The Journal of Open Source Software

CD-Dynamax: A JAX-based Python package for continuous-discrete probabilistic state space modeling and inference

Matthew Levine [1,2¶] **and Iñigo Urteaga** [3,4¶]

1 Broad Institute of MIT and Harvard, USA **2** Basis Research Institute, USA **3** BCAM – Basque Center for Applied Mathematics, Spain **4** IKERBASQUE — Basque Foundation for Science, Spain ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor:

Submitted: 09 December 2025
Published: unpublished

Summary

Dynamical systems, often modeled as stochastic differential equations (SDEs), are widely used mathematical tools to describe complex phenomena in various scientific fields, including engineering, economics, neuroscience, ecology, and climate science. In most real-world scenarios, observations of these systems are collected, subject to noise, at discrete and irregular time intervals, requiring nuanced modeling approaches.

Continuous-discrete state space models (CD-SSMs) provide a powerful probabilistic framework for modeling such systems (Särkkä & Svensson, 2023). These models describe the latent state evolution continuously over time according to an SDE, while noisy observations are obtained at specific, discrete time instants.

Mathematically, a CD-SSM is described according to:

- A (possibly unknown) stochastic dynamical system, i.e.,

$$dx(t) = f(x(t), t)dt + L(x(t), t)dw(t) \ ,$$

  where:
  - $x \in \mathbb{R}^{d_x}$ and $x(0) \sim P(x_0)$,
  - f is a (possibly time-dependent) drift function,
  - L is a (possibly state and/or time-dependent) diffusion coefficient, and
  - dw is the derivative of a d_x-dimensional Brownian motion with a covariance Q.
- Data is observed at arbitrary times $\{t_k\}_{k=1}^K$ via a measurement process

$$y(t) = h(x(t)) + \eta(t) \ ,$$

  where:
  - $h : \mathbb{R}^{d_x} \to \mathbb{R}^{d_y}$; i.e., h transforms the d_x-dimensional state of the dynamical system $x(t)$ (a realization of the SDE) to a d_y-dimensional observation, and
  - $\eta(t)$ is an independent and identically distributed noise process that corrupts the observations.
- The collection of CD-SSM parameters is denoted with θ, which may include parameters governing the latent dynamics (e.g., parameters of f and L) and/or parameters of the observation model (e.g., parameters of h and, in the Gaussian observation noise case, its covariance matrix R).

This mathematical framework describes ***continuous (dynamics) - discrete (observation)* state space models**. Under this formulation, CD-SSMs enable accurate modeling of dynamical systems where noisy data are collected at irregular intervals and the underlying processes

evolve continuously over time. Note that a CD-SSM may also include inputs (i.e., controls), $u_1, \ldots, u_K$, also occurring at times $\{t_k\}_{k=1}^K$ to steer the latent state dynamics and influence the observations.

When constructing a CD-SSM for a specific application, the modeler must define the functional forms of the latent dynamics and observation models. Namely, there are two key design choices to make:

1. How do the latent states evolve over time? E.g., are the latent dynamics linear or nonlinear? What is the form of the drift governing the latent state evolution? How random is the evolution, i.e., what is the diffusion coefficient? How is the randomness structured, i.e., what is the covariance of the possibly multi-dimensional driving Brownian motion?
2. How are the observations related to the latent states? E.g., is the observation model linear or nonlinear? How noisy are the observations? Is the observation noise Gaussian or non-Gaussian?

Due to the range and combination of choices available to the modeler when defining a CD-SSM, these can be tailored to capture the specific characteristics of the system being modeled, making CD-SSMs highly versatile and applicable across a wide range of domains.

However, the flexibility and expressiveness of CD-SSMs come at the cost of increased complexity (in implementation and in computational resources) for state inference and parameter estimation tasks. Hence, efficient and robust tools for CD-SSM modeling, inference and learning are crucial to researchers in both theoretical and applied domains.

In general, for a given set of observations $Y_K = [y(t_1),\ \ldots,\ y(t_K)]$ of a CD-SSM, the main objectives of interest to theorists and practitioners are:

- **Filtering**: to estimate the distribution of $x(t_K) \mid Y_K,\ \theta$.
- **Smoothing**: to estimate the distribution of $\{x(t)\}_{t \le t_K} \mid Y_K,\ \theta$.
- **Forecasting**: to estimate the distribution of $\{x(t)\}_{t > t_K} \mid Y_K,\ \theta$.
- **Parameter learning**: to estimate $\theta \mid Y_K$, either point-wise or in distribution.

With this context in mind, we present `cd-dynamax`: a **CD-SSM modeling framework, with inference and learning algorithms** for the tasks outlined above.

`cd-dynamax` is a `JAX`-based open-source Python package for continuous-discrete state space modeling and inference:

- `cd-dynamax` not only supports canonical CD-SSMs, e.g., the continuous-discrete linear dynamical system (CD-LGSSM), but allows for easy construction, modeling and inference of flexible CD-SSMs as needed: the practitioner is only required to specify the drift function f, the diffusion coefficient L of the latent SDE, and the observation function h for each specific model of interest.
- `cd-dynamax`'s flexibility with respect to model definition means that users can define and work with a wide range of custom CD-SSMs that include mechanistic and/or flexible (e.g., neural network) components for the latent dynamics and observation models.
- `cd-dynamax` provides robust implementations of several, state-of-the-art continuous-discrete inference algorithms in an efficient, autodifferentiable framework, enabling the use of modern general-purpose libraries for parameter inference (e.g., stochastic gradient descent, Hamiltonian Monte Carlo). `cd-dynamax` is designed to allow users to flexibly choose among a host of algorithms for their specific CD-SSM model and application.

Statement of need

`cd-dynamax` is a `JAX`-based (Bradbury et al., 2018) open-source Python package for continuous-discrete state space modeling, where observations are made at specified discrete times (rather

than at regular intervals) and are driven by latent SDEs.

Other Python libraries exist for state space modeling (Aicher et al., 2025; Corenflos & Särkkä, 2021; Johnson, 2020; Lee et al., 2023; S. Linderman et al., 2020; Weiss et al., 2024), which are primarily focused on Hidden Markov Models and discrete-time state space models, with their corresponding Bayesian inference algorithms. Amongst the JAX-native libraries, `dynamax` (S. W. Linderman et al., 2025) provides a comprehensive framework for discrete-time state space modeling and inference, while (Lysy, 2023) offers particle filtering capabilities for discrete-time state space models. The `rodeo` (Wu & Lysy, 2025) library provides JAX-based probabilistic numerics tools for approximating likelihoods of noisy partially observed data under deterministic continuous-time systems (i.e., ordinary differential equations) but, crucially, does not address state-stochasticity.

To the best of our knowledge, there is no existing Python-based library that provides a comprehensive framework for continuous-discrete state space modeling and inference.

`cd-dynamax` fills this gap by providing a user-friendly interface for defining CD-SSMs, along with efficient implementations of state-of-the-art filtering, smoothing, forecasting, and parameter learning algorithms specifically designed for continuous-discrete dynamical systems:

- `cd-dynamax` extends the `dynamax` (S. W. Linderman et al., 2025) library by exploiting `diffrax` (Kidger, 2021) —a JAX-based library providing numerical differential equation solvers— to enable accurate and efficient simulation of continuous-time dynamics, as well as gradient-based backpropagation through automatic differentiation. By relying on JAX, `cd-dynamax` supports automatic autodifferentiation and just-in-time (JIT) compilation for hardware acceleration on CPU, GPU, and TPU machines.
- `cd-dynamax` is particularly suited for domains where continuous-time dynamics are prevalent, and observations are collected at irregular intervals. Its internal structure is designed to interact with any CD-SSM model (linear or nonlinear) in a unified way (rather than being treated separately) for model definition, state-inference and system-identification, producing a consistently structured library.
- `cd-dynamax` is developed for both methodological researchers (interested in advancing state space modeling and inference algorithms) and practitioners (interested in applying CD-SSMs to real-world problems in fields such as systems biology, neuroscience, finance, and engineering).

On the importance of continuous-time modeling

While continuous-time SSMs can be represented as discrete-time SSMs when sampling at fixed intervals, there remain fundamental differences between these two modeling paradigms: the former cannot be perfectly translated into the latter without loss of information or introduction of artifacts.

Succinctly put, the relationship between the discrete and continuous frameworks is one of approximation —a mapping that may involve significant information loss: while it is possible to derive a discrete-time model from a continuous-time model through discretization, the reverse process of obtaining a continuous-time model from a discrete-time model is generally ill-posed and non-unique.

There are two fundamental issues introduced by discretization:

- **Information Loss**: Sampling inevitably obscures the system's true dynamics, distorting the signal in a process known as aliasing. Discretization results in the loss of inter-sample behavior, and hence, a system can appear stable at the sampling points while actually experiencing oscillations between them.
- **Artifact Creation**: The choice of a discrete-time representation of a model, along with the definition of its sampling interval, can create non-physical, artificial dynamics.

Discretization choices can introduce entirely new behaviors not present in the original continuous-time system. For instance, naive sampling can induce the emergence (or destruction) of chaos in simple discrete maps (entirely absent, or assured, in their stable continuous-time counterparts) or instability of control-systems (where a stable continuous-time system can be rendered catastrophically unstable by choosing incorrect sampling intervals).

There are **significant benefits of a continuous-time treatment of dynamical systems**:

- *Data agnosticism*: continuous-time models are inherently suited to handle real-world, irregularly-spaced, and missing data: they model the underlying process, not the measurement grid. Thus, continuous-time models naturally generalize to arbitrary observation time grids without retraining or modification.
- *Discretize at the end, not at the beginning*: a continuous-time framing allows for discretization choices to be deferred until the final stages of analysis, enabling the use of adaptive solvers and multi-rate sampling strategies that can better capture the system's dynamics. A history of successes in numerical analysis has shown that delaying discretization until the final stages of computation often leads to more accurate and stable results.
- *Physical interpretability*: continuous-time model parameters represent fundamental, invariant physical properties of the system (e.g., reaction rates, physical constants, clearance rates), whereas discrete-time parameters are a conflation of physical properties and the choices of sampling intervals. In physics-aware modeling, prior knowledge is often most naturally expressed in a continuous-time formulation.
- *First-principles-based theory*: continuous-time models, expressed as differential equations, are the "first principles" foundation for many physical and life sciences. The discrete-time model is most accurately viewed as a subsequent numerical implementation or approximation of this theoretical truth.

`cd-dynamax` aims to facilitate the adoption of continuous-discrete state space models in various scientific domains, removing the user's burden of implementing continuous-time dynamical system models, solvers, and inference algorithms from scratch. Instead, users can focus on defining their inductive modeling biases, their priors and high-level inference choices.

Additionally, it enables methodological research in continuous-time state space modeling and inference by providing a single, flexible framework for experimentation, development, and benchmarking of new algorithms.

CD-dynamax modeling and inference framework

For researchers and practitioners interested in continuous-time modeling, `cd-dynamax` provides a robust, efficient, and user-friendly framework for CD-SSM modeling, inference, and learning.

`cd-dynamax` provides (i) continuous-discrete linear and nonlinear state space model definitions, (ii) state-of-the-art filtering and smoothing algorithm implementations, and (iii) flexible tools for system identification and model parameter estimation.

Currently, `cd-dynamax` offers:

1. A set of modular definitions of CD-SSM models, capturing both linear (CD-LGSSM) and nonlinear (CD-NLGSSM) dynamics and observation functions, seamlessly incorporating non-regular, noisy observation time instants. More information about state space modeling can be found in the textbooks by Murphy (2023) and Särkkä & Svensson (2023).
2. Low-level, probabilistic inference algorithms for filtering and smoothing. There exist many algorithms for state inference and parameter estimation in CD-SSMs (Särkkä & Svensson,

2023), e.g., Extended Kalman Filter/Smoother, Unscented Kalman Filter/Smoother, Particle Filter/Smoother. Specifically, `cd-dynamax` provides `JAX` implementations for:

- Kalman filtering and smoothing for linear Gaussian CD-SSMs,
- Extended Kalman filtering and smoothing for nonlinear CD-SSMs,
- Unscented and Ensemble Kalman filtering for nonlinear CD-SSMs.

3. A high-level interface for constructing and fitting probabilistic SSMs. We provide readily usable functions for:

   - point-estimation of model parameters via gradient-based or black-box optimization (Scipy, Scipy-jaxopt) of the (approximate) marginal log-likelihood
   - Bayesian posterior parameter estimation, i.e., Markov Chain Monte Carlo via the BlackJAX library.

The publicly available `cd-dynamax` documentation and demos provide informative resources describing the use of `cd-dynamax` for CD-SSM modeling, filtering, smoothing, forecasting and fitting to data, for experts and newcomers alike.

Acknowledgements

Iñigo Urteaga acknowledges the support of ''la Caixa'' foundation's LCF/BQ/PI22/11910028 award, as well as funds by MICIU/AEI/10.13039/501100011033 and the BERC 2022-2025 program funded by the Basque Government. Matthew Levine acknowledges support by Basis Research Institute and the Eric and Wendy Schmidt Center at the Broad Institute of MIT and Harvard.

References

Aicher, C., Putcha, S., Nemeth, C., Fearnhead, P., & Fox, E. (2025). Stochastic gradient MCMC for nonlinear state space models. *Bayesian Analysis*, *20*(1). https://doi.org/10.1214/23-BA1395

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., & Zhang, Q. (2018). *JAX: Composable transformations of Python+NumPy programs* (Version 0.3.13). http://github.com/google/jax

Corenflos, A., & Särkkä, S. (2021). *Code companion for Bayesian Filtering and Smoothing* (Version 1.0). https://github.com/EEA-sensors/Bayesian-Filtering-and-Smoothing

Johnson, M. J. (2020). *PyHSMM: Bayesian inference in HSMMs and HMMs* (Version 0.0.0). https://github.com/mattjj/pyhsmm

Kidger, P. (2021). *On Neural Differential Equations* [PhD thesis, University of Oxford]. https://docs.kidger.site/diffrax/

Lee, M. A., Yi, B., Martín-Martín, R., Savarese, S., & Bohg, J. (2023). *TorchFilter: Differentiable particle filtering in PyTorch*. https://github.com/stanford-iprl-lab/torchfilter

Linderman, S. W., Chang, P., Harper-Donnelly, G., Kara, A., Li, X., Duran-Martin, G., & Murphy, K. (2025). Dynamax: A python package for probabilistic state space modeling with JAX. *Journal of Open Source Software*, *10*(108), 7069. https://doi.org/10.21105/joss.07069

Linderman, S., Antin, B., Zoltowski, D., & Glaser, J. (2020). *SSM: Bayesian Learning and Inference for State Space Models* (Version 0.0.1). https://github.com/lindermanlab/ssm

Lysy, M. (2023). *PFJax: Particle filtering in JAX*. https://github.com/mlysy/pfjax

Murphy, K. P. (2023). *Probabilistic machine learning: Advanced topics*. MIT Press. http://probml.github.io/book2

Särkkä, S., & Svensson, L. (2023). *Bayesian filtering and smoothing* (Vol. 17). Cambridge University Press. https://doi.org/10.1017/CBO9781139344203

Weiss, R., Du, S., Grobler, J., Cournapeau, D., Pedregosa, F., Varoquaux, G., Mueller, A., Thirion, B., Nouri, D., Louppe, G., Vanderplas, J., Benediktsson, J., Buitinck, L., Korobov, M., McGibbon, R., Lattarini, S., Niculae, V., Gramfort, A., Lebedev, S., … Rockhill, A. (2024). *Hmmlearn* (Version 0.3.2). https://github.com/hmmlearn/hmmlearn

Wu, M., & Lysy, M. (2025). *Rodeo: Probabilistic methods of parameter inference for ordinary differential equations*. https://arxiv.org/abs/2506.21776