.

We simulated $N = 1000$ data points from this model using LSODA as the ODE solver. The algorithm sub-steps we used were: `binseg` for switch detection with auto-regressive cost, $\sigma = 10^{-5}$, $s_g = 5$, Differential Evolution for a global parameter search where each parameter is in $[0, 10.00]$ over a given time interval, Forward Euler for numerical integration of the model during the optimization step. Our switch detection method achieved accuracy in terms of $H_s = 0$ and $E_{N_s} = 0$. When assuming $n_{x,y}$ and $k_{x,y}$ to be static parameters as described in Section (2.4), we have $E_p = 0.08$ and $E_t = 0.0041$ for the given PVTS as opposed to leaving this assumption out of our algorithm for a worse performance of $E_p = 0.2$ and $E_t = 0.0041$. Thus, instantiating a subset of parameters as constant granted us more accurate parameter estimates than if we had assumed all parameters to be varying.

A



B

	TRUE	*Approx*	TRUE	*Approx*
α_1	1.00	1.00	8.00	7.96
α_2	1.00	1.01	8.00	7.37
δ_1	0.30	0.30	0.60	0.60
δ_2	0.30	0.30	0.60	0.59
n_x	1.00	0.99	1.00	0.99
n_y	1.00	1.00	1.00	1.00
k_x	3.35	3.49	3.35	3.49
k_y	3.35	3.49	3.35	3.49
	$t \in [0,12)$		$t \in (12,24]$	

Figure 3: (A) Solution to parameter varying toggle switch and reconstruction of solution with estimated parameters. (B) Estimated parameters for parameter varying toggle switch.

We see that we are able to accurately estimate the static and bi-phasic parameters of the PVTS as desired. Achieving this in our framework was non-trivial in that the parameter space contains several local optima over each time interval. To get obtain the level of accuracy we have shown, we employed the global optimization scheme Differential Evolution to minimize E_t, and we made the *a priori* instantiation of the n_{xy} and k_{xy} parameters to be estimated as constant functions. Using Nelder-Mead or Powell's methods as opposed to the Differential Evolution gave us worse errors of $E_t = 0.0049$ and $E_p = 0.79$.

3.1 PDEs and parametric curves

To illustrate how our framework can handle PDE's, we set up the pertinent algorithmic sub-steps and we ran the workflow on the heat equation (5) where the diffusion coefficient is the parameter varying function $D(t) = p_1(t)t^{p_2(t)}$ as shown in Equation (22) where $p_1(t)$ and $p_2(t)$ are piece-wise functions in time.

$$\frac{\partial}{\partial t}u(t,x) = p_1(t)t^{p_2(t)}\frac{\partial^2}{\partial x^2}u(t,x) \tag{22}$$

$$\text{where} \quad p_1(t) = \begin{cases} 0.1 & \text{if} \quad t \in [0, 0.25] \\ 2.1 & \text{if} \quad t \in (0.25, 0.5] \\ 4.1 & \text{if} \quad t \in (0.5, 0.75] \\ 6.1 & \text{if} \quad t \in (0.75, 1] \end{cases} \tag{23}$$

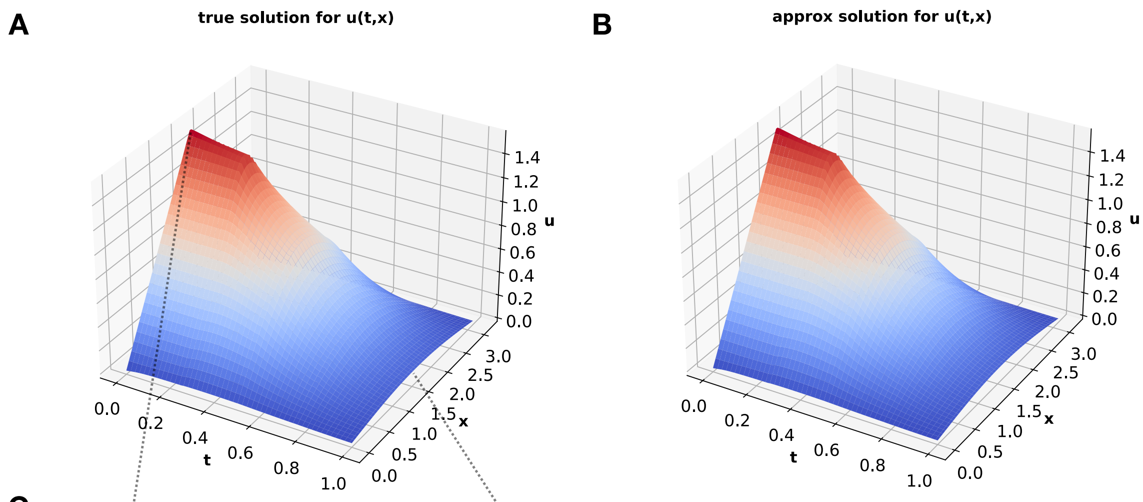
$$p_2(t) = \begin{cases} 0.5 & \text{if} \quad t \in [0, 0.25] \\ 0.4 & \text{if} \quad t \in (0.25, 0.5] \\ 0.3 & \text{if} \quad t \in (0.5, 0.75] \\ 0.2 & \text{if} \quad t \in (0.75, 1] \end{cases} \tag{24}$$

$$\text{with initial condition} \quad u(t=0,x) = \begin{cases} x & \text{if} \quad x \in [0, \frac{\pi}{2}] \\ \pi - x & \text{if} \quad x \in (\frac{\pi}{2}, \pi] \end{cases} \tag{25}$$

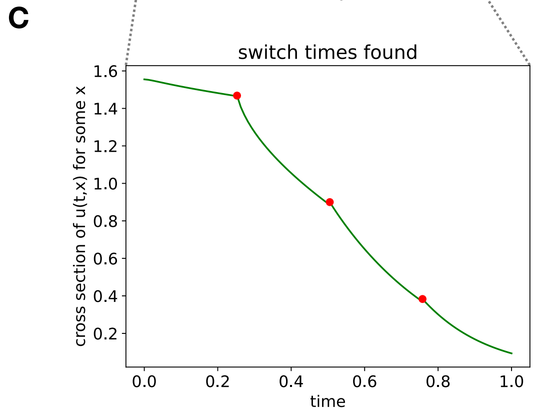
$$\text{and with boundary condition} \quad u(t, x=0) = u(t, x=\pi) = 0 \quad . \tag{26}$$

A

B



C



D

$t \in$	[0,0.25]	(0.25,0.5]	(0.5,0.75]	(0.75,1]
p_1	0.1	2.1	4.1	6.1
$\hat{p}_1$	0.1	2.4	4.0	5.9
p_2	0.5	0.4	0.3	0.2
$\hat{p}_2$	0.5	0.5	0.3	0.2

Figure 4: (A) True solution to heat equation PDE. (B) Reconstruction of solution to heat equation PDE with estimated parameters . (C) Switches detected from cross-sectional slice of PDE data. (D) Estimated parameters for heat equation PDE from data.

The theoretical physics underlying this example problem provides valuable insight into the behavior of complex materials under dynamic thermal conditions. In this case, the 1-D heat equation models the temperature distribution along a thin rod with an initial heat profile $u(0, x)$ as defined in Equation (25). As time progresses, the parameter representing the rod's heat diffusivity, $D(t)$, evolves dynamically. This behavior reflects the rod's composition—a non-homogeneous composite material consisting of three distinct compounds, each with unique thermal properties and phase-change characteristics.

As the rod cools, the heat diffusivity of one compound undergoes a significant change at its critical temperature, corresponding to the first parameter switch. Similar transitions occur for the other two compounds, resulting in the discrete parameter switches observed in the measured data. Assuming that previous studies have shown that the heat diffusivity of such composites changes exponentially with cooling, this behavior can be represented by the parameter structure $D(t) = p_1(t)t^{p_2(t)}$. The non-homogeneous nature of the composite material justifies the discrete switching observed in the heat diffusivity.

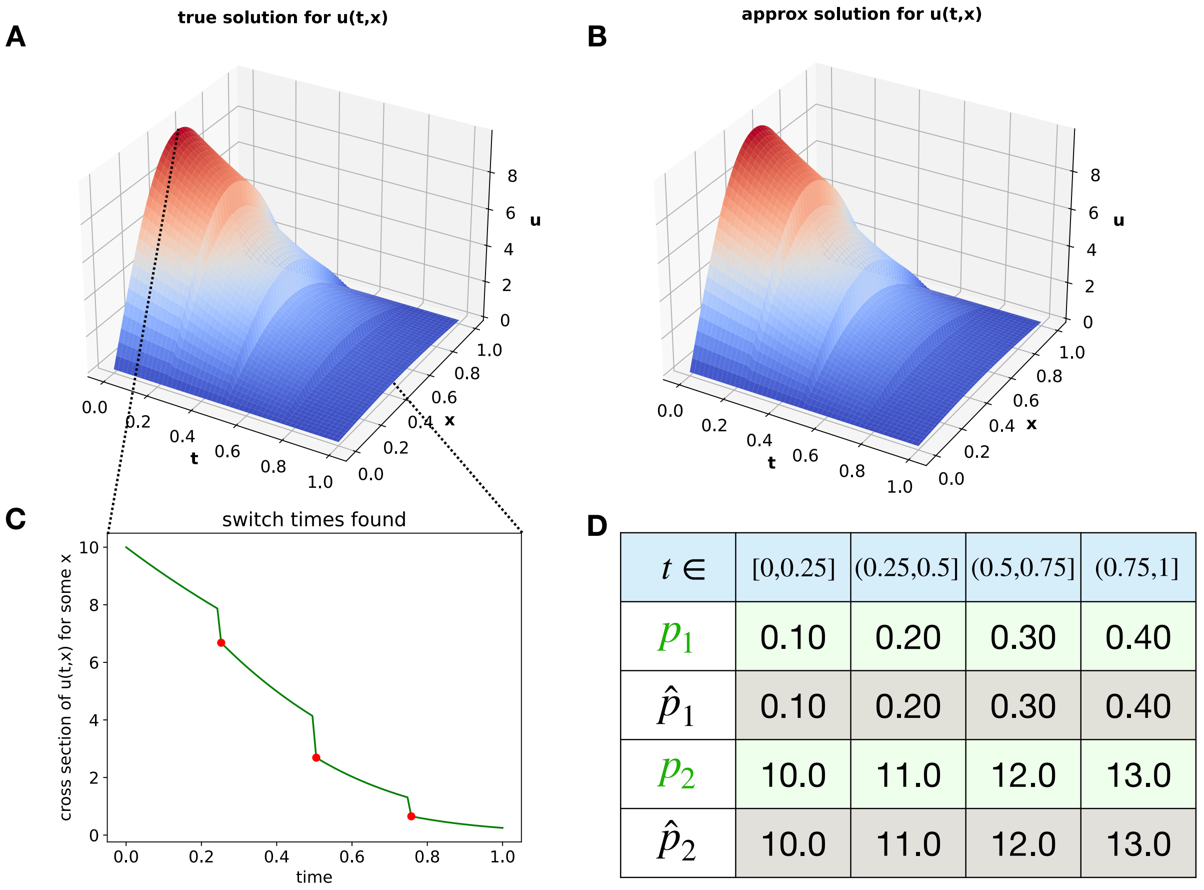
We used the method of lines as explicitly written in Equation (22) to represent the PDE as a system of ODEs as shown in Equation (10) with 100 uniformly spaced grid points in x $(M = 100)$ and 100 time points over the domain $(t, x) = [0, 1] \times [0, \pi]$. The ODE solver in the optimization sub-step was Forward Euler, we used `binseg` for switch detection with auto-regressive cost, $\sigma = 10^{-3}$, $s_g = 10$, and Nelder-Mead for locally optimal parameters with no pre-set bounds. We see that we have achieved $H_s = 0$, $E_{N_s} = 0$, $E_p = 0.19$, and $E_t = 0.001$ via the algorithm specifications. Thus, we have shown that that an algorithm specified within our framework can accurately estimate the varying parameters of a PDE.

To illustrate our algorithm's capabilities in working with parametric curves, we tested it on the function shown in Equation (27) where $p_1(t)$ and $p_2(t)$ are piece-wise functions in time.

$$u(t,x) = p_2(t) sin(\pi x) e^{-p_1(t)\pi^2 t} \tag{27}$$

$$\text{where} \quad p_1(t) = \begin{cases} 0.1 & \text{if} \quad t \in [0, 0.25] \\ 0.2 & \text{if} \quad t \in (0.25, 0.5] \\ 0.3 & \text{if} \quad t \in (0.5, 0.75] \\ 0.4 & \text{if} \quad t \in (0.75, 1] \end{cases} \tag{28}$$

$$p_2(t) = \begin{cases} 10.0 & \text{if} \quad t \in [0, 0.25] \\ 11.0 & \text{if} \quad t \in (0.25, 0.5] \\ 12.0 & \text{if} \quad t \in (0.5, 0.75] \\ 13.0 & \text{if} \quad t \in (0.75, 1] \end{cases} \tag{29}$$



$t \in$	[0,0.25]	(0.25,0.5]	(0.5,0.75]	(0.75,1]
p_1	0.10	0.20	0.30	0.40
$\hat{p}_1$	0.10	0.20	0.30	0.40
p_2	10.0	11.0	12.0	13.0
$\hat{p}_2$	10.0	11.0	12.0	13.0

Figure 5: (A) 3D parametric curve with parameter switches. (B) Reconstruction of 3D parametric curve with estimated parameters. (C) Switches detected from cross-sectional slice of 3D parametric curve data. (D) Estimated parameters for 3D parametric curve.

We used our algorithm described in Figure (2) on data simulated from Equation (27) 100 uniformly spaced grid points in x and 100 discretized time points over the domain $(t, x) = [0, 1] \times [0, 1]$. The methods used here are the same as those used in the heat equation example, except there was no numerical integration done because we have a parametric curve model and not a differential equation.

We see that for the parametric curve model, we have achieved $H_s = 0$, $E_{N_s} = 0$, $E_p = 2.5 \times 10^{-5}$, $E_t = 0.0002$ which are extraordinarily accurate results in comparison to that of the PDE example. These relatively small errors are the result of there not being numerical integration steps in the handling of these models, so numerical integration error cannot be propagated through algorithmic sub-steps as would happen with differential equation models.

3.2 Robustness of Switch detection and parameter estimation given noisy data

Quantifying the noise-robustness of our framework is needed given the ever-increasing need to process and understand noisy data for system identification and control [57, 58, 59]. To quantify the robustness of our algorithm, we added white noise (zero mean and finite variance) to synthetic data that was generated from a known ground-truth model. We then ran this noisy data through our algorithm and identified relationships between the noisiness of the data and 3 types of error described in Table (1). The ground truth model we used is a biological system given in Equation (30) where m is mRNA concentration, p is protein concentration, $\alpha_{m,p}$ are respective rates of synthesis, and $\delta_{m,p}$ are the respective degradation rates. This is a standard protein synthesis model in which mRNA is transcribed into a given protein. Note that in this model, we have that the protein production is dependent on the transcription of mRNA, but not vice versa. Hence, the model is in agreement with the central dogma of biology.

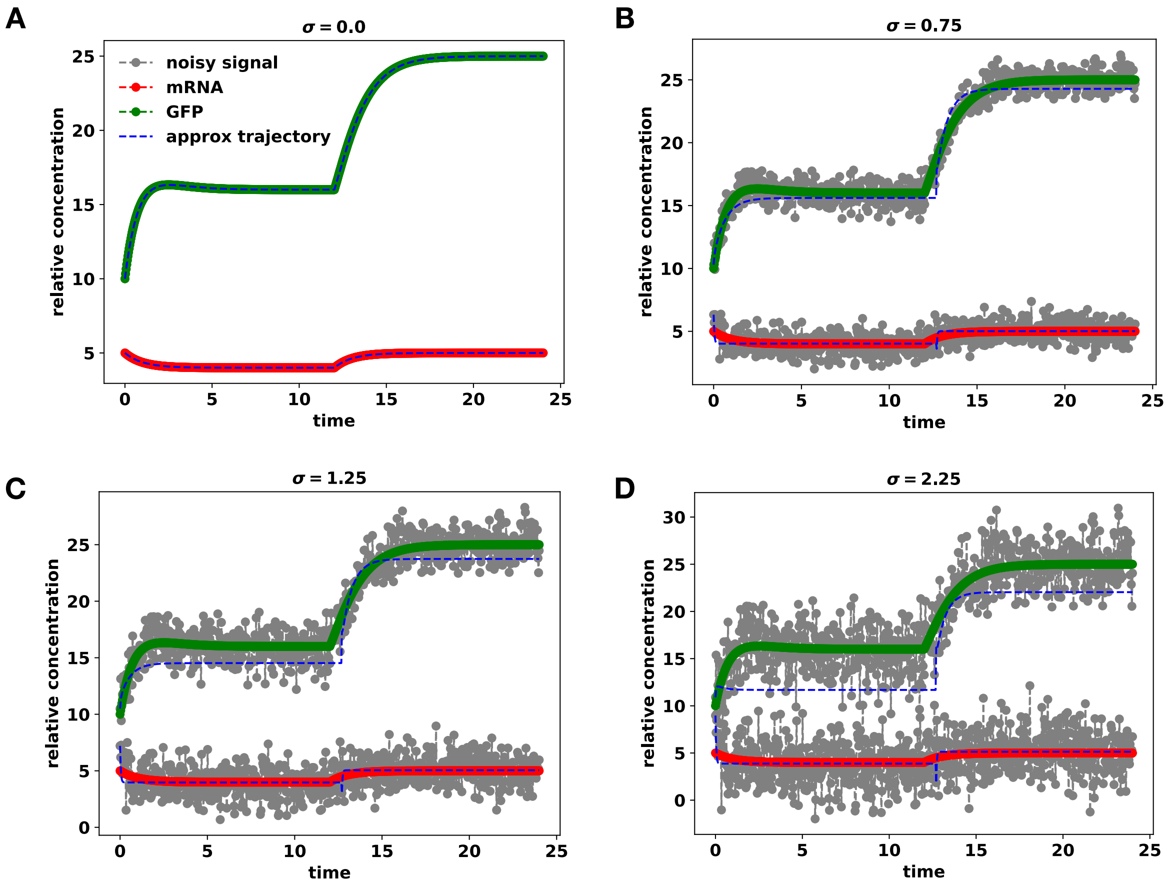


Figure 6: (A) Synthetic data and reconstruction from estimated parameters. (B) Synthetic data with white noise ($\sigma = 0.75$) and reconstruction from estimated parameters. (C) Synthetic data with white noise ($\sigma = 1.25$) and reconstruction from estimated parameters (D) Synthetic data with white noise ($\sigma = 2.25$) and reconstruction from estimated parameters.

A motivating factor for this work was to validate novel genetic circuit models with time-varying kinetic rates such as the one below. Based on the equations provided, it is clear that the $\alpha_{m,p}$ rates of synthesis are the varying parameters of the model, and that the respective degradation rates are modeled as unchanging. In an experimental setting, the data that would be collected would be subjected to some measurement noise from the instruments that are used to quantify time-series gene expression levels.

$$\dot{m} = \alpha_m(t) - \delta_m(t)m \tag{30}$$
$$\dot{p} = \alpha_p(t)m - \delta_p(t)p \tag{31}$$
$$\alpha_m(t) = \alpha_p(t) = \begin{cases} 4.0 & \text{if} \quad t \in [0, 12] \\ 5.0 & \text{if} \quad t \in (12, 24] \end{cases} \tag{32}$$
$$\delta_m(t) = \delta_p(t) = 1 \quad \forall t \in [0, 24] \quad . \tag{33}$$

Let the observables of our synthetic data be denoted as $x := [m \quad p]^T$ and let the white noise we add to it be denoted by v_w. Each element of the white noise vector v_w is sampled from a normal distribution with zero mean and finite standard deviation, σ. Our algorithm was used to estimate the time-varying parameters of the given model (30) using $X_{data} := x + v_w$. A subset of the these runs are found in Figure (6), where we manually set the number of switches to be detected in the data to be $N_s = 1$ so that we could measure noise effects in this idealized best case scenario. In the worst case scenario where the number of parameter switches, N_s, is unknown, switch detection algorithms often mistake "jumps" in the data from the noise as switches which leads to worse performance than what we show here. This worst case scenario often gives relatively accurate data reconstruction, but unreliable parameters and untrustworthy detection of switches.

The ODE solver we used for the numerical integration in the optimization sub-step was Forward Euler and the algorithm sub-steps we used were: `binseg` for switch detection with auto-regressive cost, $\sigma = N_s = 1$, $s_g = 10$, Nelder-Mead for locally optimal parameters with no pre-set bounds.

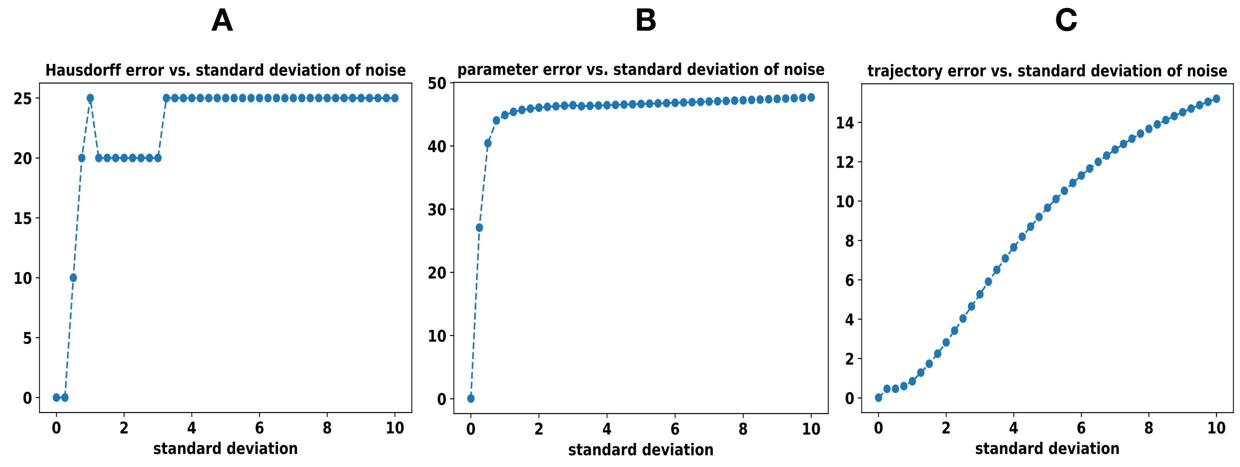


Figure 7: (A) Hausdorff error vs. standard deviation of noise. (B) Parameter error vs. standard deviation of noise. (C) Trajectory error vs. standard deviation of noise.

From both Figure (6) and Figure (7), it is clear that our method for parameter estimation shows increases in all error metrics with with increases to the noise in data. There is a cascading effect of error propagation due to noise (v_w) : `binseg` inaccurately detects switches which leads to inaccurate data segmentation ($\uparrow D_s$ and $\uparrow E_{N_s}$) which causes inaccurate parameter fitting (E_p), and thusly, parameterized model trajectories inaccurately approximate data (E_t). This cascading effect is summarized as

$$\uparrow \|v_w\| \implies (\uparrow H_s \text{ and } \uparrow E_{N_s}) \implies \uparrow E_p \implies \uparrow E_t \quad . \tag{34}$$

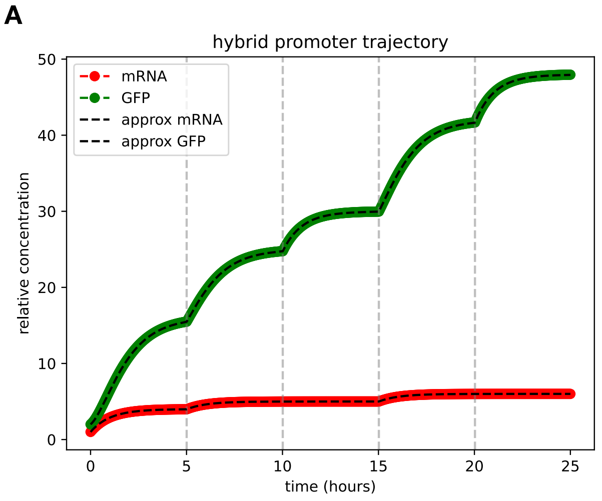
This cascading propagation of error is one of the drawbacks of algorithms in our framework. Methodological sub-steps specializing in the robust handling of noisy data can be added on at various points within our framework for improvements in this weak-point, but this is outside the scope of this paper and may be further expanded on in future work.

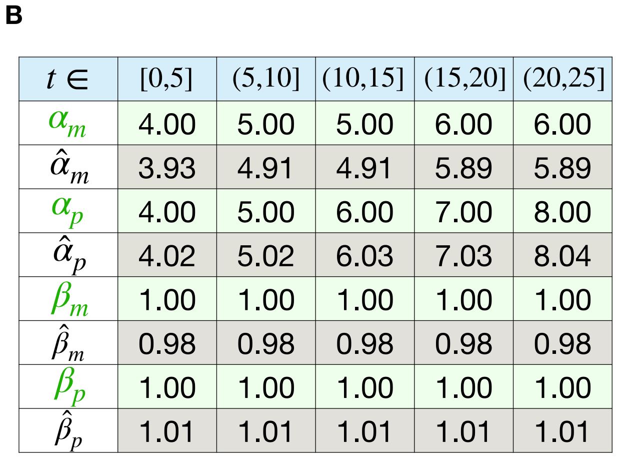
3.3 Switching parameters at non-uniform intervals

The special handling of our adapted switch detection framework to detect discretely switching parameters in dynamical systems is heavily motivated by cases in which parameters across the states of a dynamical systems are switching independently from each other. Suppose now that we have the same system as in Equation (30) but with parameters that switch non-uniformly on different time intervals as depicted in Equation (35).

$$\dot{m} = \alpha_m(t) - \delta_m(t)m \tag{35}$$
$$\dot{p} = \alpha_p(t)m - \delta_p(t)p \tag{36}$$
$$\text{where} \quad \alpha_m(t) = \begin{cases} 4.0 & \text{if} \quad t \in [0,5] \\ 5.0 & \text{if} \quad t \in (5,15] \\ 6.0 & \text{if} \quad t \in (15,24] \end{cases} \tag{37}$$
$$\alpha_p(t) = \begin{cases} 4.0 & \text{if} \quad t \in [0,5] \\ 5.0 & \text{if} \quad t \in (5,10] \\ 6.0 & \text{if} \quad t \in (10,15] \\ 7.0 & \text{if} \quad t \in (15,20] \\ 8.0 & \text{if} \quad t \in (20,25] \end{cases} \tag{38}$$
$$d_m(t) = d_p(t) = \begin{cases} 1 & \text{if} \quad t \in [0,25] \end{cases}. \tag{39}$$

We test our algorithm on System (35) to show that our adaptation of `binseg` is able to handle these scenarios as shown in Figure (8). The ODE solver we used for the numerical integration in the optimization sub-step was Forward Euler and the algorithm sub-steps we used were: `binseg` for switch detection with auto-regressive cost, $\sigma = N_s = 1$, $s_g = 10$, Nelder-Mead for locally optimal parameters with no pre-set bounds. With this algorithmic set-up, we achieved $H_s = 0$, $E_{N_s} = 0$, $E_t = 0.001$ and $E_p = 0.05$.




B

$t \in$	[0,5]	(5,10]	(10,15]	(15,20]	(20,25]
α_m	4.00	5.00	5.00	6.00	6.00
$\hat{\alpha}_m$	3.93	4.91	4.91	5.89	5.89
α_p	4.00	5.00	6.00	7.00	8.00
$\hat{\alpha}_p$	4.02	5.02	6.03	7.03	8.04
β_m	1.00	1.00	1.00	1.00	1.00
$\hat{\beta}_m$	0.98	0.98	0.98	0.98	0.98
β_p	1.00	1.00	1.00	1.00	1.00
$\hat{\beta}_p$	1.01	1.01	1.01	1.01	1.01

Figure 8: (A) Trajectory of hybrid promoter model with different parameters switching at different time intervals. (B) Approximated parameters of hybrid promoter model in which parameters switch at different time intervals.

3.4 Sparse dictionary regression for continuously varying parameters

To test our method outlined in Section (2.5), we estimate the time-varying parameter $\alpha(t)$ in the PDE model for advection-diffusion found in Equation (40) using an algorithm specified within our framework adjoined with sparse regression. The main differences between this example and that of Section (3.1) are that we now have an advection term in the PDE denoted by the first order spatial derivative and that we are not assuming knowledge of the structure of our varying parameter, $\alpha(t)$. We proceed in the estimation of $\alpha(t)$ as a continuously varying parameter for two reasons: due to the horrendous accuracy obtained if estimating all parameters as constant, and the lack of coherent discrete switches detected in the data. Intuitively, it is clear from the data found in Figure (9) that the direction and velocity of advection is changing. We assumed in our test run that the diffusion coefficient D was to be estimated as a constant parameter.

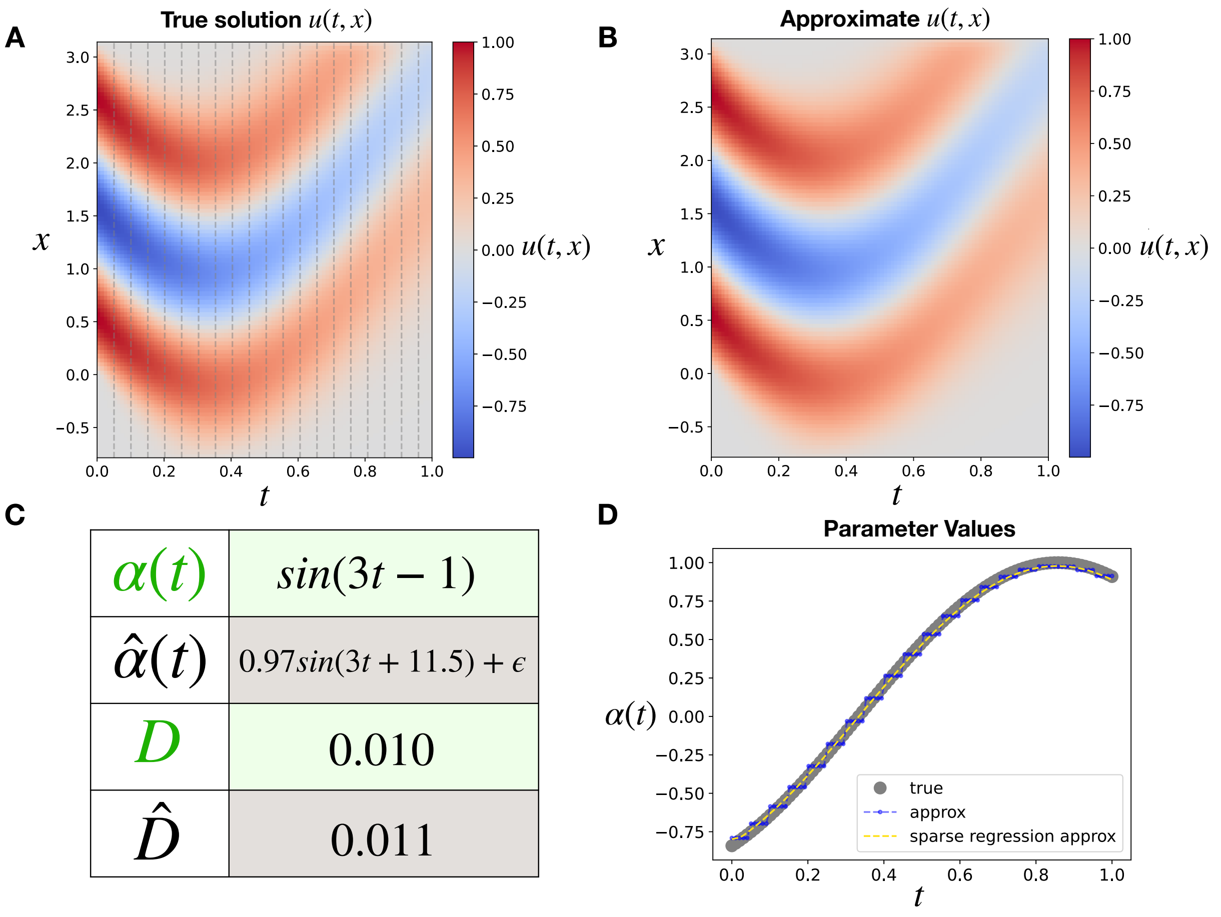
$$\frac{\partial}{\partial t}u(t,x) = \alpha(t)\frac{\partial}{\partial x}u(t,x) + D\frac{\partial^2}{\partial x^2}u(t,x) \tag{40}$$

$$\text{where} \qquad \alpha(t) = sin(3t-1) \quad \text{and} \quad D = 0.01 \tag{41}$$

$$\text{with initial condition} \quad u(0,x) = \begin{cases} sin(3x) & \text{if} \quad x \in [0,\pi] \\ 0 & \text{if} \quad \text{else} \end{cases} \tag{42}$$

We used a dictionary of three functions found below

$$\mathcal{D}_\theta(t) = \begin{bmatrix} | & | & | \\ e^{\theta_1 t^{\theta_2}} & sin(\theta_3 t + \theta_4) & t^{\theta_5} \\ | & | & | \end{bmatrix} \quad . \tag{43}$$



$\alpha(t)$	$sin(3t-1)$
$\hat{\alpha}(t)$	$0.97sin(3t+11.5)+\epsilon$
D	0.010
$\hat{D}$	0.011

Figure 9: (A) Synthetic data from advection-diffusion equation containing a continuously varying parameter. The vertical gray lines indicate the segmentation of the data for the sampling of the continuously varying parameter. (B) Reconstruction of heat equation data using estimated model parameters from algorithm with sparse regression. (C) Table of parametrized true and estimated parameters from the set of dictionary functions. (D) True and approximate parameters from our algorithm with and without sparse regression.

We used the method of lines to set up this PDE and we used LSODA to simulate the data with 100 uniformly spaced grid points in x $(M = 100)$ and 100 time points over the domain $(t, x) = [0, 1] \times [-\pi/4, \pi]$. The ODE solver we used for the numerical integration in the optimization sub-step was Forward Euler and the algorithm sub-steps we used were: `binseg` for data segmentation with auto-regressive cost, the number of assumed switches as $\sigma = N_s = N/6$, the switch gap as $s_g = 1$, Nelder-Mead for locally optimal parameters with pre-set bounds such that $|\alpha(t)| < 10$ and $|D| < 10$, and sparse regression with regularization term as $\lambda\|w\|_1$ where $\lambda = 0.01$. With this algorithmic setup, we achieved $E_t = 0.001$ and $E_p = 0.001$.

Without our implementation of sparse regression, we achieved a parameter error of $E_p = 0.16$. With sparse regression we achieved a parameter error of $E_p = 0.05$, thus showcasing an improvement in parameter estimation accuracy through the utilization of sparse regression. Further, sparse regression has allowed us to reconstruct the parameterization of $\alpha(t)$ as $\hat{\alpha}(t) = 0.97 sin(3t + 11.5) + \epsilon$ where ϵ contains the minuscule terms from the other dictionary functions, namely, $\epsilon = (10^{-13})e^{-2.55t^{0.6}} + 0.01t^{0.96}$.

4 Conclusion

In conclusion, we present a modular framework for estimating varying parameters in dynamical systems, focusing initially on the sub-class of problems where parameters switch discretely and are modeled as piecewise constant functions. By combining the strengths of switch detection, numerical integration, and optimization into a cohesive structure, our framework offers accuracy and flexibility to accommodate user-specific requirements. Our approach has been validated across diverse examples, including biological systems and physical models, demonstrating its adaptability to scenarios with fixed and switching parameters, non-uniform switching, and time-varying dynamics. Moreover, we characterized the framework's robustness to measurement noise and extended its capabilities to continuously varying parameters using dictionary-based sparse regression with trigonometric and polynomial functions. This work establishes a foundation for addressing a wide range of parameter-varying system problems and provides a versatile tool set for advancing research in dynamical system modeling and control.

Acknowledgments

For insightful discussions related to this work, the authors would like to thank Charles Johnson, Yanran Wang, Taishi Kotsuka, Harnoor Lali, and João Hespanha. This work was funded in part by an NSF CAREER Award 2240176, the Army Young Investigator Program Award W911NF2010165 and the Institute of Collaborative Biotechnologies/Army Research Office grants W911NF19D0001, W911NF22F0005, W911NF190026, W911NF1920026 and W911NF2320006. This work was also supported in part by a subcontract awarded by the Pacific Northwest National Laboratory for the Secure Biosystems Design Science Focus Area "Persistence Control of Engineered Functions in Complex Soil Microbiomes" sponsored by the U.S. Department of Energy Office of Biological and Environmental Research.

Additional information

We report no conflicts of interest.

The code used to produce all data and figures in this paper can be found at the following Github repository. We also provide other supplementary examples not covered in this paper.

References

[1] E.P. Ryan and A. Ichikawa. Parameter estimation and control for distributed systems. *IFAC Proceedings Volumes*, 12(7):149–155, 1979. IFAC Symposium on computer Aided Design of Control Systems, Zurich, Switzerland, 29-31 August.

[2] John A. Burns, Eugene M. Cliff, and Kasie Farlow. Parameter estimation and model discrepancy in control systems with delays. *IFAC Proceedings Volumes*, 47(3):11679–11684, 2014. 19th IFAC World Congress.

[3] Eshan D Mitra and William S Hlavacek. Parameter estimation and uncertainty quantification for systems biology models. *Current opinion in systems biology*, Dec 2019.

[4] David R. Penas, Patricia González, Jose A. Egea, Ramón Doallo, and Julio R. Banga. Parameter estimation in large-scale systems biology models: A parallel and self-adaptive cooperative strategy. *BMC Bioinformatics*, 18(1), Jan 2017.

[5] Choujun Zhan and Lam F Yeung. Parameter estimation in systems biology models using spline approximation. *BMC Systems Biology*, 5(1):14, 2011.

[6] Gloria I. Valderrama-Bahamóndez and Holger Fröhlich. Mcmc techniques for parameter estimation of ode based models in systems biology. *Frontiers in Applied Mathematics and Statistics*, 5, Nov 2019.

[7] Stefano Giampiccolo, Federico Reali, Anna Fochesato, Giovanni Iacca, and Luca Marchetti. Robust parameter estimation and identifiability analysis with hybrid neural ordinary differential equations in computational biology. *Robust parameter estimation and identifiability analysis with hybrid neural ordinary differential equations in computational biology*, Jun 2024.

[8] Maksat Ashyraliyev, Yves Fomekong-Nanfack, Jaap A. Kaandorp, and Joke G. Blom. Systems biology: parameter estimation for biochemical models. *The FEBS Journal*, 276(4):886–902, 2009.

[9] M. Douspis, J. G. Bartlett, A. Blanchard, and M. Le Dour. Concerning parameter estimation using the cosmic microwave background. *Astronomy and Astrophysics*, 368(1):1–14, Mar 2001.

[10] Coryn A.L. Bailer-Jones. *Astronomical Object Classification and Parameter Estimation with the Gaia Galactic Survey Satellite*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[11] Zhu Wang, Haoran An, and Xionglin Luo. Switch detection and robust parameter estimation for slowly switched hammerstein systems. *Nonlinear Analysis: Hybrid Systems*, 32:202–213, 2019.

[12] Wenju Zheng, Yiming Wan, Fan Yang, Chao Shang, Hao Ye, Ming Jiang, and Jia Wang. Identification of switched dynamic system for electric multiple unit train modeling. *Control Engineering Practice*, 143:105815, 2024.

[13] Miao Yu, Federico Bianchi, and Luigi Piroddi. A randomized method for the identification of switched narx systems. *Nonlinear Analysis: Hybrid Systems*, 49:101364, 2023.

[14] Federico Bianchi, Valentina Breschi, Dario Piga, and Luigi Piroddi. Model structure selection for switched narx system identification: A randomized approach. *Automatica*, 125:109415, 2021.

[15] Haoyu Li, Ke Zhang, and Minghu Tan. A novel system identification algorithm for nonlinear markov jump system. *Information Sciences*, 616:348–366, 2022.

[16] Charles A Johnson and Enoch Yeung. Nonlinear data-driven control via koopman models: the interplay of stabilization and subspace closure error. *arXiv e-prints*, 2024.

[17] Yoshihiko Susuki, Kohei Eto, Naoto Hiramatsu, and Atsushi Ishigame. Control of oscillatory temperature field in a building via damping assignment to nonlinear koopman mode. In *2022 IEEE Conference on Control Technology and Applications (CCTA)*, pages 796–801. IEEE, 2022.

[18] Yoshihiko Susuki, Kohei Eto, Naoto Hiramatsu, and Atsushi Ishigame. Control of in-room temperature field via damping assignment to nonlinear koopman mode. *IEEE Transactions on Control Systems Technology*, 2024.

[19] Johan Kon, Jeroen van de Wijdeven, Dennis Bruijnen, Roland Tóth, Marcel Heertjes, and Tom Oomen. Direct learning for parameter-varying feedforward control: A neural-network approach, 2023.

[20] Ning Xu and Feng Ding. Parameter estimation for a class of time-varying systems with the invariant matrix. *International Journal of Robust and Nonlinear Control*, 33(3):2163–2181, 2023.

[21] Luiz Cláudio Andrade Souza and Reinaldo Martinez Palhares. Parameter estimation on linear time-varying systems. *Journal of the Franklin Institute*, 348(4):777–789, 2011.

[22] Ryan S. Johnson, Stefano Di Cairano, and Ricardo G. Sanfelice. Parameter estimation for hybrid dynamical systems using hybrid gradient descent. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 4648–4653, 2021.

[23] Liming Qian. Bayesian parameter estimation for linear parameter-varying systems with outliers and missing observations. *IEEE Access*, pages 1–1, 2024.

[24] Fatimah Al-Taie, Ibtisam Kamil Hanan, and Fadhel S. Fadhel. Extended robust controller design for parameter varying multi-dimensional systems. *AIP Conference Proceedings*, 3079(1):050005, 04 2024.

[25] Charles Truong, Laurent Oudre, and Nicolas Vayatis. Selective review of offline change point detection methods. *Signal Processing*, 167:107299, 2020.

[26] Lucas de Oliveira Prates. A more efficient algorithm to compute the rand index for change-point problems, 2021.

[27] E.N. Sarmin and L.A. Chudov. On the stability of the numerical integration of systems of ordinary differential equations arising in the use of the straight line method. *USSR Computational Mathematics and Mathematical Physics*, 3(6):1537–1543, 1963.

[28] Ryan Prescott Adams and David J. C. MacKay. Bayesian online changepoint detection, 2007.

[29] P. Fearnhead R. Killick and I. A. Eckley. Optimal detection of changepoints with a linear computational cost. *Journal of the American Statistical Association*, 107(500):1590–1598, 2012.

[30] Leonid I. Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259–268, 1992.

[31] E. S. PAGE. CONTINUOUS INSPECTION SCHEMES. *Biometrika*, 41(1-2):100–115, 06 1954.

[32] B. Jackson, J.D. Scargle, D. Barnes, S. Arabhi, A. Alt, P. Gioumousis, E. Gwin, P. Sangtrakulcharoen, L. Tan, and Tun Tao Tsai. An algorithm for optimal partitioning of data on an interval. *IEEE Signal Processing Letters*, 12(2):105–108, 2005.

[33] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[34] Zaïd Harchaoui and Olivier Cappé. Retrospective mutiple change-point estimation with kernels. *2007 IEEE/SP 14th Workshop on Statistical Signal Processing*, pages 768–772, 2007.

[35] S. Mallat. *A wavelet tour of signal processing*. Science Direct, 1999.

[36] Jean-philippe Vert and Kevin Bleakley. Fast detection of multiple change-points shared by many signals using group lars. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010.

[37] Song Liu, Makoto Yamada, Nigel Collier, and Masashi Sugiyama. Change-point detection in time-series data by relative density-ratio estimation. *Neural Networks*, 43:72–83, 2013.

[38] M. Lavielle and G. Teyssière. Detection of multiple change-points in multivariate time series. *Lithuanian Mathematical Journal*, 46(3):287–306, Jul 2006.

[39] Thomas C. M Lee Richard A Davis and Gabriel A Rodriguez-Yam. Structural break estimation for nonstationary time series models. *Journal of the American Statistical Association*, 101(473):223–239, 2006.

[40] Jushan Bai. Estimating multiple breaks one at a time. *Econometric Theory*, 13(3):315–352, 1997.

[41] Piotr Fryzlewicz. Wild binary segmentation for multiple change-point detection. *The Annals of Statistics*, 42(6):2243 – 2281, 2014.

[42] Abraham. Savitzky and M. J. Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chemistry*, 36(8):1627–1639, Jul 1964.

[43] Michael Schmid, David Rath, and Ulrike Diebold. Why and how savitzky–golay filters should be replaced. *ACS Measurement Science Au*, 2(2):185–196, 2022.

[44] E. T. Whittaker. On a new method of graduation. *Proceedings of the Edinburgh Mathematical Society*, 41:63–75, 1922.

[45] J. A. Nelder and R. Mead. A Simplex Method for Function Minimization. *The Computer Journal*, 7(4):308–313, 01 1965.

[46] M. J. D. Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal*, 7(2):155–162, 01 1964.

[47] Attila Gábor and Julio R. Banga. Robust and efficient parameter estimation in dynamic models of biological systems. *BMC Systems Biology*, 9(1), Oct 2015.

[48] David J. Wales and Jonathan P. Doye. Global optimization by basin-hopping and the lowest energy structures of lennard-jones clusters containing up to 110 atoms. *The Journal of Physical Chemistry A*, 101(28):5111–5116, Jul 1997.

[49] Jouni Lampinen and Rainer Storn. *Differential Evolution*, pages 123–166. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[50] Giovanni Righini. A double annealing algorithm for discrete location/allocation problems. *European Journal of Operational Research*, 86(3):452–468, 1995.

[51] Stefan C. Endres, Carl Sandrock, and Walter W. Focke. A simplicial homology algorithm for lipschitz optimisation. *Journal of Global Optimization*, 72(2):181–217, Mar 2018.

[52] D. R. Jones, C. D. Perttunen, and B. E. Stuckman. Lipschitzian optimization without the lipschitz constant. *Journal of Optimization Theory and Applications*, 79(1):157–181, Oct 1993.

[53] Raoult Teukam Dabou, Innocent Kamwa, Jacques Tagoudjeu, and Francis Chuma Mugombozi. Sparse signal reconstruction on fixed and adaptive supervised dictionary learning for transient stability assessment. *Energies*, 14(23), 2021.

[54] L De Vito, E Picariello, F Picariello, S Rapuano, and I. A Tudosa. Dictionary optimization method for reconstruction of ecg signals after compressed sensing. *Sensors (Basel)*, 2021.

[55] Chao Zhang and Mirko Van Der Baan. Signal processing using dictionaries, atoms, and deep learning: A common analysis-synthesis framework. *Proceedings of the IEEE*, 110(4):454–475, 2022.

[56] Jamiree Harrison and Enoch Yeung. Stability analysis of parameter varying genetic toggle switches using koopman operators. *Mathematics*, 9(23), 2021.

[57] Shahriar Akbar Sakib and Shaowu Pan. Learning noise-robust stable koopman operator for control with physics-informed observables. *arXiv e-prints*, pages arXiv–2408, 2024.

[58] Shahriar Akbar Sakib and Shaowu Pan. Learning noise-robust stable koopman operator for control with hankel dmd. *arXiv preprint arXiv:2408.06607*, 2024.

[59] Subhrajit Sinha, Sai P Nandanoori, and DA Barajas-Solano. Online real-time learning of dynamical systems from noisy streaming data. *Scientific Reports*, 13(1):22564, 2023.