ODE-RSSM: Learning Stochastic Recurrent State Space Model from Irregularly Sampled Data

Zhaolin Yuan [1] , **Xiaojuan Ban** [1,2,3*] , **Zixuan Zhang** [1] , **Xiaorui Li** [1] , **Hong-Ning Dai** [4]

[1]School of Intelligence Science and Technology, Beijing Key Laboratory of Knowledge Engineering for Materials Science, University of Science and Technology Beijing, Beijing 100083, China.
[2]Beijing Advanced Innovation Center for Materials Genome Engineering, University of Science and Technology Beijing.
[3]Key Laboratory of Intelligent Bionic Unmanned Systems, Ministry of Education, University of Science and Technology Beijing, Beijing 100083, China
[4]Department of Computer Science, Hong Kong Baptist University, Hong Kong, China
18810919727@163.com, banxj@ustb.edu.cn, zhangzixuan1120@163.com, lixiaorui@xs.ustb.edu.cn, hndai@ieee.org

Abstract

For the complicated input-output systems with nonlinearity and stochasticity, Deep State Space Models (SSMs) are effective for identifying systems in the latent state space, which are of great significance for representation, forecasting, and planning in online scenarios. However, most SSMs are designed for discrete-time sequences and inapplicable when the observations are irregular in time. To solve the problem, we propose a novel continuous-time SSM named Ordinary Differential Equation Recurrent State Space Model (ODE-RSSM). ODE-RSSM incorporates an ordinary differential equation (ODE) network (ODE-Net) to model the continuous-time evolution of latent states between adjacent time points. Inspired from the equivalent linear transformation on integration limits, we propose an efficient reparameterization method for solving batched ODEs with non-uniform time spans in parallel for efficiently training the ODE-RSSM with irregularly sampled sequences. We also conduct extensive experiments to evaluate the proposed ODE-RSSM and the baselines on three input-output datasets, one of which is a rollout of a private industrial dataset with strong long-term delay and stochasticity. The results demonstrate that the ODE-RSSM achieves better performance than other baselines in open loop prediction even if the time spans of predicted points are uneven and the distribution of length is changeable. Code is availiable at https://github.com/yuanzhaolin/ODE-RSSM.

Introduction

Deep learning-based dynamical system modeling, also named deep system identification, exploits the advantages of deep neural networks (DNN) to identify the dynamics of black-box systems from offline input-output data. The learned model is generally utilized for prediction, model prediction control and model-based reinforcement learning (MBRL) (Moerland, Broekens, and Jonker 2020).

However, there are several common challenges in modeling real complicated systems. First, messy or irregular sampling data is ubiquitous in practical industrial applications (Kidger 2021). Most existing neural network-based system identification models are discrete-time models depending on the assumption of uniform sampling intervals. These models are restricted in real-time industrial system with irregularly sampling time points, especially in online scenarios where offline interpolations are incapable. Second, although some existing continuous-time system identification models, such as Time-Aware Recurrent Neural Network (RNN) (Demeester 2020), SNODE (Quaglino et al. 2020; Yildiz, Heinonen, and Lähdesmäki 2021) have tackled the learning problem with irregularly sampled input-output data, most of them only learn the system in the deterministic state space. Deterministic models are not only inconvenient for Monte-Carlo sampling, but are also incapable of learning the systems that suffer from high stochasticity. Finally, the identification model should be applicable to online mode when handling continuous streaming data and predicting the outputs with arbitrary prediction range (Liu et al. 2020). This requirement further restricts the employment of end-to-end encoder-decoder models, such as Latent ODE and ODE2VAE (Rubanova, Chen, and Duvenaud 2019; Yildiz, Heinonen, and Lähdesmäki 2019), while prefers choosing the models with recurrent state updating.

Although previous studies have partially solved some of the above issues, none of existing works can simultaneously address all of them, to the best of our knowledge. In this paper, we propose the Ordinary Differential Equation Recurrent State Space Model (ODE-RSSM), which is a stochastic transition model defined in continuous-time (CT) domain. The proposed ODE-RSSM incorporates an ODE-Net in the state evolution to learn the dynamical systems with stochasticity from irregularly sampled data. Under the setting of irregularly sampling, in order to support parallel prediction for accelerating the training of ODE-RSSM, a reparameterization method is proposed for solving the batched ODE-Nets with non-uniform time spans in parallel.

In experiments, we use three black-box system identification datasets to evaluate the performance of the proposed model in multi-step open-loop prediction (Monte Carlo roll-

*Corresponding author

Model	Stochasticity	Unevenly sampling	Online prediction
RNN	✗	✗	✓
RSSM	✓	✗	✓
Time-Aware RNN	✗	✓	✓
ODE-RNN	✗	✓	✓
Latent ODE	✓	✓	✗
ODE-RSSM	✓	✓	✓

Table 1: Comparison of neural network-based system modeling methods

outs). One of the three datasets is a private real-world dataset exported from an industrial paste thickening system. The strong stochasticity and the time-delay put forward higher requirements on models in learning long-term dependencies and predicting long-time-delayed outputs. The results on three datasets demonstrate that the ODE-RSSM outperforms other baselines when the time spans of predicted points are uneven and changeable. By examining the proposed reparameterization method and other competitors in solving batched ODEs with non-uniform time spans, we find that the proposed method can solve the batched ODEs in parallel under enough GPU memory.

The key contributions are highlighted as follows:

1. We propose ODE-RSSM to identify input-output systems with stochasticity from unevenly sampled sequences.
2. For improving the training speed under the irregular-sampling settings, we propose a reparameterization method to solve the batched ODE-Nets with non-uniform time spans in parallel.
3. We provide a comprehensive and systematic evaluation of the proposed model on three input/output datasets and demonstrate the effectiveness of the proposed method.

Related Work

Learning system dynamics, also known as system identification (Åström and Eykhoff 1971), is a task that learns the dynamics of an input-output system from offline data. It plays an important role in diverse areas, such as system prediction and model-based reinforcement learning (Ke et al. 2019). This paper focuses on employing approximated parametric neural networks (Wang 2017; Ogunmolu et al. 2016) to model black-box systems where both of the functional forms and parameters are unknown.

In many real-world systems, the sequential data are unevenly sampled from the identified system. Spectral Discretization of Neural ODEs (SNODE) (Quaglino et al. 2020) introduces the spectral element method and gradient matching for improving the training speed and accuracy of learned Neural ODEs in system identification. Moreover, ODE-Recurrent Neural Network (ODE-RNN) (Rubanova, Chen, and Duvenaud 2019) inserts an ODE-Net module between adjacent RNN updates for modeling the continuous-time evolution of hidden states. In order to improve the long-term prediction, previous studies such as (Demeester 2020; Yuan et al. 2022) incorporate the advantages of recurrent neural networks and differential equations to tackle the unit root problem and propose the Time-Aware RNN. The above-mentioned models are capable of tackling the problem of unevenly sampling though they still have limitations in modeling stochastic systems because their internal states evolution is essentially deterministic.

In the perspective of modeling stochastic system, probabilistic recurrent state-space models (PR-SSM) (Doerr et al. 2018) and probabilistic inference for learning control (PILCO) (Deisenroth and Rasmussen 2011) introduce non-parametric Gaussian Processes to learn the probabilistic transition models of input/output systems. As parametric models, deep temporal generative models (Fraccaro et al. 2016; Chung et al. 2015; Karl et al. 2017) extend the Variational Auto-encoders (VAEs) (Kingma and Welling 2014) to sequential situations. These models are capable of learning the stochasticity in the sequences by introducing stochastic latent variables. Recurrent state space model (RSSM)(Hafner et al. 2019) mixes deterministic state evolution and stochastic state evolution in the latent state space for identifying stochastic systems. With the benefits of efficient Monte Carlo Sampling, the learned generative models are employed in many applications, such as missing data imputation (Fraccaro et al. 2017), open-loop prediction (Hafner et al. 2019) and model-based reinforcement learning (RL) (Hafner et al. 2019).

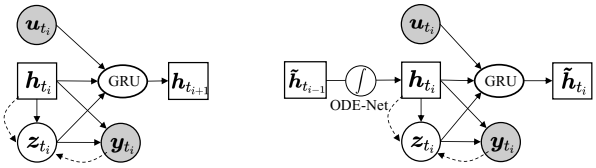
Some previous studies also integrate the temporal variational inference framework and differential equation networks to learn a stochastic continuous-time process. Specifically, LatentODE (Rubanova, Chen, and Duvenaud 2019), ODE2VAE (Yildiz, Heinonen, and Lähdesmäki 2019) infer the posterior distribution of an initial latent state from the irregular observed sequences. With a sampled initial state, models can interpolate the missing positions and predict the system outputs by solving the differential equation networks. However, these models are typically offline since they have to read the complete irregular time series for encoding before prediction and interpolation (Liu et al. 2020). In some specific tasks, such as online control, the prediction model is required to be working in an online mode for handling streaming data (Lesort et al. 2018), similar to the recurrent neural network, which recursively updates internal states.

In this paper, we propose a novel ODE-RSSM model, which satisfies all the above requirements in practical industrial applications, such as learning from irregular sampled data, modeling stochasticity, and online prediction. Table 1 compares the proposed model with other neural network-based modeling methods.

Model

Preliminary and notations

For an input-output system, we define the irregularly sampled system trajectories as $\{\boldsymbol{u}_{k,t_1:t_{N_k}}, \boldsymbol{y}_{k,t_1:t_{N_k}}\}_{k=1}^{K}$, where K is the number of sequences in the dataset, N_k is the length of the k-th sequence, and the time subscript t_i ($i \in [1, 2, \cdots, N_k]$) is the sampling time of the i-th position. Omitting the subscript k, the conditional generation of $\boldsymbol{y}_{t_1:t_N}$ under $\boldsymbol{u}_{t_1:t_N}$ is



(a) RSSM (b) ODE-RSSM

Figure 1: The overall structure of both generative process and inference process in ODE-RSSM. Circles and squares are stochastic variables and deterministic variables, respectively. Solid lines and dashed lines denote the generative processes and the inference processes, respectively.

assumed to be a Markov model with latent states $\boldsymbol{s}_{t_i}$:

$$p(\boldsymbol{y}_{t_1:t_N}|\boldsymbol{u}_{t_1:t_N}) = \int p\left(\boldsymbol{y}_{t_1:t_N}, \boldsymbol{s}_{t_1:t_N} \mid \boldsymbol{u}_{t_1:t_N}, \boldsymbol{s}_{t_0}\right) \mathrm{d}\boldsymbol{s}_{t_1:t_N} = \int \prod_{i=1}^{N} p\left(\boldsymbol{y}_{t_i} \mid \boldsymbol{s}_{t_i}\right) p\left(\boldsymbol{s}_{t_i} \mid \boldsymbol{u}_{t_{i-1}}, \boldsymbol{s}_{t_{i-1}}\right) \mathrm{d}\boldsymbol{s}_{t_1:t_N}. \tag{1}$$

We aim to identify a deep temporal VAE framework consisting of (i) an inference model for inferring the sequential latent states $\{\boldsymbol{s}_{t_i}\}_{i=1}^N$ given both system inputs and outputs, and (ii) a generative model for predicting the distribution of $\{\boldsymbol{y}_{t_i}\}_{i=1}^N$ given system inputs $\{\boldsymbol{u}_{t_i}\}_{i=1}^N$.

Generative Model

As a conditional probabilistic model, the generative process describes the joint probability distributions of latent states and system outputs under given system inputs. Similar to the RSSM model (Hafner et al. 2019), we define the latent states as $\boldsymbol{s}_{t_i} = [\boldsymbol{h}_{t_i}, \boldsymbol{z}_{t_i}]$ consisting of both deterministic component $\boldsymbol{h}_{t_i}$ and stochastic component $\boldsymbol{z}_{t_i}$. Given an initial $\boldsymbol{s}_{t_1}$, the conditional prior $\boldsymbol{s}_{t_i}$ is controlled by system inputs:

$$p\left(\boldsymbol{s}_{t_i} \mid \boldsymbol{s}_{t_{i-1}}, \boldsymbol{u}_{t_{i-1}}\right) = p\left(\boldsymbol{z}_{t_i} \mid \boldsymbol{h}_{t_i}\right) p\left(\boldsymbol{h}_{t_i} \mid \boldsymbol{s}_{t_{i-1}}, \boldsymbol{u}_{t_{i-1}}\right), \tag{2}$$

where $p\left(\boldsymbol{h}_{t_i} \mid \boldsymbol{s}_{t_{i-1}}, \boldsymbol{u}_{t_{i-1}}\right)$ is a Dirac distribution. Since the sampling interval $t_i - t_{i-1}$ for different positions i are not uniform in irregular settings, the common discrete-time sequential models are not applicable to modeling the evolution of latent states. Inspired by the differential equation models (Chen et al. 2018; Rubanova, Chen, and Duvenaud 2019), we aim to learn the CT evolution of deterministic latent states.

Concretely, the evolution of $\boldsymbol{h}_{t_i}$ between two adjacent time points consists of two stages: In the first stage, we employ a GRU network to model the influences from the system input $\boldsymbol{u}_{t_{i-1}}$ and $\boldsymbol{z}_{t_{i-1}}$ on the last state $\boldsymbol{h}_{t_{i-1}}$ and produce an intermediate variable $\tilde{\boldsymbol{h}}_{t_{i-1}}$ as follows:

$$\tilde{\boldsymbol{h}}_{t_{i-1}} = \mathrm{GRU}(\left[\boldsymbol{u}_{t_{i-1}}, \boldsymbol{z}_{t_{i-1}}\right], \boldsymbol{h}_{t_{i-1}}). \tag{3}$$

Next, we introduce a parametric ODE-Net f_θ to model the CT evolution of $\boldsymbol{h}_{t_i}$ and the updated state at the next time point can be predicted by solving the ODE:

$$\boldsymbol{h}_{t_i} = \mathrm{ODESolver}(f_\theta, \tilde{\boldsymbol{h}}_{t_{i-1}}, t_{i-1}, t_i) = \tilde{\boldsymbol{h}}_{t_{i-1}} + \int_{t_{i-1}}^{t_i} f_\theta(\boldsymbol{h}_t, \boldsymbol{u}_{t_{i-1}})\mathrm{d}t, \tag{4}$$

where the system input $\boldsymbol{u}_{t_{i-1}}$ is regarded as a piecewise constant signal, which keeps unchanged until $\boldsymbol{u}_{t_i}$ is given. This setting is consistent with the online scenarios. With the solved $\boldsymbol{h}_{t_i}$, the prior Gaussian distribution of $\boldsymbol{z}_{t_i}$ is predicted as follows:

$$p_\theta\left(\boldsymbol{z}_{t_i} \mid \boldsymbol{h}_{t_i}\right) = \mathcal{N}(\boldsymbol{z}_{t_i} \mid \boldsymbol{\mu}_{t_i}^{\text{prior}}, \boldsymbol{\sigma}_{t_i}^{\text{prior}}), \tag{5}$$

where $\boldsymbol{\mu}_{t_i}^{\text{prior}}$ and $\boldsymbol{\sigma}_{t_i}^{\text{prior}}$ are estimated from a Multilayer perceptron (MLP) with deterministic latent states $\boldsymbol{h}_{t_i}$ as input.

An intuitive explanation of the designed generative model is that the equation (4) tackles the problem of uneven time spans and predicts the latent state $\boldsymbol{h}_{t_i}$ at time t_i before perceiving the information brought from $\boldsymbol{u}_{t_i}$. The equation (3) models the stochasticity by incorporating the stochastic information from $\boldsymbol{z}_{t_i}$ and producing the new latent state $\tilde{\boldsymbol{h}}_{t_i}$. The ODE-RSSM is analogous to the combination of ODE-RNN and RSSM, which is a sophisticated and rational choice for modeling stochastic systems under irregular setting.

A decoder module is introduced to determine the predicted Gaussian distribution of system outputs $\boldsymbol{y}_{t_i}$ under specific latent states:

$$p\left(\boldsymbol{y}_{t_i} \mid \boldsymbol{h}_{t_i}, \boldsymbol{z}_{t_i}\right) = \mathcal{N}(\boldsymbol{y}_{t_i} \mid \boldsymbol{\mu}_{t_i}^{\text{dec}}, \boldsymbol{\sigma}_{t_i}^{\text{dec}}), \tag{6}$$

where the distribution parameters $\boldsymbol{\mu}_{t_i}^{\text{dec}}, \boldsymbol{\sigma}_{t_i}^{\text{dec}}$ depend on both deterministic and stochastic latent states.

Given sequential system inputs, the sequential latent states can be predicted by repeatedly sampling the states $\boldsymbol{s}_{t_i}$ from the predicted distribution of single step and feeding the samples into the generative model. We can further predict the open loop system outputs by feeding the sampled latent states into the decoder module.

Inferencing and Learning

In order to train the parameters θ in the generative model, we need to maximize the log likelihood of system outputs $\mathcal{L} = \sum_{i=1}^N \log p_\theta\left(\boldsymbol{y}_{t_1:t_N} \mid \boldsymbol{u}_{t_1:t_N}\right)$. A regular way is to introduce a variational distribution q_ϕ to infer the approximate posterior distribution of the stochastic latent state as follows:

$$q_\phi(\boldsymbol{z}_{t_1:t_N} \mid \boldsymbol{y}_{t_1:t_N}, \boldsymbol{u}_{t_1:t_N}) = \prod_{i=1}^{N} q_\phi(\boldsymbol{z}_{t_i} \mid \boldsymbol{h}_{t_i}, \boldsymbol{y}_{t_i}) = \prod_{i=1}^{N} \mathcal{N}(\boldsymbol{z}_{t_i} \mid \boldsymbol{\mu}_{t_i}^{\text{enc}}, \boldsymbol{\sigma}_{t_i}^{\text{enc}}), \tag{7}$$

where the variational distribution $q_\phi(\boldsymbol{z}_{t_i}|\cdot)$ is a Gaussian distribution whose parameters $\boldsymbol{\mu}_{t_i}^{\text{enc}}$ and $\boldsymbol{\sigma}_{t_i}^{\text{enc}}$ are estimated from the encoder module, built as an MLP. Since solving $\boldsymbol{h}_{t_i}$ by (3) and (4) requires the previous stochastic latent states $\boldsymbol{z}_{t_{i-1}}$, the generative process and inference process are alternating for inferring the sequential posterior $q_\phi(\boldsymbol{z}_{t_1:t_N})$. Ideally, the predicted prior distribution $p(\boldsymbol{z}_{t_i}|\boldsymbol{h}_{t_i})$ in (5) has more entropy than $q(\boldsymbol{z}_{t_i})$ in (7) because $\boldsymbol{y}_{t_i}$ is unknown. The prior p is enforced to be as close to q as possible by minimizing KL divergence. Figure 1 illustrates the differences between the ODE-RSSM and RSSM in terms of the structures of generative model and inference model.

In order to improve multi-step predictions, we introduce the latent overshooting technique (Hafner et al. 2019) in

training, through which the parameters ϕ and θ are trained to maximize the multi-step evidence lower bound (ELBO) of the observed system outputs:

$$\ln p_d\left(\boldsymbol{y}_{t_1:t_N}\right) \geq \mathrm{E}_{q(\boldsymbol{s}_{t_1:t_N})}\left[\ln p\left(\boldsymbol{y}_{t_1:t_N} \mid \boldsymbol{s}_{t_1:t_N}\right)\right] - \frac{1}{D} * \text{KL-D} \tag{8}$$

where KL-D is given by

$$\begin{aligned}\text{KL-D} &= \sum_{i=1}^{N}\sum_{d=1}^{D} \mathrm{E}_{\substack{p\left(\boldsymbol{s}_{t_{i+d}} \mid \boldsymbol{s}_{t_i}\right) \\ q\left(\boldsymbol{s}_{t_i}\right)}}\left[\mathrm{KL}\left[q\left(\boldsymbol{s}_{t_{i+d}}\right) \| p\left(\boldsymbol{s}_{t_{i+d}} \mid \boldsymbol{s}_{t_{i+d-1}}\right)\right]\right] \\ &= \sum_{i=1}^{N} \mathrm{E}_{\hat{\boldsymbol{s}}_{t_i}\sim q(\boldsymbol{s}_{t_i})} \mathrm{E}_{p(\hat{\boldsymbol{s}}_{t_{i+1}:t_{i+D-1}}|\hat{\boldsymbol{s}}_{t_i})} \sum_{d=1}^{D} \mathrm{kl}(i+d),\end{aligned} \tag{9}$$

where $\mathrm{kl}(i+d)$ is the abbreviation of $\mathrm{KL}\left[q\left(\boldsymbol{s}_{t_{i+d}}\right) \| p\left(\boldsymbol{s}_{t_{i+d}} \mid \hat{\boldsymbol{s}}_{t_{i+d-1}}\right)\right]$, $\hat{\boldsymbol{s}}_{t_i}$ is sampled from the approximate posterior distribution, and $\hat{\boldsymbol{s}}_{t_{i+1}:t_{i+D-1}}$ are sampled by ancestral sampling, i.e., iteratively sampling the states from one-step predicted distribution and feeding the samples as inputs for predicting a new one. Figure 2 illustrates the process of sampling states and calculating the KL divergence. Because the predictive distribution $p(\cdot)$ implicitly contains GRU and ODE-Net, all parameters are trained end-to-end by maximizing ELBO.

Besides the approximation of the posterior distribution for training generative model, the inference model is also a temporal encoder model for representation. As an recurrent encoder-decoder framework, the ODE-RSSM consists of an encoder for inference and a generative model for prediction. Both these components are iteratively invoked in some tasks with online streaming data. For example, in the model-prediction control, the inference model is utilized to infer the latent states given monitored system outputs and the generative model predicts the system outputs under optimized control inputs.

Solving Batched ODEs with Non-uniform Time Spans

Networks are typically trained in batches to take advantage of efficient data-parallel GPU operations. For the multi-step predictions $p(\hat{\boldsymbol{s}}_{t_{i+1}:t_{i+D-1}} \mid \hat{\boldsymbol{s}}_{t_i})$ in all positions $1 \leq i \leq N$, it is difficult to solve the ODEs at all positions in parallel because their integrating limits are non-uniform when the sequence is unevenly sampled. Most existing differential-equation libraries assume that the solved time points are identical for all the sequences in the given batch and do not support batching over different regions of integration (Kidger 2021). In order to synchronously solve all ODEs in a minibatch, previous time series models based on ODE-nets (Yildiz, Heinonen, and Lähdesmäki 2021; Rubanova, Chen, and Duvenaud 2019) solve the solutions at the union of all the time points in the batch; this process is still time-consuming in practical applications.

To address this issue, we reparameterize the original ODE-Net to accelerate the solving process of the batched ODEs in the training phase. From a general perspective, we assume that the ODE-Net is solved from an initial batched states $\boldsymbol{H_T} = [\boldsymbol{h}^1_{t_1}, \cdots, \boldsymbol{h}^I_{t_I}]$ at batched time points $\boldsymbol{T} = [t_1, \cdots, t_I]$ to a terminal states $\boldsymbol{H_{T'}}$ at batched time points $\boldsymbol{T'} = [t'_1, \cdots, t'_I]$, where I is the size of batched states. For example, when we solve the multi-step predictions stated in (9), the terminal state $\boldsymbol{H_{T'}}$ can be solved by concatenating the solutions of I ODEs, each of which has different initial states and integrating timespans:

$$\boldsymbol{H_{T'}} = \begin{bmatrix} \text{ODESolver}(f_\theta, \boldsymbol{h}^1_{t_1}, t_1, t'_1) = \boldsymbol{h}_{t_1} + \int_{t_1}^{t'_1} f_\theta(\boldsymbol{h}^1_t)\mathrm{d}t \\ \cdots \\ \text{ODESolver}(f_\theta, \boldsymbol{h}^I_{t_I}, t_I, t'_I) = \boldsymbol{h}_{t_I} + \int_{t_I}^{t'_I} f_\theta(\boldsymbol{h}^I_t)\mathrm{d}t \end{bmatrix}. \tag{10}$$

To solve I ODEs in parallel, we give a theorem to describe the property of linear transformation on integration limits.

Theorem 1 ***The invariance of linearly changing the integration limits.*** *For an arbitrary scalar function $f(\cdot)$, whose integration defined on limits $[a, b]$ satisfies with:*

$$\int_a^b f(t)\mathrm{d}t = \int_0^1 f(\tau(b-a)+a)(b-a)\mathrm{d}\tau. \tag{11}$$

This theorem can be easily proven by substitution of variables with $\tau = \frac{t-a}{b-a}$(Wikipedia 2022). It inspires us to reparameterize the ODEs for normalizing the I time spans as an identical integration limit. Based on the original ODEs, we define an auxiliary initial value problem (IVP) given the derivative of $\boldsymbol{R}(\tau)$ with respect to the scalar time step τ and the initial state $\boldsymbol{R}(0)$:

$$\begin{cases} \dfrac{\mathrm{d}\boldsymbol{R}(\tau)}{\mathrm{d}\tau} = g_\theta(\boldsymbol{R}(\tau), \tau) = f_\theta(\boldsymbol{R}(\tau)) \circ (\boldsymbol{T'} - \boldsymbol{T}), \\ \boldsymbol{R}(0) = \boldsymbol{H_T}, \end{cases} \tag{12}$$

where $\circ$ is the point-wise production. The solution of the auxiliary IVP at $\tau = 1$ is equal to the concatenated solutions of the original I ODEs as follows,

$$\boldsymbol{H_{T'}} = \boldsymbol{R}(1) = \text{ODESolver}(g_\theta, \boldsymbol{R}(0), 0, 1). \tag{13}$$

The equation reveals that the regions of integration in batched ODEs are changeable. It is easy to solve the auxiliary IVP in parallel because of the uniform timespan $[0, 1]$ in the batch. Thereafter, we can efficiently solve the original batched ODEs.

Experimental Results

In this section, we evaluate the ODE-RSSM and representative baselines including discrete-time and continuous-time models on three input/output datasets. We investigate three research questions:

- **RQ1**: Does the ODE-RSSM outperform the existing discrete-time and continuous-time state space models on identifying unevenly sampled systems?
- **RQ2**: Does the ODE-RSSM generalize enough for the prediction tasks where the distribution of time spans is different from that of the training dataset.
- **RQ3**: Does the proposed reparameterization method efficiently solve the batched ODEs with non-uniform time spans?

(a) CSTR dataset

	25% (uneven)		25% (even)		50% (uneven)		50% (even)		100%	
	RRSE	RMSE	RRSE	RMSE	RRSE	RMSE	RRSE	RMSE	RRSE	RMSE
VAE-RNN	0.3118	0.2224	0.203	0.1407	0.1977	0.1372	0.1791	0.1238	0.1607	0.1099
STORN	0.3198	0.2235	0.1993	0.1379	0.2372	0.1664	0.1628	0.1115	0.155	0.1054
RSSM	0.3155	0.2158	0.1457	0.1026	0.2017	0.1447	0.0811	0.0594	0.0684	0.0499
RSSM-O	0.3114	0.2127	0.1507	0.1103	0.151	0.11	0.0872	0.0648	0.0797	0.0596
ODE-RNN	0.2627*	**0.1791**	0.14*	0.0996*	0.1466*	0.1047*	0.0784*	0.057*	0.0668*	0.0489*
Time-Aware	0.3134	0.2138	0.1581	0.1131	0.1965	0.1411	0.136	0.0991	0.108	0.0786
Latent-ODE	**0.2595**	0.1797*	0.1982	0.1407	0.1672	0.1194	0.1591	0.1149	0.1427	0.1016
Latent-SDE	0.3019	0.2046	0.1587	0.1127	0.1633	0.1171	0.0971	0.071	0.0827	0.0604
ODE-RSSM	0.2979	0.1987	**0.1376**	**0.0975**	0.1486	0.1059	0.0798	0.0593	0.0739	0.0517
ODE-RSSM-O	0.2807	0.1913	0.1411	0.1003	**0.1336**	**0.0956**	**0.0654**	**0.0477**	**0.0659**	**0.0474**

(b) Winding dataset

	25% (uneven)		25% (even)		50% (uneven)		50% (even)		100%	
	RRSE	RMSE	RRSE	RMSE	RRSE	RMSE	RRSE	RMSE	RRSE	RMSE
VAE-RNN	0.5242	0.5186	0.4276	0.4346	0.4708	0.4706	0.4521	0.4589	0.4192	0.4167
STORN	0.5282	0.5251	0.4151	0.4231	0.4799	0.4786	0.4058	0.4111	0.384	0.3807
RSSM	0.5667	0.5593	0.4155	0.4234	0.4932	0.4918	0.4013	0.4066	0.4011	0.3976
RSSM-O	0.5366	0.5313	0.4124	0.4202	0.5236	0.523	0.4098	0.415	0.3911	0.3876
ODE-RNN	0.5018*	0.4953*	**0.402**	**0.4079**	0.4247*	0.4208*	0.3852	0.3891	0.3808	0.3751
Time-Aware	0.6009	0.5961	0.4653	0.4748	0.4615	0.4613	0.431	0.4369	0.4017	0.398
Latent-ODE	0.5314	0.5224	0.4716	0.4796	0.4779	0.4725	0.4558	0.4629	0.4515	0.4511
Latent-SDE	0.6610	0.6561	0.4674	0.4771	0.4577	0.4562	0.3690*	**0.3723**	**0.3287**	**0.3248**
ODE-RSSM	0.5389	0.5327	0.4648	0.4738	0.4444	0.4429	0.3837	0.3880	0.3683	0.3633
ODE-RSSM-O	**0.4768**	**0.4709**	0.4169*	0.4244*	**0.4045**	**0.4029**	**0.3689**	0.3726*	0.3324*	0.3283*

(c) Thickening dataset

	25% (uneven)		25% (even)		50% (uneven)		50% (even)		100%	
	RRSE	RMSE	RRSE	RMSE	RRSE	RMSE	RRSE	RMSE	RRSE	RMSE
VAE-RNN	16.255	0.4696	14.471	0.4589	14.779	0.4361	13.915	0.4319	13.163	0.4065
STORN	11.336	0.3249	8.6047	0.2765	7.9986	0.2463	7.7936	0.2473	6.8105	0.2184
RSSM	1.619	0.0553	1.5165	0.0561	1.5808	0.0534	1.439	0.0532	1.5001	0.0548
RSSM-O	1.3725*	0.0529	1.375	0.0517	1.4275	0.0519	1.4214	0.0518	1.4153	0.0522
ODE-RNN	1.4789	0.0507*	**1.2927**	0.0494*	1.6023	0.0566	1.3138*	**0.0489**	**1.3193**	**0.0487**
Time-Aware	3.302	0.11	1.7698	0.0636	1.4732	0.0531	1.3882	0.0509	1.4108	0.0516
Latent-ODE	6.8852	0.225	6.7402	0.228	4.7758	0.1479	4.7416	0.1546	3.3587	0.1157
Latent-SDE	4.3114	0.1441	3.2254	0.1116	2.4136	0.0823	2.1839	0.0763	2.1006	0.0734
ODE-RSSM	1.5349	0.0558	1.4791	0.0549	1.4174*	0.0522*	1.4625	0.0544	1.5856	0.0577
ODE-RSSM-O	**1.3678**	**0.0504**	1.2947*	**0.0493**	**1.3064**	**0.0487**	**1.3082**	0.0493*	1.3302*	0.05*

Table 2: Main results on three Input/output datasets. In each experimental group, the best results are bolded and the second-best results are marked with '*'. The models with suffix '-O' are trained with latent overshooting.

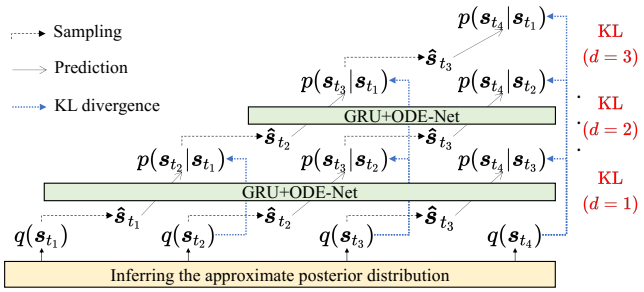


Figure 2: The process of solving KL-D in latent overshooting with $k = 4$ and $D = 3$.

Datasets

We use three input/output datasets for conducting the experiments. Two of them, *CSTR* (one in two out) and *Winding* (five in two out), are public datasets (Demeester 2020). The third one is the rollout of the provided paste thickening dataset (four in one out), which is derived from a realistic running paste thickener. This paste thickener is manufactured by the FLSmidth company and utilized in a copper mining paste backfilling station for producing high concentrated slurry. There are four main monitoring items in the thickening system: feed concentration, feed flow rate, underflow concentration and underflow rate are defined as the system inputs for predicting the mud pressure, which is the single system output. Because the thickening system is incompletely observed and the feedback delay of controlling is extremely long, this dataset raises higher requirements on representing stochastic dynamics and predicting long-time delay system.

Experimental Setup

Each dataset is split to three partitions for training (the foremost 60%), validation (the middle 20%), and test (the last 20%). We further move a sliding window on each partition to generate data batches, each of which is a tensor with the shape $B \times N \times K$, where B is the batch size, N denotes the length of the sequence, and K is the sum of the dimensions of both system inputs and system outputs.

In the training phase, the sequence with length N is totally fed to the model and the multi-step ELBO is maximized. In the phases for validation and test, each sequence is further split into two sequences: an encoding sequence with length M and a generative sequence with length L. The encoding sequence including the system inputs and outputs is fed into the inference encoder to infer latent state $\boldsymbol{s}_{t_M}$ at the M-th position. Next, as an initial latent state, state $\boldsymbol{s}_{t_M}$ is fed to the generative model to predict system outputs given sequential system inputs in the generative sequence with length L. The predicted sequence is compared with the true system outputs in the generative sequence to evaluate the prediction accuracy. The root mean squared error (RMSE) and the root relative squared error (RRSE) are chosen as metrics.

The basic time difference $|t_{i+1} - t_i|$ between any adjacent sampling points is uniformly defined as 0.1 for all datasets. To unevenly construct sampling datasets, we randomly subsample 25%, 50% data points from each dataset respectively. Meanwhile, we also generate two evenly sampled datasets as control groups by subsampling 25%, 50% points with even time spans.

Baselines

Because the ODE-RSSM is a continuous-time state space model with stochastic hidden states, we choose three kinds of models as baselines:

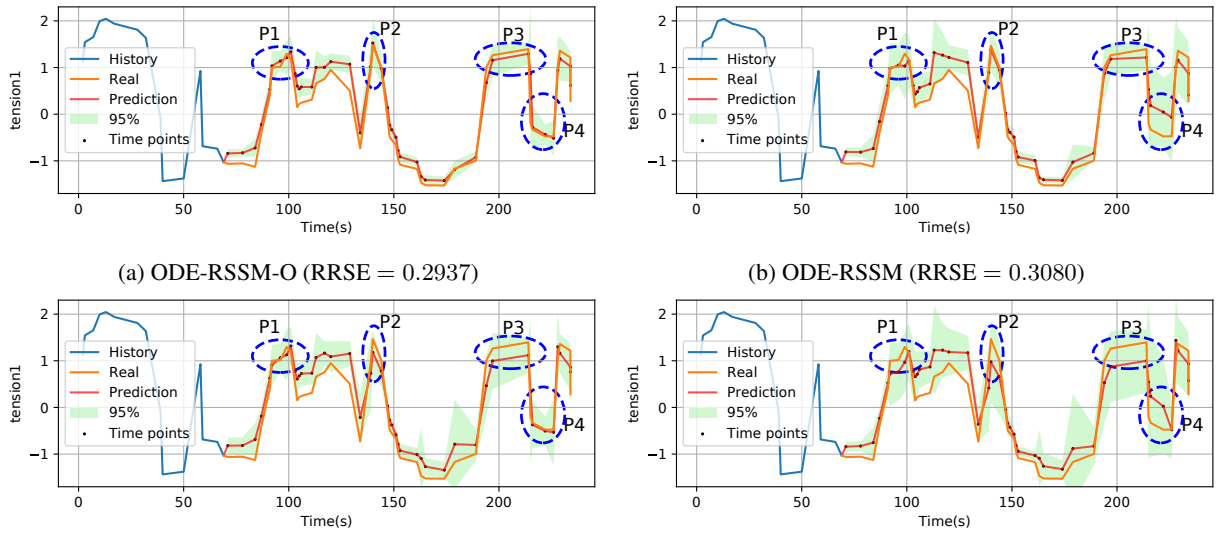
- VAE-based discrete-time models having stochastic latent states, include VAE-RNN (Fraccaro 2018), STORN (Bayer and Osendorfer 2014), and RSSM (Hafner et al. 2019).
- Continuous-time models having deterministic hidden states, include ODE-RNN (Rubanova, Chen, and Duvenaud 2019) and Time-Aware RNN (Demeester 2020).
- Continuous-time models having stochastic latent states, including Latent ODE(Rubanova, Chen, and Duvenaud 2019) and Latent SDE(Li et al. 2020).

For the ODE-RSSM and the discrete-time models with stochastic states transitions, we repeatedly predict $n_{\text{traj}} = 32$ trajectories in parallel from the generative model and measure the mean prediction error between the sampled trajectories and the ground-truth. For the ODE-RNN and the Time-Aware RNN, we introduce two RNN modules as the encoder and the decoder to separately infer the deterministic latent states and predict outputs. To evaluate the discrete-time models on unevenly sampled dataset, we incorporate the time difference $|t_{i+1} - t_i|$ as an extra input variable into the system inputs $\boldsymbol{u}_{t_i}$. Because the Latent ODE and SDE are time series models instead of controlled identification model. We embed the system inputs u(t), interpolated by zero order spline interpolation, in solving ODE and SDE, just as (4). All of the models are trained by the Adam optimizer where the learning rate is 5e-4. The training does not stop until the validation loss increases for 100 epochs.

RQ1: ODE-RSSM vs. Unevenly Sampling Baselines

Table 2 illustrates the evaluations of the ODE-RSSM and the baselines on three datasets. The two continuous-time models outperform the discrete-time models when the dataset is unevenly downsampled. Specially, for the unevenly downsampled datasets with sampling ratios 0.25 and 0.5, the prediction error of the discrete-time models on unevenly sampled dataset is significantly higher than the results on evenly sampled dataset. By contrast, the degradation of the continuous-time models is less serious. As an approximation of pure NODE, Time-Aware performs worse than ours.

ODE-RNN sometimes outperform the ODE-RSSM When latent overshooting is not used. ODE-RNN is trained with backpropagation through time (BPTT) which is vital for improving long-term prediction. Latent overshooting is analogous to BPTT in training temporal generative models, such as ODE-RSSM and RSSM. Without latent overshooting, ODE-RSSM is only trained by the loss in single step predictions. It is more fair to compare with ODE-RNN and ODE-RSSM-O. In most cases, the ODE-RSSM trained with latent overshooting has a lower error than ODE-RNN.



(a) ODE-RSSM-O (RRSE = 0.2937)

(b) ODE-RSSM (RRSE = 0.3080)

(c) RSSM-O (RRSE = 0.3345)

(d) RSSM (RRSE = 0.3926)

Figure 3: This figure illustrates the system outputs of *Winding* dataset predicted by four models in open loop. ODE-RSSM trained with latent overshooting outperforms the other three ones.

The Latent SDE and ODE also perform well in CSTR and Winding dataset, but their accuracies degrade obviously in the third dataset. The main reason is that the decoders of Latent ODE/SDE are pure differential equations which cannot handle the long-time delay in thickening dataset well in comparison to the GRU in ODE-RSSM.

In Figure 3, we visualize the predicted system outputs of ODE-RSSM and RSSM on the *Winding* test dataset. At the positions P1 and P4, the models trained by latent overshooting (a and c) outperform the ones without trained by latent overshooting (b and d). It demonstrates that training the generative model with multi-step KL-term improves the open-loop predictions. Furthermore, the two ODE-RSSMs significantly outperform the discrete-time RSSMs at P2 and P3, where the distances between adjacent time points are relatively longer. Although the long spacing $|t_{i+1} - t_i|$ leads to a considerable information loss, the increased prediction error of the ODE-RSSM model are smaller than the increments of RSSM. This result is consistent with the previous findings that ODE-Nets are good at extracting temporal feature from sparse time series (Quaglino et al. 2020). It can be concluded that it is more effective to handle irregular time steps by learning the system evolution in the continuous-time domain with a differential equation network, in contrast to adding the time delta as an extra input variable.

RQ2: Studying the Generalization Ability of ODE-RSSM in Varying Sampling Intervals

For studying RQ2, we first train the ODE-RSSM and RSSM models with the *Winding* dataset and the random sampling ratios are set to 0.25 and 0.5. For simplicity, the trained four models are named as ODE-RSSM (25%) ODE-RSSM (50%), RSSM (25%) and RSSM (50%). The percentages in parentheses represent the random sampling ratios of the training dataset. Next, we evaluate the four trained models on the test datasets where the random sampling ratios are changed to 25%, 50%, and 100% respectively and the results are shown in Table 3. The values in parentheses represent the rise or the decline of the prediction error compared with the error evaluated on training sampling ratio.

We first find that raising the sampling ratio of the test dataset from 25% to 50% leads to a decline of prediction error and the improvements on ODE-RSSMs are more obvious than RSSM models. When the sampling ratio in test dataset is lower than the ratio in training dataset, for instance, evaluating ODE-RSSM (50%) and RSSM (50%) under test dataset with 25% sampling ratio, the prediction error increases obviously and the increments of both ODE-RSSM (50%) and RSSM (50%) are closed to each other. Because lower sampling ratio increases the overall distance between adjacent sampling points, the time ranges for solving ODE-Net greatly exceed the overall distances of time intervals in training. The performance of ODE-RSSM is not guaranteed in such extrapolated predictions, but it is still better than discrete-time RSSM. Overall, the ODE-RSSMs show good generalization on the changeable sampling intervals.

RQ3: Evaluating the Time Efficiency of the Parallel Batched ODE Solver

We compare the proposed parallel ODE solver and competitors by evaluating their time consumption in solving $I = N \times B$ ODEs during training, where B is the constant batch size and N is the sequential length in solving KL-D as in (9).

In Figure 4, the lines marked with 'Parallel' denote employing the reparameterization method (13) for solving the ODEs in parallel. The ones marked with 'Union' solve the batched ODEs on the ordered union set of the time points (Rubanova,

Model	25% (uneven)				50% (uneven)				100%			
	RRSE		RMSE		RRSE		RMSE		RRSE		RMSE	
RSSM (25%)	0.5366	(0)	0.5313	(0)	0.4621	(-0.0745)	0.4617	(-0.0696)	0.4596	(-0.077)	0.4588	(-0.0725)
RSSM (50%)	0.6149	(**0.149**)	0.6069	(**0.1449**)	0.4659	(0)	0.462	(0)	0.4598	(-0.0061)	0.4583	(-0.0037)
ODE-RSSM (25%)	0.4934	(0)	0.4878	(0)	0.3976	(**-0.0958**)	0.3962	(**-0.0916**)	0.3861	(**-0.1073**)	0.3837	(**-0.1041**)
ODE-RSSM (50%)	0.5754	(0.1545)	0.5696	(0.1498)	0.4209	(0)	0.4198	(0)	0.3794	(-0.0415)	0.3771	(-0.0427)

Table 3: The results of evaluating ODE-RSSM and RSSM when the sampling ratio is different from the ratio in training dataset.

Chen, and Duvenaud 2019). Rk4 and dopri5 are two classical numerical approximate solvers (Chen et al. 2018). When the solver rk4 is used, the number of evaluations of the derivative net f_θ between any two adjacent time points is four, and the number of dopri5 is adaptive to ensure the approximation error be restricted in a given tolerance.

We first find that the time consumption of two methods marked with 'Union' increases linearly with I. In particular, the solution 'Union+rk4' consumes much more time than the solution 'Union+dopri5'. For the ordered union set, the distances between adjacent time points are small. When the adaptive dopri5 is used, the average number of ODE-Net evaluations between adjacent time points is less than four.

In comparison with solving the ODEs on the union set of time points, the two solutions marked with 'Parallel' are more time-efficient. For the choice 'Parallel+dopri5', the model has to guarantee the constraint from the given tolerances and it requires more time to evaluate the ODE-nets with the increase of length N. For 'Parallel+rk4', the number of evaluations is only four and the time consumption is irrelevant to the sequence length N and batch size B, which provides a solution for solving batched ODEs with approximate constant time complexity.

In most cases, the sequence length N is large, we therefore tend to employ the combination of reparameterization method and RK4 in the training stage. On the assumption of infinite GPU memory and taking full advantage of CUDA parallelization, the actual time consumption of estimating the multi-step ELBO for training is approximated to $\mathcal{O}(N + D)$, where the time consumption of sequential posterior inference is $\mathcal{O}(N)$ and the time of estimating KL-D is only $\mathcal{O}(D)$ because Parallel+rk4 consumes $\mathcal{O}(1)$ time in solving single-step batched ODEs.

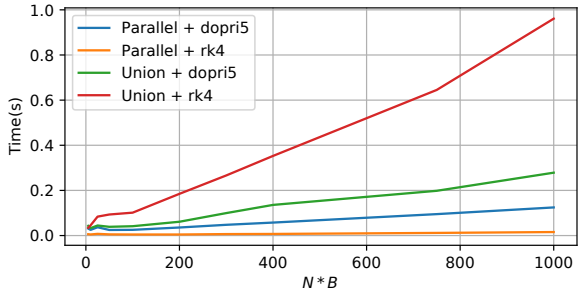


Figure 4: Efficiency of solving batched ODEs with different solvers.

Conclusion

This paper focuses on identifying input/outputs system with stochasticity from unevenly-spaced observations. We design a novel ODE-RSSM model, which introduces an ODE-Net to model the CT evolution of deterministic path between adjacent observed time points. The proposed model supports online inferring and prediction. Relying on the equivalence of linearly changing the integration limits, we further propose an efficient reparameterization method for solving the batched ODEs with non-uniform time spans. The method accelerates the model training by solving multi-step predictions at different positions in parallel. The experiments conducted on three input/output system datasets evaluate the proposed ODE-RSSM and other baseline models under the impact of randomly irregular downsampling. The results indicate that the ODE-RSSM model shows great accuracy in irregular settings and the proposed reparameterization method is quite time-efficient.

Acknowledgments

The authors acknowledge financial support from the National Natural Science Foundation of China (No. 61873299, No. 61902022, No. 61972028), the Departmental Start-up Fund of the Department of Computer Science in HKBU, Scientific and Technological Innovation Foundation of Shunde Graduate School, USTB(No.BK21BF002), and the Fundamental Research Funds for the Central Universities of China (FRF-TP-20-061A1Z). The computing work is partly supported by USTB MatCom of Beijing Advanced Innovation Center for Materials Genome Engineering.

References

Åström, K. J.; and Eykhoff, P. 1971. System identification—a survey. *Automatica*, 7(2): 123–162.

Bayer, J.; and Osendorfer, C. 2014. Learning stochastic recurrent networks. *arXiv preprint arXiv:1411.7610*.

Chen, R. T. Q.; Rubanova, Y.; Bettencourt, J.; and Duvenaud, D. 2018. Neural Ordinary Differential Equations. *Advances in Neural Information Processing Systems*.

Chung, J.; Kastner, K.; Dinh, L.; Goel, K.; Courville, A.; and Bengio, Y. 2015. A recurrent latent variable model for sequential data. *Advances in Neural Information Processing Systems*, 2015-Janua: 2980–2988.

Deisenroth, M.; and Rasmussen, C. E. 2011. PILCO: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, 465–472.

Demeester, T. 2020. System identification with time-aware neural sequence models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 3757–3764.

Doerr, A.; Daniel, C.; Schiegg, M.; Duy, N.-T.; Schaal, S.; Toussaint, M.; and Sebastian, T. 2018. Probabilistic recurrent state-space models. In *International Conference on Machine Learning*, 1280–1289. PMLR.

Fraccaro, M. 2018. *Deep latent variable models for sequential data*. Ph.D. thesis, Technical University of Denmark.

Fraccaro, M.; Kamronn, S.; Paquet, U.; and Winther, O. 2017. A disentangled recognition and nonlinear dynamics model for unsupervised learning. *Advances in Neural Information Processing Systems*, 2017-Decem(section 5): 3602–3611.

Fraccaro, M.; Sønderby, S. K.; Paquet, U.; and Winther, O. 2016. Sequential neural models with stochastic layers. *Advances in Neural Information Processing Systems*, 2207–2215.

Hafner, D.; Lillicrap, T.; Fischer, I.; Villegas, R.; Ha, D.; Lee, H.; and Davidson, J. 2019. Learning latent dynamics for planning from pixels. *36th International Conference on Machine Learning, ICML 2019*, 2019-June: 4528–4547.

Karl, M.; Soelch, M.; Bayer, J.; and van der Smagt, P. 2017. Deep Variational Bayes Filters: Unsupervised Learning of State Space Models from Raw Data. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Ke, N. R.; Singh, A.; Touati, A.; Goyal, A.; Bengio, Y.; Parikh, D.; and Batra, D. 2019. Modeling the Long Term Future in Model-Based Reinforcement Learning. In *International Conference on Learning Representations*.

Kidger, P. 2021. *On Neural Differential Equations*. Ph.D. thesis, University of Oxford.

Kingma, D. P.; and Welling, M. 2014. Auto-Encoding Variational Bayes. In Bengio, Y.; and LeCun, Y., eds., *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Lesort, T.; Díaz-Rodríguez, N.; Goudou, J.-F.; and Filliat, D. 2018. State representation learning for control: An overview. *Neural Networks*, 108: 379–392.

Li, X.; Wong, T.-K. L.; Chen, R. T.; and Duvenaud, D. 2020. Scalable gradients for stochastic differential equations. In *International Conference on Artificial Intelligence and Statistics*, 3870–3882. PMLR.

Liu, Y.; Wang, X.; Xing, Y.; Jin, D.; Yang, X.; and Shi, J. 2020. Learning continuous-time dynamics by Stochastic Differential Networks. *arXiv*, 1–13.

Moerland, T. M.; Broekens, J.; and Jonker, C. M. 2020. Model-based reinforcement learning: A survey. *arXiv preprint arXiv:2006.16712*.

Ogunmolu, O.; Gu, X.; Jiang, S.; and Gans, N. 2016. Nonlinear systems identification using deep dynamic neural networks. *arXiv preprint arXiv:1610.01439*.

Quaglino, A.; Gallieri, M.; Masci, J.; and Koutník, J. 2020. SNODE: Spectral Discretization of Neural ODEs for System Identification. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

Rubanova, Y.; Chen, R. T.; and Duvenaud, D. 2019. Latent ODEs for irregularly-sampled time series. *Advances in Neural Information Processing Systems*, 32(NeurIPS).

Wang, Y. 2017. A new concept using lstm neural networks for dynamic system identification. In *2017 American control conference (ACC)*, 5324–5329. IEEE.

Wikipedia. 2022. Integration by substitution — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Integration_by_substitution&oldid=1107237458. Accessed: 2022-12-01.

Yildiz, Ç.; Heinonen, M.; and Lähdesmäki, H. 2019. ODE2VAE: Deep generative second order ODEs with Bayesian neural networks. *Advances in Neural Information Processing Systems*, 32(NeurIPS).

Yildiz, C.; Heinonen, M.; and Lähdesmäki, H. 2021. Continuous-time Model-based Reinforcement Learning. In *International Conference on Machine Learning*, 12009–12018. PMLR.

Yuan, Z.; Li, X.; Wu, D.; Ban, X.; Wu, N.; Dai, H.-n.; and Wang, H. 2022. Continuous-Time Prediction of Industrial Paste Thickener System With Differential ODE-Net. *IEEE/-CAA Journal of Automatica Sinica*, 9(4): 686–698.