Individualized Dosing Dynamics via Neural Eigen Decomposition

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Dosing models often use differential equations to model biological dynamics. Neural differential equations in particular can learn to predict the derivative of a process, which permits predictions at irregular points of time. However, this temporal flexibility often comes with a high sensitivity to noise, whereas medical problems often present high noise and limited data. Moreover, medical dosing models must generalize reliably over individual patients and changing treatment policies. To address these challenges, we introduce the Neural Eigen Stochastic Differential Equation algorithm (**NESDE**). NESDE provides individualized modeling (using patient-level parameters); generalization to new treatment policies (using decoupled control); tunable expressiveness according to the noise level (using piecewise linearity); and fast, continuous, closed-form prediction (using spectral representation). We demonstrate the robustness of NESDE in real medical problems, and use the learned dynamics to publish simulated medical gym environments.

1 Introduction

Sequential forecasting in irregular points of time is required in many real-world problems, such as modeling dosing dynamics of various medicines (pharmacodynamics). Consider a patient whose physiological or biochemical state requires continuous monitoring, while blood tests are only available with a limited frequency. Pharmacodynamics models often rely on an ordinary differential equation models (ODE) for forecasting. Additional expressiveness can be obtained via customized learned models, such as neural-ODE, which learns to predict the derivative of the process [Chen et al., 2018, Liu et al., 2019]. By predicting the *derivative*, neural-ODE can make irregular predictions at flexible time-steps, unlike regular models that operate in constant time-steps (e.g., Kalman filter, Kalman [1960] and recurrent neural networks, Rumelhart et al. [1986]).

However, real-world forecasting remains a challenge for several reasons. First, the variation between patients often requires personalized modeling. Second, neural-ODE methods are often data-hungry: they aggregate numerous derivatives provided by a non-linear neural network, which is often sensitive to noise. Training over a large dataset may stabilize the predictions, but data is often limited. Third, most neural-ODE methods only provide a point-estimate, while uncertainty estimation is often critical in medical settings. Fourth, for every single prediction, the neural-ODE runs a numeric ODE solver, along with multiple neural network calculations of the derivative. This computational overhead in inference may limit latency-sensitive applications.

A fifth challenge comes from control. In the framework of retrospective forecasting , a control signal (drug dosage) is often considered part of the observation [De Brouwer et al., 2019]. However, this approach raises difficulties if the control is observed at different times or more frequently than other observations. If the control is part of the model output, it may also bias the train loss away from

Submitted to the DLDE-III Workshop in the 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

the true objective. Finally, by treating control and observations together, the patterns learned by the model may overfit the control policy used in the data – and generalize poorly to new policies.

Generalization to out-of-distribution control policies is essential when the predictive model supports decision-making, as the control policy may be affected by the model. Such decision-making is an important use-case of sequential prediction: model-based reinforcement learning and control problems require a reliable model [Moerland et al., 2020, Angermueller et al., 2019], in particular in risk-sensitive control [Yu et al., 2021, Greenberg et al., 2022, Greenberg and Mannor, 2021].

Contribution:

- We characterize the main challenges in continuous forecasting for medication dosing control.
- We design the Neural Eigen-SDE algorithm, which addresses the challenges described above.
- We use NESDE to improve modeling accuracy in two medication dosing processes. Based on the learned models, we simulate gym environments for future research of healthcare control.

2 Neural Eigen-SDE

Problem setup: We focus on online sequential prediction of a process $Y(t) \in \mathbb{R}^m$. To predict $Y(t_0)$ at a certain t_0, we use noisy observations $\hat{Y}(t)$ (at given times $t < t_0$); a control signal $u(t) \in \mathbb{R}^k$ ($\forall t < t_0$); offline data of Y and u; and samples of per-sequence contextual information $C \in \mathbb{R}^{d_c}$.

We assume the observations $Y(t)$ to originate from an unobservable latent process $X(t) \in \mathbb{R}^n$:

$$dX(t) = F_C\big(X(t), u(t)\big),\ Y(t) = X(t)_{1:m},\ \hat{Y}(t) = Y(t) + \nu_C(t) \tag{1}$$

where F_C is a stochastic operator (which may depend on C); Y is the first m coordinates of X; $\hat{Y}$ is the corresponding observation; and $\nu_C(t)$ is its i.i.d Gaussian noise with zero-mean and covariance $R_C \in \mathbb{R}^{m\times m}$ (which may also depend on C). Our goal is to predict Y. If Y is not available, we measure our prediction accuracy against $\hat{Y}$. The control $u(t)$ is modeled separately from $\hat{Y}$.

Model: The Neural Eigen-SDE algorithm (NESDE, Fig. 1) predicts the signal $Y(t)$ continuously at any required time t. It relies on a piecewise linear approximation which reduces Eq. (1) into:

$$\forall t \in \mathcal{I}_i:\ dX(t) = [A_i \cdot (X(t) - \alpha) + B \cdot u(t)] + dW(t) \tag{2}$$

where $\mathcal{I}_i = (t_i, t_{i+1})$ is a time interval, dW is a Brownian noise with covariance matrix Q_i, and $A_i \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{n\times k}, Q_i \in \mathbb{R}^{n\times n}, \alpha \in \mathbb{R}^n$ form the linear dynamics model corresponding to the interval $\mathcal{I}_i$. To solve Eq. (2) within every $\mathcal{I}_i$, NESDE has to learn the parameters $\{A_i, Q_i\}_i, \alpha, B$.

The end of $\mathcal{I}_{i-1}$ typically represents one of two events: either an update of the dynamics A, or the arrival of a new observation. A new observation at time t_i triggers an update of $X(t_i)$ according to the conditional distribution $X(t_i)|\hat{Y}(t_i)$. Then, the prediction continues for $\mathcal{I}_i$ according to Eq. (2).

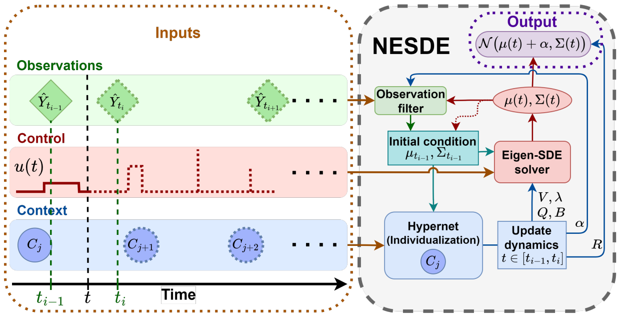


Figure 1: NESDE algorithm. Hypernet uses the context and the estimated state to determine the SDE parameters; Eigen-SDE solver uses them to make predictions for the next time-interval; the filter updates the state upon arrival of a new observation, which initiates a new interval. For more frequent updates of the dynamics, the initial condition becomes the last prediction.

Eigen-SDE solver (ESDE) – spectral representation: A_i is only represented implicitly through the parameters V, λ defining its eigenfunction $\Phi(t)$ (Appendix C). The spectral representation allows solving $X(t)$ analytically for any $t \in \mathcal{I}_i$ at once. This is particularly useful in the sparsely-observable setup. Many SDE solvers apply recursive numeric integration [Chen et al., 2018, De Brouwer et al., 2019]. In NESDE, however, thanks to the spectral decomposition, the integration only depends on known functions of t, hence the computation can be paralleled. Furthermore, if the control has analytically-integrable form over $\mathcal{I}_i$, Appendix E shows how to solve the integration *analytically*.

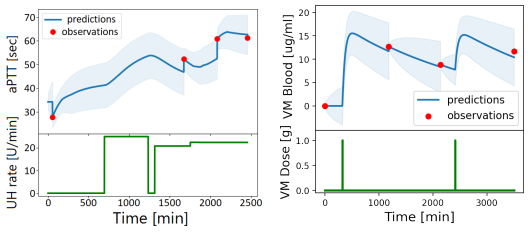
Updating solver and filter parameters: NESDE provides the parameters V, λ, Q, B, α to the Eigen-SDE solver, as well as the noise R to the observation filter. As NESDE assumes a *piecewise* linear model, it separates the time into intervals $\mathcal{I}_i = (t_i, t_{i+1})$ (the interval length is a hyperparameter), and uses a dedicated model to predict new parameters at the beginning t_i of every interval.

The model receives the current state $X(t_i)$ and the context C, then returns the parameters for $\mathcal{I}_i$. We use Hypernet [Ha et al., 2016], where one neural network $g_1(C; \Theta)$ returns the weights of another: $(V, \lambda, Q, B, \alpha, R) \coloneqq g_2(X; W) = g_2(X; g_1(C; \Theta))$. In our implementation, V, λ, Q are renewed every time interval, α and R are predicted once per sequence, and B is a global parameter.

Training: The parameters of NESDE are the control mapping B and Hypernet's parameters Θ (which determine the rest of the parameters). To optimize them, the training relies on a dataset of sequences of control signals $\{u_{seq}(t_j)\}_{seq,j}$, states and observations $\{(Y_{seq}(t_j), \hat{Y}_{seq}(t_j))\}_{seq,j}$. The latent space dimension n and the model-update frequency Δt are determined as hyperparameters. Then, we use the standard Adam optimizer [Diederik P. Kingma, 2015] to optimize the parameters with respect to the loss $NLL(j) = -\log P(Y(t_j) | \mu(t_j), \Sigma(t_j))$.

3 Experiments: Medication Dosing

As discussed in Section 1, many medical applications could potentially benefit from ODE-based methods. Specifically, we address medication dosing problems, where observations are often sparse, the dosing is a control signal, and uncertainty estimation is crucial. We test **NESDE** on two such domains. As baselines, we choose recent ODE-based methods that provide Bayesian uncertainty estimation: **GRU-ODE**-Bayes [De Brouwer et al., 2019] and **CRU** [Schirmer et al., 2022]. Additionally, we design a dedicated **LSTM** model that supports irregular predictions, as described in Appendix I.2. We also add a **naive** model with "no-dynamics" (predicts the last observed value).



(a) Heparin (b) Vancomycin

Figure 2: A sample of patients from (a) the UH dosing dataset, and (b) the VM dosing dataset. The lower plots correspond to medication dosage (UH in (a) and VM in (b)). The upper plots correspond to the continuous prediction of NESDE (aPTT levels in (a) and VM concentration in (b)), with 95% confidence intervals. In both settings, the prediction at every point relies on all the observations up to that point.

The benchmarks in this section were derived from the MIMIC-IV dataset [Johnson et al., 2020]. The dataset contains a vast amount of side-information (e.g., weight and heart rate). We use some of this information as an additional input – for each model according to its structure (context-features for the hyper-network of NESDE, covariates for GRU-ODE-Bayes, state variables for CRU, and embedding units for the LSTM). Some context features correspond to online measurements which are updated frequently. We constraint the process eigenvalues λ to be negative, to reflect the stability of the biophysical processes. Indeed, the spectral representation of NESDE provides us with a natural way to incorporate such domain knowledge, which often cannot be used otherwise. For all models, in both domains, we use a 60-10-30 train-validation-test data partition. See more implementation details in Appendix I.

Unfractionated Heparin Dosing: Unfractionated Heparin (UH) is a widely used anticoagulant. It may be given in a continuous infusion to patients with life-threatening clots. The drug's activity is usually monitored using a lab test performed on a blood sample: activated Partial Thromboplastin Time (aPTT) test. The clinical objective is to keep the aPTT level in a certain range. The problem poses several challenges: different patients respond differently; monitoring and control are required in higher frequency than measurements; and deviations of the aPTT from the objective range may be fatal. Here we focus on continuous prediction as a key component for aPTT control.

Following the preprocessing described in Appendix I.1, we derive 5866 trajectories of a continuous UH control signal, an irregularly-observed aPTT signal, and 42 context features. It is known that UH does not affect the aPTT directly (Delavenne et al. [2017]); thus, we mask the control mapping B

Table 1: Test mean square errors (MSE) and negative log-likelihood (NLL, for models that provide probabilistic prediction) in the medication-dosing benchmarks.

Model	UH Dosing		Vancomycin Dosing	
	MSE	NLL	MSE	NLL
Naive	613.3 ± 13.48	–	112.2 ± 16.4	–
LSTM	482.1 ± 6.52	–	92.89 ± 11.3	–
GRU-ODE-Bayes	491 ± 6.88	4.52 ± 0.008	80.54 ± 11.8	6.38 ± 0.12
CRU	450.4 ± 8.27	4.49 ± 0.012	76.4 ± 12.8	3.87 ± 0.2
NESDE (ours)	$\mathbf{411.2 \pm 7.39}$	$\mathbf{4.43 \pm 0.01}$	$\mathbf{70.71 \pm 12.3}$	$\mathbf{3.69 \pm 0.13}$

to have no direct effect on the aPTT metric, but only on the latent variable. The control (UH) and observations (aPTT) are one-dimensional ($m = 1$), and we set the whole state dimension to $n = 4$.

Vancomycin Dosing: Vancomycin (VM) is an antibiotic that has been in use for several decades. However, the methodology of dosing VM remains a subject of debate [Rybak et al., 2009], and there is a significant degree of variability among patients [Marsot et al., 2012]. The dosage of VM is critical; it could become toxic if overdosed [Filippone et al., 2017], and ineffective if underdosed. The VM level in the blood can be measured through lab tests, which are often infrequent.

Here, the goal is to predict the VM concentration in the blood at any given time, where the dosage and other patient measurements are known. Following the preprocessing described in Appendix I.1, the dataset derives 3564 trajectories of VM dosages at discrete times, blood concentration of VM ($m = 1$) at irregular times, and similarly to UH dosing, 42 context features. This problem is less noisy than the UH dosing problem, as the task is to learn the direct dynamics of the VM concentration, and not the effects of the antibiotics. The whole state dimension is set to $n = 2$, and we also mask the control mapping B to have no direct effect on the VM concentration.

3.1 Results

Fig. 2 displays sample trajectories predicted by NESDE in both domains. As summarized in Table 1, NESDE outperforms the other baselines in both UH and VM dosing tasks, in terms of both square errors (MSE) and likelihood (NLL). For the UH dosing problem, Fig. 3 also presents the errors vs. prediction horizon (the time passed since the last observation). Evidently, **NESDE provides the best accuracy in all the horizons**. While most of the data corresponds to horizons of 5-7 hours (see Fig. 12 in the appendix), NESDE provides reliable prediction at other horizons as well. By contrast, LSTM and GRU-ODE-Bayes have difficulty with short horizons; they only become competitive with the *naive* model after 6 hours. CRU provides more robust predictions, but is still outperformed by NESDE.

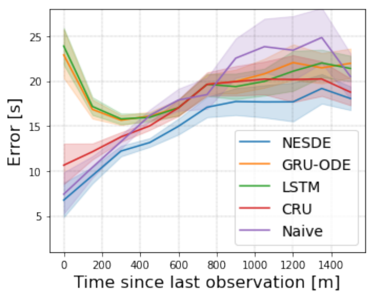


Figure 3: aPTT prediction errors in the UH problem, vs. the time passed since the last aPTT test.

Despite the large range of aPTT levels in the data, 50% of all the predictions have errors lower than $12.4s$ – an accuracy level that is considered clinically safe. Fig. 3 shows that indeed, up to 3 hours after the last lab test, the average error is smaller than $10s$.

4 Conclusion

Motivated by medical forecasting and control problems, we characterized a set of challenges in modeling dosing dynamics: sample efficiency, uncertainty estimation, personalized modeling, continuous inference and generalization to different control. To address them, we introduced the novel NESDE algorithm, based on a stochastic differential equation with spectral representation. We demonstrated the reliability of NESDE in a variety of synthetic (Appendix H) and real data experiments.

As demonstrated in the experiments, NESDE provides robust, reliable and uncertainty-aware continuous *forecasting*. This paves the way to development of *decision making* in continuous high-noise decision processes, including medical treatment, finance and operations management. Future research may address medical optimization via both control policies (e.g., to control medication dosing) and sampling policies (to control measurements timing, e.g., of blood tests).

References

Christof Angermueller, David Dohan, David Belanger, Ramya Deshpande, Kevin Murphy, and Lucy Colwell. Model-based reinforcement learning for biological sequence design. In *International conference on learning representations*, 2019.

Gianluca Bontempi, Souhaib Ben Taieb, and Yann-Aël Le Borgne. Machine learning strategies for time series forecasting. *Lecture Notes in Business Information Processing*, 138, 01 2013. doi: 10.1007/978-3-642-36318-4_3.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. *arXiv preprint arXiv:1806.07366*, 2018.

Huseyin Coskun, Felix Achilles, Robert DiPietro, Nassir Navab, and Federico Tombari. Long short-term memory kalman filters:recurrent neural estimators for pose regularization. *ICCV*, 2017. URL `https://github.com/Seleucia/lstmkf_ICCV2017`.

Edward De Brouwer, Jaak Simm, Adam Arany, and Yves Moreau. Gru-ode-bayes: Continuous modeling of sporadically-observed time series. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/file/455cb2657aaa59e32fad80cb0b65b9dc-Paper.pdf`.

X. Delavenne, E. Ollier, S. Chollet, F. Sandri, J. Lanoiselée, S. Hodin, A. Montmartin, J.-F. Fuzellier, P. Mismetti, and L. Gergelé. Pharmacokinetic/pharmacodynamic model for unfractionated heparin dosing during cardiopulmonary bypass. *BJA: British Journal of Anaesthesia*, 118(5):705–712, May 2017. ISSN 0007-0912. doi: 10.1093/bja/aex044. URL `https://doi.org/10.1093/bja/aex044`.

Jimmy Ba Diederik P. Kingma. Adam: A method for stochastic optimization. *ICLR*, 2015. URL `https://arxiv.org/abs/1412.6980`.

Morris L. Eaton. *Multivariate Statistics: a Vector Space Approach*. John Wiley and Sons, 1983.

Edward J Filippone, Walter K Kraft, and John L Farber. The nephrotoxicity of vancomycin. *Clinical Pharmacology & Therapeutics*, 102(3):459–469, 2017.

Chang Gao, Junkun Yan, Shenghua Zhou, Bo Chen, and Hongwei Liu. Long short-term memory-based recurrent neural networks for nonlinear target tracking. *Signal Processing*, 164, 05 2019. doi: 10.1016/j.sigpro.2019.05.027.

Ido Greenberg and Shie Mannor. Detecting rewards deterioration in episodic reinforcement learning. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 3842–3853. PMLR, 18–24 Jul 2021. URL `https://proceedings.mlr.press/v139/greenberg21a.html`.

Ido Greenberg, Netanel Yannay, and Shie Mannor. Noise estimation is not optimal: How to use kalman filter the right way. *arXiv preprint arXiv:2104.02372*, 2021.

Ido Greenberg, Yinlam Chow, Mohammad Ghavamzadeh, and Shie Mannor. Efficient risk-averse reinforcement learning. *Advances in Neural Information Processing Systems*, 2022.

David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

Luis Herrera, Hector Pomares, I. Rojas, Alberto Guillén, Alberto Prieto, and Olga Valenzuela. Recursive prediction for long term time series forecasting using advanced models. *Neurocomputing*, 70:2870–2880, 10 2007. doi: 10.1016/j.neucom.2006.04.015.

Florian Herzog. Stochastic differential equations, 2013. URL `https://ethz.ch/content/dam/ethz/special-interest/mavt/dynamic-systems-n-control/idsc-dam/Lectures/Stochastic-Systems/SDE.pdf`.

Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. *Neural Computation*, 1997. URL `https://direct.mit.edu/neco/article/9/8/1735/6109/Long-Short-Term-Memory`.

A Johnson, L Bulgarelli, T Pollard, S Horng, LA Celi, and R Mark. Mimic-iv, 2020.

R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35–45, 03 1960. ISSN 0021-9223. doi: 10.1115/1.3662552. URL `https://doi.org/10.1115/1.3662552`.

John F. Kolen and Stefan C. Kremer. *Gradient Flow in Recurrent Nets: The Difficulty of Learning LongTerm Dependencies*, pages 237–243. Wiley-IEEE Press, 2001. doi: 10.1109/9780470544037.ch14.

Rahul G. Krishnan, Uri Shalit, and David Sontag. Deep kalman filters, 2015.

Weihua Li, Sirish L. Shah, and Deyun Xiao. Kalman filters in non-uniformly sampled multirate systems: For fdi and beyond. *Automatica*, 44(1):199–208, jan 2008. ISSN 0005-1098. doi: 10.1016/j.automatica.2007.05.009. URL `https://doi.org/10.1016/j.automatica.2007.05.009`.

Xuanqing Liu, Si Si, Qin Cao, Sanjiv Kumar, and Cho-Jui Hsieh. Neural sde: Stabilizing neural ode networks with stochastic noise. *arXiv preprint arXiv:1906.02355*, 2019.

James Lu, Kaiwen Deng, Xinyuan Zhang, Gengbo Liu, and Yuanfang Guan. Neural-ode for pharmacokinetics modeling and its advantage to alternative machine learning models in predicting new dosing regimens. *Iscience*, 24(7):102804, 2021.

Amélie Marsot, Audrey Boulamery, Bernard Bruguerolle, and Nicolas Simon. Vancomycin. *Clinical pharmacokinetics*, 51(1):1–13, 2012.

Thomas M. Moerland, Joost Broekens, and Catholijn M. Jonker. Model-based reinforcement learning: A survey. *CoRR*, abs/2006.16712, 2020. URL `https://arxiv.org/abs/2006.16712`.

P. A. P. Moran and Peter Whittle. Hypothesis testing in time series analysis, 1951.

Shamim Nemati, Mohammad M Ghassemi, and Gari D Clifford. Optimal medication dosing from suboptimal clinical examples: A deep reinforcement learning approach. In *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 2978–2981. IEEE, 2016.

Dominic A. Neu, Johannes Lahann, and Peter Fettke. A systematic literature review on state-of-the-art deep learning methods for process prediction. *CoRR*, abs/2101.09320, 2021. URL `https://arxiv.org/abs/2101.09320`.

Se Yong Park and Anant Sahai. Intermittent kalman filtering: Eigenvalue cycles and nonuniform sampling. In *Proceedings of the 2011 American Control Conference*, pages 3692–3697, 2011. doi: 10.1109/ACC.2011.5991285.

Guy Revach, Nir Shlezinger, Xiaoyong Ni, Adria Lopez Escoriza, Ruud J. G. van Sloun, and Yonina C. Eldar. Kalmannet: Neural network aided kalman filtering for partially known dynamics, 2021.

Yulia Rubanova, Ricky T. Q. Chen, and David K Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/file/42a6845a557bef704ad8ac9cb4461d43-Paper.pdf`.

David E. Rumelhart et al. Learning representations by back-propagating errors. *Nature*, 1986. URL `https://www.nature.com/articles/323533a0`.

Michael Rybak, Ben Lomaestro, John C. Rotschafer, Jr. Moellering, Robert, William Craig, Marianne Billeter, Joseph R. Dalovisio, and Donald P. Levine. Therapeutic monitoring of vancomycin in adult patients: A consensus review of the American Society of Health-System Pharmacists, the Infectious Diseases Society of America, and the Society of Infectious Diseases Pharmacists. *American Journal of Health-System Pharmacy*, 66(1):82–98, 01 2009. ISSN 1079-2082. doi: 10.2146/ajhp080434.

Mona Schirmer, Mazin Eltayeb, Stefan Lessmann, and Maja Rudolph. Modeling irregular time series with continuous recurrent units. In *International Conference on Machine Learning*, pages 19388–19405. PMLR, 2022.

B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M.I. Jordan, and S.S. Sastry. Kalman filtering with intermittent observations. *IEEE Transactions on Automatic Control*, 49(9):1453–1464, 2004. doi: 10.1109/TAC.2004.834121.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

Yi-Jen Wang and Chin-Teng Lin. Runge-kutta neural network for identification of dynamical systems in high accuracy. *IEEE Transactions on Neural Networks*, 9(2):294–307, 1998.

Chao Yu, Jiming Liu, Shamim Nemati, and Guosheng Yin. Reinforcement learning in healthcare: A survey. *ACM Computing Surveys (CSUR)*, 55(1):1–36, 2021.

Appendices

Table of Contents

A NESDE Algorithm

Algorithm 1 NESDE

```
  Input: context $C$; control signal $u(t)$; update times $\mathcal{I} \in \mathbb{R}^T$; prediction times $\{P_{\mathcal{I}_i}\}_{\mathcal{I}_i \in \mathcal{I}}$
  Initialize: $\mu, \Sigma, \alpha, R \leftarrow \textbf{Prior}(C)$
  for $\mathcal{I}_i$ in $\mathcal{I}$: do
    $V, \lambda, Q, B, \alpha, R \leftarrow \textbf{Hypernet}(C, \mu, \Sigma)$
    for $t$ in $P_{\mathcal{I}_i}$ do
      $\mu_t, \Sigma_t \leftarrow \textbf{ESDE}\big(\mu, \Sigma, u, t; V, \lambda, Q, B\big)$
      predict: $\tilde{Y}_t \sim \mathcal{N}\big(\mu_t + \alpha, \Sigma_t + R\big)$
      if given observation $\hat{Y}_t$ then
        $\mu, \Sigma \leftarrow \textbf{Filter}(\mu_t, \Sigma_t, R, \hat{Y}_t)$
      end if
    end for
  end for
```

B Related Work

Classic filtering: Classic models for sequential prediction in time-series include ARIMA models [Moran and Whittle, 1951] and the Kalman filter (KF) [Kalman, 1960]. The KF provides probabilistic distributions and in particular uncertainty estimation. While the classic KF is limited to linear dynamics, many non-linear extensions have been suggested [Krishnan et al., 2015, Coskun et al., 2017, Revach et al., 2021, Greenberg et al., 2021]. However, such models are typically limited to a constant prediction horizon (time-step). Longer-horizon predictions are often made by applying the model recursively [Herrera et al., 2007, Bontempi et al., 2013], This poses a significant challenge to many optimization methods [Kolen and Kremer, 2001], as also demonstrated in Appendix H.5.

Limited types of irregularity can also be handled by KF with intermittent observations [Park and Sahai, 2011, Sinopoli et al., 2004] or periodical time-steps [Li et al., 2008].

Recurrent neural networks: Sequential prediction is often addressed via neural network models, relying on architectures such as RNN [Rumelhart et al., 1986], LSTM [Hochreiter and Schmidhuber, 1997] and transformers [Vaswani et al., 2017]. LSTM, for example, is a key component in many SOTA algorithms for non-linear sequential prediction [Neu et al., 2021]. LSTM can be extended to a filtering framework to alternately making predictions and processing observations, and even to provide uncertainty estimation [Gao et al., 2019]. However, these models are typically limited to constant time-steps, and thus suffer from the limitations discussed above.

Neural-ODE models: Parameterized ODE models can be optimized by propagating the gradients of a loss function through an ODE solver [Chen et al., 2018, Liu et al., 2019, Rubanova et al., 2019]. By predicting the process *derivative* and using an ODE solver in real-time, these methods can choose the effective time-steps flexibly. Uncertainty estimation can be added via process variance prediction [De Brouwer et al., 2019]. However, since neural-ODE methods learn a non-linear dynamics model, the ODE solver operates numerically and recursively on top of multiple neural network calculations. This affects running time, training difficulty and data efficiency as discussed above. While neural-ODE models have been studied for medical applications with irregular data [Lu et al., 2021], simpler models are commonly preferred in practice. For example, the effects of Heparin on blood coagulation is usually modeled by either using discrete models [Nemati et al., 2016] or manually based on domain knowledge [Delavenne et al., 2017].

Our method uses SDE with piecewise linear dynamics (note this is *different* from a piecewise linear process). The linear dynamics per time interval permit efficient and continuous closed-form forecasting of both mean and covariance. Schirmer et al. [2022] also rely on a linear ODE model, but only support operators with real-valued eigenvalues (which limits the modeling of periodic processes), and do not separate control signal from observations (which limits generalization to out-of-distribution control). Our piecewise linear architecture, tested below against alternative methods including De Brouwer et al. [2019] and Schirmer et al. [2022], is demonstrated to be more robust to noisy, sparse or small datasets, even under out-of-distribution control policies.

C Preliminaries: Linear SDE

We consider a particular case of the general linear Stochastic Differential Equation (SDE):

$$dX(t) = [A \cdot X(t) + \tilde{u}(t)] + dW(t) \tag{3}$$

where $X : \mathbb{R} \to \mathbb{R}^n$ is a time-dependent state; $A \in \mathbb{R}^{n \times n}$ is a fixed dynamics operator; $\tilde{u} : \mathbb{R} \to \mathbb{R}^n$ is the control signal; and $dW : \mathbb{R} \to \mathbb{R}^n$ is a Brownian motion vector with covariance $Q \in \mathbb{R}^{n \times n}$.

General SDEs can be solved numerically using the first-order approximation $\Delta X(t) \approx \Delta t \cdot dX(t)$, or using more delicate approximations [Wang and Lin, 1998]. The linear SDE, however, and in particular Eq. (3), can be solved analytically [Herzog, 2013]:

$$X(t) = \Phi(t) \left(\Phi(t_0)^{-1} X(t_0) + \int_{t_0}^{t} \Phi(\tau)^{-1} \tilde{u}(\tau) d\tau + \int_{t_0}^{t} \Phi(\tau)^{-1} dW(\tau) \right) \tag{4}$$

where $X(t_0)$ is an initial condition, and $\Phi(t)$ is the eigenfunction of the system. More specifically, if V is the matrix whose columns $\{v_i\}_{i=1}^n$ are the eigenvectors of A, and Λ is the diagonal matrix whose diagonal contains the corresponding eigenvalues $\lambda = \{\lambda_i\}_{i=1}^n$, then

$$\Phi(t) = V e^{\Lambda t} = \begin{pmatrix} | & | & | & | & | \\ v_1 \cdot e^{\lambda_1 t} & \dots & v_i \cdot e^{\lambda_i t} & \dots & v_n \cdot e^{\lambda_n t} \\ | & | & | & | & | \end{pmatrix} \tag{5}$$

If the initial condition is given as $X(t_0) \sim N(\mu_0, \Sigma_0)$, Eq. (4) becomes

$$\begin{aligned} & X(t) \sim N\left(\mu(t), \Sigma(t)\right) \\ & \mu(t) = \Phi(t) \left(\Phi(t_0)^{-1} \mu_0 + \int_{t_0}^{t} \Phi(\tau)^{-1} \tilde{u}(\tau) d\tau \right), \ \Sigma(t) = \Phi(t) \Sigma'(t) \Phi(t)^\top \end{aligned} \tag{6}$$

where $\Sigma'(t) = \Phi(t_0)^{-1} \Sigma_0 (\Phi(t_0)^{-1})^\top + \int_{t_0}^{t} \Phi(\tau)^{-1} Q (\Phi(\tau)^{-1})^\top d\tau$.

Note that if $\forall i : \lambda_i < 0$ and $\tilde{u} \equiv 0$, we have $\mu(t) \xrightarrow{t \to \infty} 0$ (stable system). In addition, if λ is complex, Eq. (6) may produce a complex solution; Appendix F explains how to use a careful parameterization to only calculate the real solutions.

D Observation Filtering: The Conditional Distribution and the Relation to Kalman Filtering

As described in Section 2, the NESDE algorithm keeps an estimated Normal distribution of the system state $X(t)$ at any point of time. The distribution develops continuously through time according to the dynamics specified by Eq. (2), except for the discrete times where an observation $\hat{Y}(t)$ is received: in every such point of time, the $X(t)$ estimate is updated to be the conditional distribution $X(t)|\hat{Y}(t)$.

Calculating the conditional Normal distribution: The conditional distribution can be derived as follows. Recall that $X \sim N(\mu, \Sigma)$ (we remove the time index t as we focus now on filtering at a single point of time). Denote $X = (Y, Z)^\top$ where $Y \in \mathbb{R}^m$ and $Z \in \mathbb{R}^{n-m}$; and similarly, $\mu = (\mu_Y, \mu_Z)^\top$ and

$$\Sigma = \begin{pmatrix} \Sigma_{YY} & \Sigma_{YZ} \\ \Sigma_{ZY} & \Sigma_{ZZ} \end{pmatrix}$$

First consider a noiseless observation ($R = 0$): then according to Eaton [1983], the conditional distribution $X|Y = \hat{Y}$ is given by $X = (Y, Z)^\top$, $Y = \hat{Y}$ and $Z \sim N(\mu'_Z, \Sigma'_{ZZ})$, where

$$\mu'_Z := \mu_Z + \Sigma_{ZY} \Sigma_{YY}^{-1} (\hat{Y} - \mu_Y)$$

$$\Sigma'_{ZZ} := \Sigma_{ZZ} - \Sigma_{ZY} \Sigma_{YY}^{-1} \Sigma_{YZ}$$

In the general case of $R \neq 0$, we can redefine the state to include the observation explicitly: $\tilde{X} = (\hat{Y}, X)^\top = (\hat{Y}, Y, Z)^\top$, where $\tilde{\mu}, \tilde{\Sigma}$ of $\tilde{X}$ are adjusted by $\mu_{\hat{Y}} = \mu_y$, $\Sigma_{\hat{Y}\hat{Y}} = \Sigma_{YY} + R$,

$\Sigma_{\hat{Y}Y} = R$ and $\Sigma_{\hat{Y}Z} = \Sigma_{YZ}$. Then, the conditional distribution can be derived as in the noiseless case above, by simply considering the new observation as a noiseless observation of $\tilde{X}_{1:m} = \hat{Y}$.

The relation to the Kalman filtering: The derivation of the conditional distribution is equivalent to the filtering step of the Kalman filter [Kalman, 1960], where the (discrete) model is

$$\begin{aligned} X_{t+1} &= A \cdot X_t + \omega_t \qquad (\omega_t \sim N(0, Q)) \\ \hat{Y}_t &= H \cdot X_t + \nu_t \qquad (\nu_t \sim N(0, R)), \end{aligned}$$

Our setup can be recovered by substituting the following observation model $H \in \mathbb{R}^{m \times n}$, which observes the first m coordinates of X and ignores the rest:

$$H = \begin{pmatrix} 1 & & & & 0 & ... & 0 \\ & 1 & & & & & \\ & & ... & & | & | & | \\ & & & 1 & & & \\ & & & & 1 \; 0 & ... & 0 \end{pmatrix}$$

and the Kalman filtering step is then

$$\begin{aligned} K &:= \Sigma H^\top (H \Sigma H^\top + R)^{-1} \\ \mu' &:= \mu + K(\hat{Y} - H\mu) \\ \Sigma' &:= \Sigma - KH\Sigma \end{aligned}$$

Note that while the standard Kalman filter framework indeed supports the filtering of distributions upon arrival of a new observation, its progress through time is limited to discrete and constant time-steps (see the model above), whereas our SDE-based model can directly make predictions to any arbitrary future time t.

E Integrator Implementation

Below, we describe the implementation of the integrator of the Eigen-SDE solver mentioned in Section 2.

Numerical integration given $u(t)$: In the presence of an arbitrary (continuous) control signal $u(t)$, it is impossible to compute the integral that corresponds with $u(t)$ (Eq. (4)) analytically. On the other hand, $u(t)$ is given in advance, and the eigenfunction, $\Phi(t)$, is a known function that can be calculated efficiently at any given time. By discretizing the time to any fixed Δt, one could simply replace the integral by a sum term

$$\int_{t_0}^{t} \Phi(\tau)^{-1} u(\tau) d\tau \approx \sum_{i=0}^{\frac{t-t_0}{\Delta t}} \Phi(t_0 + i \cdot \Delta t) u(t_0 + i \cdot \Delta t) \Delta t$$

while this sum represent $\frac{t-t_0}{\Delta t}$ calculations, it can be computed efficiently, as it does not require any recursive computation, as both $\Phi(t)$ and $u(t)$ are pre-determined, known functions. Each element of the sum is independent of the other elements, and thus the computation could be parallelized.

Analytic integration: The control u is often constant over any single time-interval $\mathcal{I}$ (e.g., when the control is piecewise constant). In such cases, for a given interval $\mathcal{I} = [t_0, t]$ in which $u(t) = u_{\mathcal{I}}$, the integral could be solved analytically:

$$\int_{t_0}^{t} \Phi(\tau)^{-1} u(\tau) d\tau = \int_{t_0}^{t} e^{-\Lambda \tau} V^{-1} u_{\mathcal{I}} d\tau = \int_{t_0}^{t} e^{-\Lambda \tau} d\tau V^{-1} u_{\mathcal{I}} = \frac{1}{\Lambda} \big(e^{-\Lambda t_0} - e^{-\Lambda t}\big) V^{-1} u_{\mathcal{I}}$$

one might notice that for large time intervals this form is numerically unstable, to address this issue, note that this integral is multiplied (Eq. (4)) by $\Phi(t) = V e^{\Lambda t}$, hence we stabilize the solution with the latter exponent:

$$\Phi(t) \frac{1}{\Lambda} \big(e^{-\Lambda t_0} - e^{-\Lambda t}\big) V^{-1} u_{\mathcal{I}} = V \frac{1}{\Lambda} \big(e^{\Lambda(t - t_0)} - e^{\Lambda(t-t)}\big) V^{-1} u_{\mathcal{I}} = V \frac{1}{\Lambda} \big(e^{\Lambda(t-t_0)} - 1\big) V^{-1} u_{\mathcal{I}}$$

to achieve a numerically stable computation.

In addition to the integral over $u(t)$, we also need to calculate the integral over Q (Eq. (6)). In this case, Q is constant, and the following holds;

$$\int_{t_0}^{t} \Phi(\tau)^{-1}Q(\Phi(\tau)^{-1})^\top d\tau = \int_{t_0}^{t} e^{-\Lambda\tau}V^{-1}Q(V^{-1})^\top(e^{-\Lambda\tau})^\top d\tau = V^{-1}Q(V^{-1})^\top \circ \int_{t_0}^{t} e^{-\tilde{\Lambda}\tau}d\tau$$

where $\circ$ denotes the Hadamard product, and

$$\tilde{\Lambda} = \begin{pmatrix} 2\lambda_1 & \cdots & \lambda_1+\lambda_n \\ \vdots & \ddots & \\ \lambda_n+\lambda_1 & \cdots & 2\lambda_n \end{pmatrix}$$

In this form, it is possible to solve the integral analytically, similarly to the integral of the control signal, and again, we use the exponent term from $\Phi(t)$ to obtain a numerically stable computation.

F The Dynamics Spectrum and Complex Eigenfunction Implementation

The form of the eigenfunction matrix as presented in Appendix C is valid for real eigenvalues. Complex eigenvalues induce a slightly different form; firstly, they come in pairs, i.e., if $z = a + bi$ is an eigenvalue of A (Eq. (3)), then $\bar{z} = a - bi$ (the complex conjugate of z) is an eigenvalue of A. The corresponding eigenvector of z is complex as well, denote it by $v = v_{real} + v_{im}i$, then $\bar{v}$ (the complex conjugate of v) is the eigenvector that correspond to $\bar{z}$. Secondly, the eigenfunction matrix takes the form:

$$\Phi(t) = e^{at}\begin{pmatrix} | & | \\ v_{real}\cdot cos(bt) - v_{im}\cdot sin(bt) & v_{im}\cdot cos(bt) + v_{real}\cdot sin(bt) \\ | & | \end{pmatrix}$$

For brevity, we consider only the elements that correspond with z, $\bar{z}$. To parametrize this form, we use the same number of parameters (each complex number need two parameters to represent, but since they come in pairs with their conjugates we get the same overall number) which are organized differently. Mixed eigenvalues (e.g., both real and complex) induce a mixed eigenfunction that is a concatenation of the two forms. Since the complex case requires a different computation, we leave the number of complex eigenvalues to be a hyperparameter. Same as for the *real* eigenvalues setting, it is possible to derive an analytical computation for the integrals. Here, it takes a different form, as the complex eigenvalues introduce trigonometric functions to the eigenfunction matrix. To describe the analytical computation, first notice that:

$$\Phi(t) = e^{at}\begin{pmatrix} | & | \\ v_{real} & v_{im} \\ | & | \end{pmatrix}\begin{pmatrix} cos(bt) & sin(bt) \\ -sin(bt) & cos(bt) \end{pmatrix}$$

and thus:

$$\Phi(t)^{-1} = e^{-at}\begin{pmatrix} cos(bt) & -sin(bt) \\ sin(bt) & cos(bt) \end{pmatrix}\begin{pmatrix} | & | \\ v_{real} & v_{im} \\ | & | \end{pmatrix}^{-1}$$

Note that here we consider a two-dimensional SDE, for the general case the trigonometric matrix is a block-diagonal matrix, and the exponent becomes a diagonal matrix in which each element repeats twice. It is clear that similarly to the real eigenvalues case, the integral term that includes u (as shown above) can be decomposed, and it is possible to derive an analytical solution for an exponent multiplied by sine or cosine. One major difference is that here we use matrix product instead of Hadamard product. The integral over Q becomes more tricky, but it can be separated and computed as well, with the assistance of basic linear algebra (both are implemented in our code).

G Solver Analysis

Below, we provide a proposition for the optimality of Eigen-SDE solver.

Proposition 1 (Eigen-SDE solver optimality: complete formulation)**.** Let $X(t)$ be a signal that follows Eq. (2) for any time interval $\mathcal{I}_i = [t_i, t_{i+1}]$, and $u(t)$ a control signal that is constant over

$\mathcal{I}_i$ for any i. For any i, consider the Eigen-SDE solver with the parameters corresponding to Eq. (2) (for the same $\mathcal{I}_i$). Assume that the first solver ($i = 0$) is initialized with the true initial distribution $X(0) \sim N(\mu_0, \Sigma_0)$, and for $i \geq 1$, the i'th solver is initialized with the $i-1$'th output, along with an observation filter if an observation was received. For any interval i and any time $t \in \mathcal{I}_i$, consider the prediction $\tilde{X}(t) \sim N(\mu(t), \Sigma(t))$ of the solver. Then, $\mu(t)$ minimizes the expected square error of the signal $X(t)$, and $\tilde{X}(t)$ maximizes the expected log-likelihood of $X(t)$.

Proof. We prove by induction over i that for any i and any $t \in \mathcal{I}_i$, $\tilde{X}(t)$ corresponds to the true distribution of the signal $X(t)$.

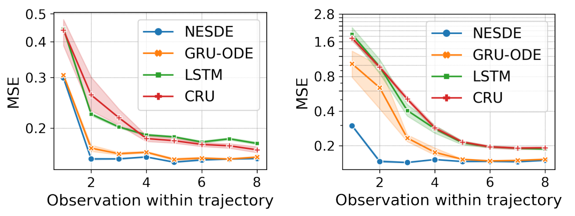
For $i = 0$, $X(t_i) = X(0)$ corresponds to the true initial distribution, and since there are no "interrupting" observations within $\mathcal{I}_0$, then the solution Eqs. (4) and (6) of Eq. (2) corresponds to the true distribution of $X(t)$ for any $t \in [t_i, t_{i+1})$. Since u is constant over $\mathcal{I}_0$, then the prediction $\tilde{X}(t)$ of the Eigen-SDE solver follows Eq. (6) accurately using the analytic integration (see Appendix E; note that if u were not constant, the solver would still follow the solution up to a numeric integration error). Regarding t_1, according to Appendix D, $\tilde{X}(t_1)$ corresponds to the true distribution of $X(t_1)$ *after* conditioning on the observation $\hat{Y}(t_1)$ (if there was an observation at t_1; otherwise, no filtering is needed). This completes the induction basis. Using the same arguments, if we assume for an arbitrary $i \geq 0$ that $\tilde{X}(t_i)$ corresponds to the true distribution, then $\tilde{X}(t)$ corresponds to the true distribution for any $t \in \mathcal{I}_i = [t_i, t_{i+1}]$, completing the induction.

Now, for any t, since $\tilde{X}(t) \sim N(\mu(t), \Sigma(t))$ is in fact the true distribution of $X(t)$, the expected square error $E[SE(t)] = E\left[(\mu - X(t))^2\right]$ is minimized by choosing $\mu := \mu(t)$; and the expected log-likelihood $E[\ell(t)] = E[\log P(X(t)|\mu, \Sigma)]$ is maximized by $\mu := \mu(t), \Sigma := \Sigma(t)$. $\square$

H Extended Experiments

H.1 Synthetic Data Experiments

In this section, we test three main aspects of NESDE: (1) prediction from partial and irregular observations, (2) robustness to out-of-distribution control (OOD), and (3) sample efficiency. We experiment with data of a simulated stochastic process, designed to mimic partially observable medical processes with indirect control.



(a) Same control distribution (b) Out of distribution control

Figure 4: MSE vs. number of observations so far in the trajectory, in the complex dynamics setting, for: (a) standard test set, and (b) test set with out-of-distribution control policy. 95% confidence intervals are calculated over 5 seeds.

The simulated data includes trajectories of a 1-dimensional signal Y, with noiseless measurements at random irregular times. The goal is to predict the future values of Y given its past observations. However, Y is mixed with a latent (unobservable) variable, and they follow linear dynamics with both decay and periodicity (i.e., complex dynamics eigenvalues). In addition, we observe a control signal that affects the latent variable (hence affects Y, but only indirectly through the dynamics). The control negatively correlated with the observations: $u_t = b_t - 0.5 \cdot Y_t$. $b_t \sim U[0, 0.5]$ is a piecewise constant additive noise (changing 10 times per trajectory).

Out-of-distribution control (OOD): We simulate two benchmarks – one with *complex* eigenvalues and another with *real* eigenvalues (no periodicity). We train all models on a dataset of 1000 random trajectories, and test on a separate dataset – with different trajectories that follow the *same distribution*. In addition, we use an *OOD* test dataset, where the control is positively correlated with the observations: $u_t = b_t + 0.5 \cdot Y_t$. This can simulate, for example, forecasting of the same biochemical process after changing the medicine dosage policy.

Table 2 and Fig. 4a summarize the prediction errors. Before changing the control policy, NESDE achieves the best accuracy in the complex dynamics, and is on par with GRU-ODE-Bayes in the real dynamics. Notice that CRU, which relies on a real-valued linear model in latent space, is indeed

Table 2: Test errors in the irregular synthetic benchmarks, estimated over 5 seeds and 1000 test trajectories per seed, with standard deviation calculated across seeds.

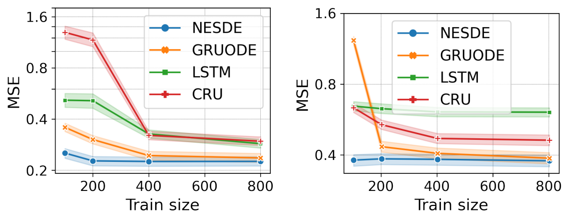
Model	Complex dynamics eigenvalues		Real dynamics eigenvalues	
	MSE	OOD MSE	MSE	OOD MSE
LSTM	0.23 ± 0.001	0.589 ± 0.02	0.381 ± 0.002	2.354 ± 0.84
GRU-ODE-Bayes	0.182 ± 0.0004	0.361 ± 0.044	$\mathbf{0.219 \pm 0.0004}$	0.355 ± 0.005
CRU	0.233 ± 0.0054	0.584 ± 0.009	0.231 ± 0.001	0.541 ± 0.026
NESDE (ours)	$\mathbf{0.176 \pm 0.0001}$	$\mathbf{0.178 \pm 0.001}$	0.222 ± 0.0005	$\mathbf{0.332 \pm 0.005}$

sub-optimal under the complex dynamics, compared to NESDE and GRU-ODE-Bayes. The LSTM presents high errors in both benchmarks.

Once the control changes, all models naturally deteriorate. Yet, NESDE presents the smallest deterioration and best accuracy in the OOD test datasets – for both complex and real dynamics. In particular, NESDE provides a high prediction accuracy after mere 2 observations (Fig. 4b), making it a useful zero-shot model. The robustness to the modified control policy can be attributed to the model of NESDE in Eq. (2), which decouples the control from the observations.

In a similar setting in Appendix H.7, the control u used in the training data has continuous knowledge of Y. Since the model only observes Y in a limited frequency, u carries additional information about Y. This results in extreme overfitting and poor generalization to different control policies – for all methods except for NESDE, which maintains robust OOD predictions in this challenging setting.

Sample efficiency: We train each method over datasets with different number of trajectories. Each model is trained on each dataset separately until convergence. As shown in Fig. 5, NESDE achieves the best test accuracy for every training dataset, and learns reliably even from as few as 100 trajectories. The other methods deteriorate significantly in the smaller datasets. Note that in the real dynamics, LSTM fails regardless of the amount of data, as also reflected in Table 2.



(a) Complex dynamics (b) Real dynamics

Figure 5: Test MSE vs. train data size. 95% confidence intervals are calculated over 1000 test trajectories.

GRU-ODE-Bayes achieves the best sample efficiency among the baselines. In Appendix H.3, we use **a benchmark from the study of GRU-ODE-Bayes itself** [De Brouwer et al., 2019], and demonstrate the superior sample efficiency of NESDE in that benchmark as well. Appendix H.4 extends the notion of sample efficiency to sparse trajectories: for a constant number of training trajectories, it reduces the number of observations per trajectory. NESDE demonstrates high robustness to the amount of data in that setting as well.

Regular LSTM: Appendix H.5 extends the experiments for regular data with constant time-steps. In the regular setting, LSTM provides competitive accuracy when observations are dense. However, LSTM fails if the signal is only observed once in multiple time-steps, possibly because gradients have to be propagated over many steps. Hence, even in regular settings, LSTM struggles to provide predictions more frequent than the measurements.

H.2 Ablation Study for Patient Individualization

To provide an insight over the importance of the dynamics-individualization, we perform an ablation study for the hypernetwork module. We use the same medical benchmarks as in Section 3, and fit a version of NESDE with neutralized hypernetwork module. In particular, we fix the context inputs of the module to be a vector of 1s, and thus prevent any propagation from the context features to the model's output. The results are presented in Table 3, and show a great degradation in model performance in the UH-dosing benchmark, approving that the hypernetwork indeed utilize the information within the context features. In the Vancomycin dosing benchmark, while we still observe a degradation comparing to NESDE, the version of NESDE without hypernetwork still outperforms LSTM in terms of MSE and the rest of the baselines (except NESDE) in terms of NLL.

Table 3: Test mean square errors (MSE) and negative log-likelihood (NLL) in the medication-dosing benchmarks. This is an extension of Table 1 with the additional results of NESDE without the hypernetwork.

Model	**UH Dosing**		**Vancomycin Dosing**	
	MSE	**NLL**	**MSE**	**NLL**
Naive	613.3 ± 13.48	–	112.2 ± 16.4	–
LSTM	482.1 ± 6.52	–	92.89 ± 11.3	–
GRU-ODE-Bayes	491 ± 6.88	4.52 ± 0.008	80.54 ± 11.8	6.38 ± 0.12
CRU	450.4 ± 8.27	4.49 ± 0.012	76.4 ± 12.8	3.87 ± 0.2
NESDE – no hypernet	529.7 ± 13.34	5.42 ± 0.067	87.32 ± 11.57	3.73 ± 0.13
NESDE (ours)	$\mathbf{411.2 \pm 7.39}$	$\mathbf{4.43 \pm 0.01}$	$\mathbf{70.71 \pm 12.3}$	$\mathbf{3.69 \pm 0.13}$

H.3 Comparison to ODE-based Methods

Appendix H.1 compares NESDE to GRU-ODE-Bayes [De Brouwer et al., 2019] – a recent ODE-based method that can provide an uncertainty estimation (which is a typical requirement in medical applications). Similarly to other recent ODE-based methods [Chen et al., 2018], GRU-ODE-Bayes relies on a non-linear neural network model for the differential equation. GRU-ODE-Bayes presents relatively poor prediction accuracy in Appendix H.1, which may be partially attributed to the benchmark settings. First, the benchmark required GRU-ODE-Bayes to handle a control signal. As proposed in De Brouwer et al. [2019], we incorporated the control as part of the observation space. However, such a control-observation mix raises time synchrony issues (e.g., most training input samples include only control signal without observation) and even affect the training supervision (since the new control dimension in the state space affects the loss). Second, as discussed above, the piecewise linear dynamics of NESDE provide higher sample efficiency in face of the 1000 training trajectories in Appendix H.1.

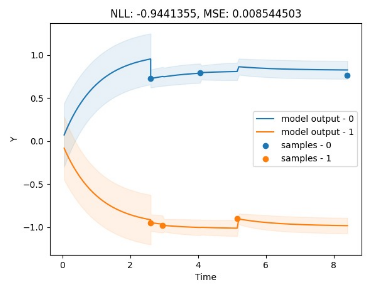


Figure 6: A sample test trajectory of the sparsely-observable OU process. The observations and the NESDE predictions (based on training over 400 trajectories) are presented separately for each of the two dimensions of the process.

In this section, we explicitly study the sample efficiency of NESDE vs. GRU-ODE-Bayes in a problem with no control signal. Specifically, we generate data from the GitHub repository of De Brouwer et al. [2019]. The data consists of irregular samples of the two-dimensional Ornstein-Uhlenbeck process, which follows the SDE

$$dx_t = \theta(\mu - x_t)dt + \sigma dWt,$$

where the noise follows a Wiener process, which is set in this experiment to have the covariance matrix

$$Cov = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix}.$$

The process is sparsely-observed: we use a sample rate of 0.6 (approximately 6 observations for 10 time units). Each sampled trajectory has a time support of 10 time units. The process has two dimensions, and each observation can include either of the dimensions or both of them. The dynamics of the process are linear and remain constant for all the trajectories; however, the stable "center" of the dynamics of each trajectory (similarly to α in Eq. (2)) is sampled from a uniform distribution, increasing the difficulty of the task and requiring to infer α in an online manner.

Fig. 6 presents a sample of trajectory observations along with the corresponding predictions of the NESDE model (trained over 400 trajectories). Similarly to De Brouwer et al. [2019], the models are tested over each trajectory by observing all the measurements from times $t \leq 4$, and then predicting the process at the times of the remaining observations until the end of the trajectory.

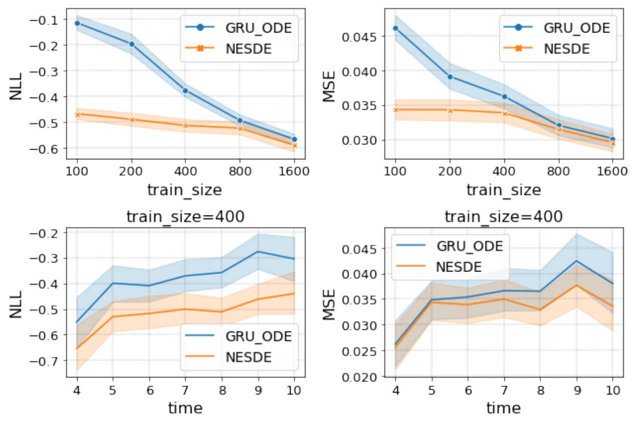
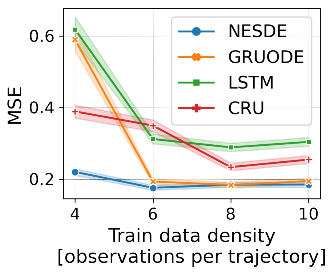


Figure 7: Top: losses of NESDE and GRU-ODE-Bayes over the OU benchmark, along with confidence intervals of 95% over the test trajectories. NESDE demonstrates higher data efficiency, as its deterioration in small training datasets is moderate in comparison to GRU-ODE-Bayes. Bottom: errors vs. time, given 400 training trajectories, where all the test predictions rely on observations from times $t \leq 4$. The advantage of NESDE becomes larger as the prediction horizon is longer.

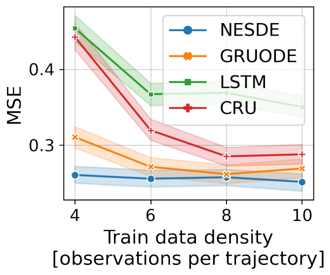
To test for data efficiency, we train both models over training datasets with different numbers of trajectories. As shown in Fig. 7, the sparsely-observable setting with limited training data causes GRU-ODE-Bayes to falter, whereas NESDE learns robustly in this scenario. The advantage of NESDE over GRU-ODE-Bayes increases when learning from smaller datasets (Fig. 7, top), or when predicting for longer horizons (Fig. 7, bottom). This demonstrates the stability and data efficiency of the piecewise linear dynamics model of NESDE in comparison to non-linear ODE models.

H.4 Sparse Observations

This experiment addresses the sparsity of each trajectory. We use the same benchmark as in Appendix H.1 and generate 4 train datasets, each one contains 400 trajectories, and a test set of 1000 trajectories. In each train-set, the trajectories have the same number of data samples, which varies between datasets (4,6,8,10). The test-set contains trajectories of varying number of observations, over the same support. For each train-set, we train all the models until convergence, and test them. Fig. 8 presents the MSE over the test set, for both the complex and the real eigenvalues settings. It is noticeable that even with very sparse observations, NESDE achieves good performance. Here, GRU-ODE-Bayes appears to be more sample-efficient than CRU and LSTM, but it is less sample efficient than NESDE.



(a) Complex dynamics



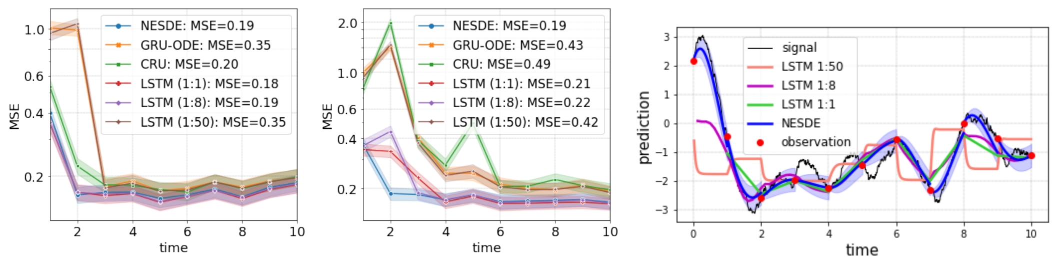
(b) Real dynamics

Figure 8: Test MSE vs. train observations-per-trajectory. 95% confidence intervals are calculated over 1000 test trajectories.

H.5 Synthetic Data Experiments with Regular Observations

While NESDE (and ODE-based models) can provide predictions at any point of time, a vanilla LSTM is limited to the predefined prediction horizon. Shorter horizons provide higher temporal resolution, but this comes with a cost: more recursive computations are needed per time interval, increasing both learning complexity and running time. For example, if medical measurements are available once per hour while predictions are required every 10 seconds, the model would have to run recursively 360 times between consecutive measurements, and would have to be trained accordingly in advance. We use the synthetic data environment from Appendix H.1, in the *complex* dynamics setting, and test both regularly and out-of-distribution control (see Appendix H.1). Here, we use LSTM models trained with resolutions of 1, 8 and 50 predictions per observation. All the LSTM models receive the control u and the current observation Y as an input, along with a boolean b_o specifying observability: in absence of observation, we set $Y = 0$ and $b_o = 0$. The models consist of a linear layer on top of an LSTM layer, with 32 neurons between the two. To compare LSTMs with various resolutions, we work with regular samples, 10 samples, one at each second. The control changes in a 10^{-2} seconds' resolution, and contains information about the true state.

In Fig. 9c we present a sample trajectory (without the control signal) with the predictions of the various LSTMs and NESDE. It can be observed that while NESDE provides continuous, smooth predictions, the resolution of the LSTMs must be adapted for a good performance. As shown in Fig. 9a, all the methods perform well from time $t = 3$ and on, still, NESDE and the low-resolution variants of LSTM attain the best results. The poor accuracy of the high-resolution LSTM demonstrates the accuracy-vs-resolution tradeoff in recursive models, moreover, GRUODE shows similar behavior in this analysis, which may hint on the recursive components within GRUODE.



(a) One-step prediction (b) Out of distribution control (c) Sample trajectory

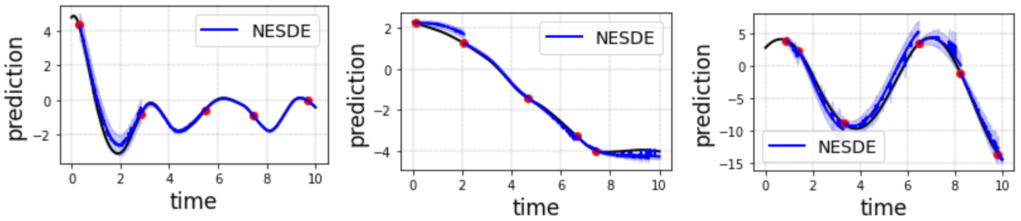
Figure 9: MSE for predictions, relying on the whole history of the trajectory for (a) the test set, and (b) out-of-distribution test set. The uncertainty corresponds to 0.95-confidence-intervals over 1000 trajectories. (c) Sample trajectory and predictions. The LSTM predictions are limited to predefined times (e.g., LSTM 1:1 only predicts at observation times), but their predictions are connected by lines for visibility. The shading corresponds to NESDE uncertainty (note that the LSTM does not provide uncertainty estimation).

The out-of-distribution test results (Fig. 9b) show that a change in the control policy could result with major errors; while NESDE achieves errors which are close to Fig. 9a, the other methods deteriorate in their performance. Notice the scale difference between the figures. The high-resolution LSTM and the ODE-based methods suffer the most, and the low-resolution variants of the LSTM, demonstrate robustness to the control change. This result is similar to the results we present in Appendix H.1, although here we see similarities between the variants of the LSTM and the ODE-based methods.

H.6 Interpretability: Inspecting the Spectrum

In addition to explicit predictions at flexible times, NESDE provides direct estimation of the process dynamics, carrying significant information about the essence of the process.

For example, consider the following 3 processes, each with one observable variable and one latent variable: $A_1 = \begin{pmatrix} -0.5 & -2 \\ 2 & -1 \end{pmatrix}$ with the corresponding eigenvalues $\lambda_1 \approx -0.75 \pm 1.98i$; $A_2 = \begin{pmatrix} -0.5 & -0.5 \\ -0.5 & -1 \end{pmatrix}$ with $\lambda_2 \approx (-1.3, -0.19)^\top$; and $A_3 = \begin{pmatrix} 1 & -2 \\ 2 & -1 \end{pmatrix}$ with $\lambda_3 \approx \pm 1.71i$. As demonstrated in Fig. 10, the three processes have substantially different dynamics: roughly speaking, real negative eigenvalues correspond to decay, whereas imaginary eigenvalues correspond to periodicity.



(a) Complex λ (b) Real λ (c) Imaginary λ

Figure 10: Sample trajectories with different types of dynamics (the control signal is not shown). In addition to the predictions, NESDE directly estimates the dynamics defined by λ.

For each process, we train NESDE over a dataset of 200 trajectories with 5-20 observations each. We set NESDE to assume an underlying dimension of $n = 2$ (i.e., one latent dimension in addition to the $m = 1$ observable variable); train it once in real mode (real eigenvalues) and once in complex mode (conjugate pairs of complex eigenvalues); and choose the model with the better NLL over the validation data. Note that instead of training twice, the required expressiveness could be obtained using $n = 4$ in complex mode (see Appendix F); however, in this section we keep $n = 2$ for the sake of spectrum interpretability.

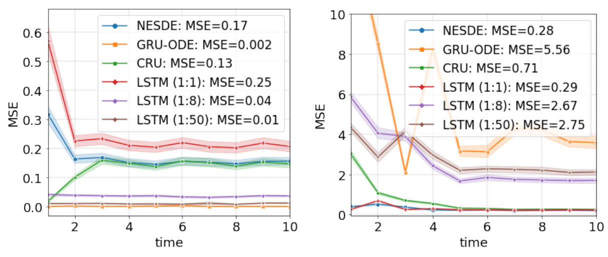
As the processes have linear dynamics, for each of them NESDE learned to predict a consistent dynamics model: all estimated eigenvalues are similar over different trajectories, with standard deviations smaller than 0.1. The learned eigenvalues for the three processes are $\tilde{\lambda}_1 = -0.77 \pm 1.98i$; $\tilde{\lambda}_2 = (-0.7, -0.19)^\top$; and $\tilde{\lambda}_3 = -0.03 \pm 0.83i$. That is, NESDE recovers the eigenvalues class (complex, real, or imaginary), which captures the essence of the dynamics – even though it only observes one of the two dimensions of the process. The eigenvalues are not always recovered with high accuracy, possibly due to the latent dimensions making the dynamics formulation ambiguous.

H.7 Model Expressiveness and Overfitting

It is well known that more complex models are capable to find complex connections within the data, but are also more likely to overfit the data. It is quite common that a data that involves control is biased or affected by confounding factors: a pilot may change his course of flight because he saw a storm that was off-the-radar; a physician could adapt his treatment according to some measure that is off-charts. Usually, using enough validation data could solve the overfitting issue, although sometimes the same confounding effects show in the validation data, which results in a model that is overfitted to the dataset. When targeting a model for control adjustment, it is important that it would be robust to changes in the control; a model that performs poorly when facing different control is unusable for control tuning. To exemplify an extreme case of confounding factors in the context of control, we add a correlation between the control (observed at all times) to the predictable measure (observed sparsely), in particular at times that the predictable is unobserved. We harness the same synthetic data benchmark as in Appendix H.1, and use regular time samples, and the same LSTM baselines as in Appendix H.5 but here we generate different two types of control signals:

1. Same Distribution (SD): at each time t, the control $u(t) = b_t - 0.8 \cdot Y_t$.
2. Out of Distribution(OOD): at each time t, the control $u(t) = b_t + 0.8 \cdot Y_t$.

b_t is a random piecewise constant and Y_t is the exact value of the measure we wish to predict. The first type is used to generate the train and the test sets, additionally we generate an out-of-distribution test-set using the second type. We observe in Fig. 11 that GRU-ODE-Bayes and the high-resolution LSTM achieve very low MSE over the SD as seen during training. CRU also achieves very low MSE, although not as much. The results over the OOD data show that the high performance over SD came with a cost – the better a model is over SD the worse it is over OOD. The results of LSTM 1:1 are not surprising, it sees the control signal only at observation-times, so it cannot exploit the hidden information within the control signal. NESDE does not ignore such information, while maintaining the robustness w.r.t. control.



(a) Same control distribution (b) Out of distribution control

Figure 11: MSE for predictions under regular time samples, where the control signal is correlated to the measure we wish to predict, even in times when it is unobserved. (a) Shows the results for a test set that has the same correlation between the control and the predictable measure as in the train set. (b) present the MSE for a different test set, with different correlation. Notice the different scales of the graphs.

I Medication Dosing Prediction: Implementation Details

Below, we elaborate on the implementation details of Section 3.

I.1 Data preprocessing

Heparin: We derive our data from the MIMIC-IV dataset [Johnson et al., 2020], available under the PhysioNet Credentialed Health Data License. For the UH dosing dataset, we extract the patients that were given UH during their intensive care unit (ICU) stay. We exclude patients that were treated with discrete (not continuous) doses of UH, or with other anticoagulants; or that were tested for aPTT less than two times. The control signal (UH dosing rate) is normalized by the patient weight. Each trajectory of measurements is set to begin one hour before the first UH dose, and is split in the case of 48 hours without UH admission. This process resulted with 5866 trajectories, containing a continuous UH signal, an irregularly-observed aPTT signal, and discretized context features. Note that we do not normalize the aPTT values.

Vancomycin: The VM dosing dataset derived similarly, from patients who received VM during their ICU stay, where we consider only patients with at least 2 VM concentration measurements. Each trajectory begins at the patient's admission time, and we also split in the case of 48 hours without VM dosage. Additionally, we add an artificial observation of 0 at time $t = 0$, as the VM concentration is 0 before any dose was given (we do not use these observations when computing the error).

General implementation details: For each train trajectory, we only sample some of the observations, to enforce longer and different prediction horizons, which was found to aid the training robustness. Hyperparameters (e.g., learning rate) were chosen by trial-and-error with respect to the validation-set (separately for each model).

Context variables C are used in both domains. We extract 42 features, some measured continuously (e.g., heart rate, blood pressure), some discrete (e.g., lab tests, weight) and some static (e.g., age, background diagnoses). Each feature is averaged (after removing outliers) over a fixed time-interval of four hours, and then normalized.

I.2 LSTM Baseline Implementation

The LSTM module we use as a baseline has been tailored specifically to the setting:

1. It includes an embedding unit for the context, which is updated whenever a context is observed, and an embedded context is stored for future use.
2. The inputs for the module include the embedded context, the previous observations, the control signal and the time difference between the current time and the next prediction time.
3. Where the control signal is piecewise constant: any time it changes we produce predictions (even though no sample is observed) that are then used as an input for the model, to model the effect of the UH more accurately.

We train it with the same methodology we use for NESDE where the training hyperparameters chosen by the best performance over the validation data.

Architecture for the medication dosing benchmarks: The model contains two fully connected elements: one for the context, with two hidden layers of size 32 and 16-dimensional output which is fed into a $Tanh$ activation; the second one uses the LSTM output to produce a one-dimensional output, which is fed into a ReLU activation to produce positive outputs, its size determined by the LSTM dimensions. The LSTM itself has an input of 19 dimensions; $16 + 1 + 1 + 1$ for the context, control, previous observations and the time interval to predict. It has a hidden size of 64 and two recurrent layers, with dropout of 0.2. All the interconnections between the linear layers include ReLU activations.

Architecture for the synthetic data benchmarks: Here, there is no context, then the model contains one fully connected element that receives the LSTM output and has two linear layers of sizes 32 and 1 with a Tanh activation between them. The LSTM has an input of 3 dimensions; for the state, control signal, and the time interval to predict. It has a hidden size of 32 and two recurrent layers, with dropout of 0.2.

I.3 Extended Results

The figures below present more detailed information for the experiments discussed in Section 3. All experiments were run on a single Ubuntu machine with eight i9-10900X CPU cores and Nvidia's RTX A5000 GPU. NESDE required several hours to train per benchmark.

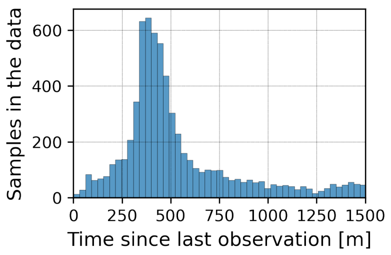


Figure 12: Histogram of prediction horizons in the UH dosing data (Section 3). Notice that the peak of the histogram around 6 hours (360 minutes) corresponds to the accuracy peak of the LSTM and GRU-ODE-Bayes in Fig. 3.