Continuous-Time Prediction of Industrial Paste Thickener System With Differential ODE-Net

Zhaolin Yuan, Xiaorui Li, Di Wu, Xiaojuan Ban, Nai-Qi Wu, *Fellow, IEEE*, Hong-Ning Dai, *Senior Member, IEEE*, and Hao Wang, *Senior Member, IEEE*

***Abstract*—It is crucial to predict the outputs of a thickening system, including the underflow concentration (UC) and mud pressure, for optimal control of the process. The proliferation of industrial sensors and the availability of thickening-system data make this possible. However, the unique properties of thickening systems, such as the non-linearities, long-time delays, partially observed data, and continuous time evolution pose challenges on building data-driven predictive models. To address the above challenges, we establish an integrated, deep-learning, continuous time network structure that consists of a sequential encoder, a state decoder, and a derivative module to learn the deterministic state space model from thickening systems. Using a case study, we examine our methods with a tailing thickener manufactured by the FLSmidth installed with massive sensors and obtain extensive experimental results. The results demonstrate that the proposed continuous-time model with the sequential encoder achieves better prediction performances than the existing discrete-time models and reduces the negative effects from long time delays by extracting features from historical system trajectories. The proposed method also demonstrates outstanding performances for both short and long term prediction tasks with the two proposed derivative types.**

Manuscript received August 29, 2021; accepted October 17, 2021. This work was supported by National Key Research and Development Program of China (2019YFC0605300), the National Natural Science Foundation of China (61873299, 61902022, 61972028), Scientific and Technological Innovation Foundation of Shunde Graduate School, University of Science and Technology Beijing (BK21BF002), Macao Science and Technology Development Fund under Macao Funding Scheme for Key R&D Projects (0025/2019/AKP), and Macao Science and Technology Development Fund (0015/2020/AMJ). Recommended by Associate Editor Xin Luo. *(Corresponding author: Xiaojuan Ban and Hong-Ning Dai.)*

Citation: Z. L. Yuan, X. R. Li, D. Wu, X. J. Ban, N.-Q. Wu, H.-N. Dai, and H. Wang, "Continuous-time prediction of industrial paste thickener system with differential ODE-net," *IEEE/CAA J. Autom. Sinica*, vol. 9, no. 4, pp. 686–698, Apr. 2022.

Z. L. Yuan, X. R. Li and X. J. Ban are with the Beijing Advanced Innovation Center for Materials Genome Engineering, Institute of Artificial Intelligence, Beijing Key Laboratory of Knowledge Engineering for Materials Science, School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China (e-mail: b20170324@xs.ustb.edu.cn; s20200654@xs.ustb.edu.cn; banxj@ustb.edu.cn).

D. Wu is with the Department of ICT and Natural Science, Norwegian University of Science and Technology, 6009 Alesund, Norway (e-mail: di.wu@ntnu.no).

N.-Q. Wu is with the Institute of Systems Engineering, and Collaborative Laboratory for Intelligent Science and Systems, Macau University of Science and Technology, Macao 999078, China (e-mail: nqwu@must.edu.mo).

H.-N. Dai is with the Department of Computing and Decision Sciences, Lingnan University, Hong Kong, China (e-mail: hndai@ieee.org).

H. Wang is with the Department of Computer Science, Norwegian University of Science and Technology, 2802 Gjøvik, Norway (e-mail: hawa@ntnu.no).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JAS.2022.105464

***Index Terms*—Industrial 24 paste thickener, ordinary differential equation (ODE)-net, recurrent neural network, time series prediction.**

I. Introduction

As a core procedure in modern mineral separation, a thickening process produces a paste with high concentration for subsequent tailing storage or backfilling [1]–[3]. During this thickening process, an industrial paste thickener achieves solid–liquid separation based on gravity sedimentation. The purpose of the industrial paste thickener is to efficiently control the final underflow concentration (UC). Most closed-loop control systems manipulate the underflow slurry pump speed and flocculant pump speed as the inputs to stabilize the underflow concentration within its specified range during operation. Previous studies [4], [5] showed that model prediction control (MPC) can facilitate the control process of thickening systems owing to the advantages of its high robustness and applicability. Hence, the accurate prediction of thickening systems has received extensive attention for the analysis and control of thickeners [5]–[7].

A complex industrial system such as the thickening system typically has the following key features:

1) Non-Linear System Dynamics: Most industrial systems have extremely complex high-order dynamical equations that are not affine or linear systems.

2) PartiaLly Observed Data: The information extracted from sensors or other available methods is incomplete. In particular, a number of unknown hidden variables exist in such systems.

3) Influence of Long Delays: The system states are influenced by external inputs or internal states that occur over a long previous time.

4) Continuous-Time (CT) Evolution: Because real industrial systems follow various physical laws, their time evolution can be expressed via CT differential equations.

The above features of a thickening system create challenges for predictive control. There are a number of studies addressing these challenges. Data-driven methods are emerging as one of the most successful techniques for modeling complex processes [8]. Traditional data-based CT system prediction methods focus on fitting high-order differential equations based on sampled noisy data from real systems. However, they lack the ability to cope with partially observed and extremely complex system dynamics. Recent advances of deep neural networks (DNNs) have shown their strengths in addressing these issues owing to their strong feature representation abilities and scalable parameter

structures, leading to the wide usage of DNNs in computer vision [9]–[11], natural language processing [12], [13], time series prediction [5], [14]–[17], and fault diagnosis [18]. However, most DNN-based system-modeling methods are based on discrete time, disregarding the CT properties of a system. The lost prior information from physical insights undoubtedly leads to the deterioration of the model accuracy.

Studies of prediction control can be categorized into two types. The first type provides short-term predictions for model-based control algorithms [5], such as model predictive control (MPC). The learned system provides deterministic prior knowledge of the dynamical systems, thereby approximating the infinite-horizon optimal control as a short-term optimization problem. The second type is mainly based on simulations, which imitate the outputs of an unknown system under a long-term feed of inputs [19]. Compared with short-term predictions, simulations require higher robustness and stability to provide long-term predictions. However, there are few studies on designing predictive models for short and long term predictions to support subsequent applications, such as MPC and simulations.

To address the above challenges, we propose a deep CT network composed of a sequential encoder, a state decoder, and a derivative module to learn the auto-regressive processes and influences from the system inputs based on real thickening data in an end-to-end manner. Specifically, the long-time system delay motivated us to utilize a sequential encoder to extract features from historical system trajectories. We designed the derivative module for the CT state space model based on a DNN. This module fits the non-linear CT evolution of the system and infers the non-observable information by introducing hidden states. Moreover, the problems of short and long term predictions are solved by feeding historical system trajectories and system inputs with arbitrary lengths to the model after incorporating the designed non-stationary and stationary systems into the trained model. The future system outputs are then predicted. The contributions of this paper are threefold:

1) We propose a novel deep-learning-based CT predictive model for a paste thickener. The deep learning network consists of three components: a sequential encoder, a state decoder, and a derivative module.

2) We design two kinds of derivative modules, named stationary and non-stationary systems, to handle the short-term and long-term prediction tasks, respectively.

3) We conduct extensive experiments on real industrial data collected from a real industrial copper mining process. The results demonstrate the outstanding performance of the proposed model in providing predictions for the thickener system with non-linear and time-delay properties. In addition, we conduct ablation studies to evaluate the effectiveness of each module in the proposed model.

The rest of the paper is organized as follows. We briefly introduce the related work in Section II. We then present the problem formulation in Section III. We next present the CT deep sequential model in Section IV. Experimental results are shown in Section V. We then summarize the paper and discuss future research directions in Section VI.

II. Preliminaries and Related Work

As a core device in a thickening system, a paste thickener is generally composed of a high sedimentation tank and a raking system. Fig. 1 depicts the general structure of a thickener and its key components. After being fed with flocculant and tailing slurry with a low concentration, underflow with a high concentration is discharged from the bottom of the thickener and is then used to produce paste in the subsequent procedures. The prediction of a thickening system refers to the estimation of the future system outputs, such as the underflow concentration and mud pressure based on the historical system trajectories and system inputs. The prediction of a thickening system is essentially similar to system identification [8]. For system identification, interpretable model structures are designed based on prior knowledge, and the parameters are determined by fitting real data. As one of the subsequent applications of identified models, the prediction forecasts the system outputs according to the inputs.

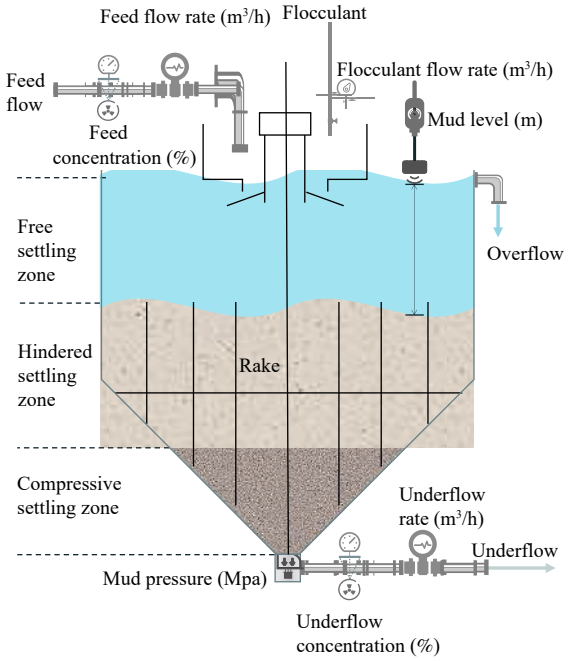


Fig. 1. Crude slurry flow with a low concentration is fed into the mix tank accompanied by flocculant. Under the effect of the flocculant, particles agglomerate to larger clumps and concentrate at the bottom. The paste thickener continuously produces underflow with a high concentration and clear water in an overflow pipe located at the top of the thickener.

A. Prediction of Thickening Systems

The prediction methods for thickening systems can be categorized into two types: 1) gray-box thickening system simulations and 2) black-box thickener system predictions. In the gray-box simulations, the sedimentation process is mainly considered from a physical perspective [4], [20], [21]. Theory-based gray-box methods can be exactly explained and implemented effectively for specific systems. However, most of them are mainly built on many ideal hypotheses and suffer from the complexity of slurry particle dynamics and external unknown environment disturbances.

In contrast, the black-box methods do not require prior assumptions or constraints to be given. A complete parameterized model with a high degree of freedom is defined to predict the system outputs and learn the optimal parameters from real data. Since the offline system trajectories from an industrial system are always available and adequate, black-box-based methods, including latent factor model [22]–[24], imitation learning [25], and deep neural network [26] have been widely used in the current industrial systems [27]–[32].

Moreover, Oulhiq *et al*. [33] used a black-box linear dynamic model with a deterministic time delay to identify an industrial thickener system using historical data. Such parameterized linear model lacks adequate expressivity to represent the non-linear properties in a thickening system. Most recently, random forest model is presented for modeling a paste thickening system based on a purely data-driven approach for modeling and evolutionary strategies [34]. Because random forest model only fits the thickening system dynamic in single step, it ignores the time delay and the correlations between adjacent positions in sequential inputs and outputs. A bidirectional gated recurrent unit (BiGRU) with an encoder–decoder deep recurrent neural network is introduced to model thickening systems [5]. Yuan *et al*. [6] proposed a dual-attention recurrent neural network to model the spatial and temporal features of a thickening system, thereby improving the prediction accuracy of the underflow concentration. However, the above studies [5], [6] only focus on discrete-time system predictions rather than a CT thickening system.

B. Prediction of Continuous-Time Systems

The prediction of physical systems based on CT models directly from sampled data has the following advantages [8]: 1) transparent physical insights into the system properties, 2) inherent data filtering, and 3) the capability of dealing with non-uniformly sampled data. For any numerical schemes for solving CT differential equations, sophisticated discretization methods have high accuracy but suffer from enormous time and memory costs. A recently developed advanced ordinary differential equation (ODE) solver [35] introduces the reverse-mode automatic differentiation of ODE solutions, thereby only requiring $O(1)$ memory cost. Meanwhile, this method also allows the end-to-end training of ODEs within a large DNN. Moreover, Demeester [19] proposed a stationary CT state space model for predicting an input–output system when the observations from a system are unevenly sampled. Although it has successfully improved the accuracy and stability of long-term predictions by introducing a stationary system, it did not take advantage of a non-stationary system in the short-term prediction task.

Compared with the existing CT models, our model considers both short-term and long-term predictions, thereby achieving outstanding performance.

III. Formulation and Notation of Paste Thickening System Prediction

In a thickening industrial process, the key system outputs $\mathbf{y}(k) \in \mathbb{R}^2$ include the underflow concentration $y_1(k)$ and mud pressure $y_2(k)$, where k is the sampling time index. Both $y_1(k)$ and $y_2(k)$ are influenced by the control inputs and other parameters, including the feeding flow rate $u_1(k)$, feeding concentration $u_2(k)$, raking speed $u_3(k)$, underflow rate $u_4(k)$, and flocculant flow rate $u_5(k)$. As the focus of this paper is on the prediction rather than the control problem, we do not distinguish the control inputs and uncontrollable measurable parameters. All of the controllable inputs or measurable parameters are regarded as system inputs $\mathbf{x}(k) \in \mathbb{R}^5$.

Because some crucial parameters of thickeners are unavailable due to the limitations on the monitoring sites, thickening systems are partially observed and have non-deterministic dynamics. Meanwhile, the influences of the system inputs are essentially non-linear and time delayed. These key characteristics of thickening systems cause us to employ a black-box data-driven method and utilize sequential data to infer the latent system state in the designed model. As a result, the prediction problem of a thickener is a problem of estimating the future system outputs under known system inputs and historical system trajectories. We first assume that the historical system input $\mathbf{X}_p^{N_x} = [\mathbf{x}(-N_x), \mathbf{x}(-N_x+1), \ldots, \mathbf{x}(-1)]$, output $\mathbf{Y}_p^{N_y} = [\mathbf{y}(-N_y), \mathbf{y}(-N_y+1), \ldots, \mathbf{y}(-1)]$, and future input $\mathbf{X}_f^M = [\mathbf{x}(0), \mathbf{x}(1), \ldots, \mathbf{x}(M-1)]$ are available. Note that $\mathbf{X}_f^M$ is available because system inputs are known signals in MPC or simulations. With a symbolic expression, the problem is then formulated as the sequential prediction structure given as follows:

$$\begin{cases} \mathbf{h}(0) = \mathcal{F}(\mathbf{X}_p^{N_x}, \mathbf{Y}_p^{N_y}) \\ \mathbf{H}^{\mathbf{M}} = D(\mathbf{h}(0), \mathbf{X}_f^M) \\ \mathbf{y}(k) = g(\mathbf{h}(k)) \end{cases} \tag{1}$$

where the module $\mathcal{F}(\cdot)$ produces the initial hidden state $\mathbf{h}(0)$, which carries the historical information from $\mathbf{X}_p^{N_x}$ and $\mathbf{Y}_p^{N_x}$. The entire system state $\mathbf{H}^M$ is estimated according to the initial hidden state and future system inputs based on the rational module $D(\cdot,\cdot)$. Each hidden state $\mathbf{h}(k)$ in sequence $\mathbf{H}^M$ carries the system information for index k. In addition, the real system output $\mathbf{y}(k)$ is available by decoding $\mathbf{h}(k)$ via the non-linear function $g(\cdot)$.

Formulation (1) maps the sequences $\mathbf{X}_p^{N_x}$, $\mathbf{Y}_p^{N_y}$, and $\mathbf{X}_f^M$ to the predicted future system output sequence $\hat{\mathbf{Y}}_f^M = [\hat{\mathbf{y}}(1), \hat{\mathbf{y}}(2), \ldots, \hat{\mathbf{y}}(M)]$, similar to the Seq2Seq model that is widely used in natural language processing (NLP) [36]. However, there is a slight distinction between an NLP task and input–output system prediction. Under the restriction of online system prediction, the calculation of $\mathbf{h}(k)$ depends on $\{\mathbf{h}(i), i \le k-1\}$ and $\mathbf{X}_f^k = \{\mathbf{x}(i), i \le k-1\}$ only, and is equivalent to a prediction problem $p(\mathbf{h}(k)|\mathbf{h}(0), \mathbf{X}_f^k)$ instead of a smoothing problem $p(\mathbf{h}(k)|\mathbf{h}(0), \mathbf{X}_f^M)$. This restriction motivates the use of an auto-regressive system in the model framework.

To construct a rational module D in the predictive model, the auto-regressive discrete-time state space model [37] is a simple and effective solution. Similarly, we have the following formulation:

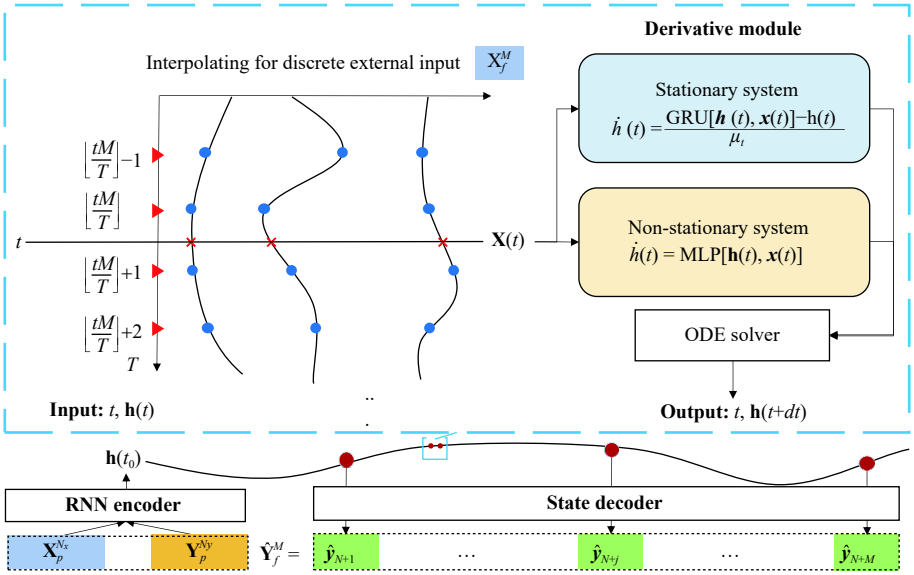


Fig. 2. Proposed model is composed of several components. The recurrent neural network (RNN) encoder outputs the initial hidden state $\mathbf{h}(t_0)$ according to historical sequences from the system. The derivative module is embedded into the ordinary differential equation (ODE) solver to calculate the hidden state at an arbitrary time. A parallel interpolation mechanism is embedded into the derivative module, which interpolates discrete input sequences to a CT series. Finally, the state decoder module transforms the hidden state to the predicted system output.

$$\mathbf{h}(k) = d(\mathbf{h}(k-1), \mathbf{x}(k-1)) \tag{2}$$

$$\mathbf{y}(k) = g(\mathbf{h}(k)) \tag{3}$$

where hidden state $\mathbf{h}(k)$ encodes the historical trajectories of the system in a dense and fixed-length vector space. The influences from external inputs can be viewed as step-by-step non-linear transformations on the hidden state. The model can predict the internal state $\mathbf{h}(k)$ immediately when $\mathbf{x}(k-1)$ and $\mathbf{y}(k)$ are provided simultaneously. However, some previous studies [38], [39] point out that a thickening system can be modeled as CT differential equations based on physical insights. Thus, we follow the prior knowledge from these previous studies and define a parameterized CT differential equation model to fit the first-order derivative of the hidden state as follows:

$$\dot{\mathbf{h}}(t) = d(\mathbf{h}(t), \mathbf{x}(t)). \tag{4}$$

We replace index k by t to represent that the new time index t is continuous in a specific range instead of a discrete integer. The formulation of the prediction of the thickening system can be summarized by (1) and (4). The goal is to learn the parameterized modules, including the sequential encoder $\mathcal{F}(\cdot)$, CT derivative module $d(\cdot,\cdot)$, and state decoder $g(\cdot)$, based on the collected system trajectories from the real thickening system.

IV. Continuous-Time Deep Sequential Model for Thickening System Predictions

We propose an integrated deep neural network to implement Formulations (1) and (4). Fig. 2 illustrates all components and their connections in the network. With specific historical trajectories $\mathbf{X}_p^{N_x}$ and $\mathbf{Y}_p^{N_y}$ in the conditional range and $\mathbf{X}_f^M$ in the predicted range, the model outputs $\hat{\mathbf{Y}}_f^M$ as an estimation of real outputs $\mathbf{Y}_f^M$.

The proposed model works in the following steps. First, a **RNN encoder**, a recurrent neural network (RNN) network, is introduced to encode historical trajectories $\mathbf{X}_p^{N_x}$ and $\mathbf{Y}_p^{N_y}$ to the hidden state $\mathbf{h}(t_0)$, where $\mathbf{h}(t_0)$ is the initial state of solved ordinary differential equation. The one-order derivative of ordinary differential equation is defined based on a **Derivative module**, which utilizes the hidden state $\mathbf{h}(t)$ and external inputs $\mathbf{x}(t)$ at arbitrary time t as inputs to estimate the instantaneous derivative of hidden state, $\dot{\mathbf{h}}(t)$. The external inputs $\mathbf{x}(t)$ at arbitrary time t is computed from a **Parallel spline interpolation**, which interpolates the discrete external inputs $\mathbf{X}_f^M$ to the continuous-time form. The ordinary differential equation defined based on the initial state $\mathbf{h}(t_0)$ and derivative module is solved by ODE solvers and the complete continuous-time hidden state $\mathbf{h}(t)$ in the time range $[0 \le t \le T]$ is produced. Finally, **State decoder**, a multi-layer perceptron (MLP) network, is employed to predict the future system outputs $\hat{\mathbf{Y}}_f^M$ from the evolved hidden states $\mathbf{h}(t)$.

We next present technical details of each module of the proposed model.

A. RNN for Encoding Historical Sequences

Because a thickening process suffers from long time delays, we introduce historical system trajectories $\mathbf{X}_p^{N_x}$ and $\mathbf{Y}_p^{N_y}$ as parts of the model inputs. We employ a basic RNN model to infer the initial value $\mathbf{h}(0)$ in ordinary differential equations by encoding two historical sequences denoted by $\mathbf{Y}_P^{N_y}$ and $\mathbf{X}_p^{N_x}$ into a fixed-length hidden state. We thus have

$$\mathbf{h}(t_0) = \mathbf{h}(0) = f_{\text{RNN}}(\mathbf{Y}_P^{N_y}, \mathbf{X}_p^{N_x}, \theta_f) \tag{5}$$

where $f_{\text{RNN}}(\cdot)$ is a forward RNN network, and N_y and N_x are two important parameters to be configured. Based on

industrial experience, we use the system delay as prior knowledge denoted by T_d. If we assume a uniform sampling of the sensors for a sampling interval T_s, N_y and N_x can be estimated according to the equation $N_y = N_x = N = T_d/T_s$. The influence of parameter N on the model accuracy is examined in Section V. In the thickening system, the correlations between the current system state and historical trajectories are mostly compressed in the short term. This property encourages us to use a simple and unidirectional RNN to encode the historical trajectories of a thickening system. The solved hidden state $\mathbf{h}(t_0)$ from encoder involves all necessary information of historical trajectories and is represented as an initial state of solved ODE.

B. ODE Solver for Modeling State Space

We employ the parameterized CT state space model to represent the relations between the system inputs, hidden states, and outputs:

$$\dot{\mathbf{h}}(t) = d(\mathbf{h}(t), \mathbf{x}(t), \theta_d) \tag{6}$$

$$\mathbf{y}(t) = g(\mathbf{h}(t)). \tag{7}$$

The state space model encodes the features from historical sequences to the fixed-length state $\mathbf{h}(t)$. The utilization of hidden state $\mathbf{h}(t)$ is crucial to handle long time delays and incomplete observations. For a predicted sequence with a length equal to M, we construct a bijective function between the discrete indices of integers $[0, 1, \ldots, M]$ and time range $[t_0 \le t_k \le t_M]$. Each $\mathbf{h}(t_k)$ associated with a specific integer index k is the ODE solution with an initial state $\mathbf{h}(0)$ at the time $t = t_k$.

To construct a learnable differential system, we employ a differentiable ODE-net [35] to learn the above state space model. For a scalar-valued loss function $L(\cdot)$, which is determined based on any prediction metrics, the input $\mathbf{h}(t_k)$ is the estimated hidden state from the ODE solver (denoted by ODESolve) at time t_k. Thus, we have

$$\begin{aligned} L(\mathbf{h}(t_k)) &= L\left(\mathbf{h}(t_0) + \int_{t_0}^{t_k} d(\mathbf{h}(t), \mathbf{x}(t), \theta_d) dt\right) \\ &= L(\text{ODESolve}(\mathbf{h}(t_0), d, t_0, t_k, \theta_d)). \end{aligned} \tag{8}$$

To train parameters θ_d and minimize $L(\cdot)$, we require gradients $\partial L/\partial \theta_d$ from (8). The gradient of the loss, which depends on the hidden state, is called the adjoint $\mathbf{a}(t) = \partial L/\partial \mathbf{h}(t)$. Its dynamics are described by another ODE, which can be derived according to the chain rule, as follows:

$$\frac{d\mathbf{a}(t)}{dt} = -\mathbf{a}^T(t) \frac{\partial d(\mathbf{h}(t), \mathbf{x}(t), \theta_d)}{\partial \mathbf{h}(t)}. \tag{9}$$

The gradients from loss $L(\cdot)$ with respect to parameters θ_d can be obtained by performing a third integration,

$$\frac{\partial L}{\partial \theta_d} = -\int_0^{t_M} \mathbf{a}^T(t) \frac{\partial d(\mathbf{h}(t), \mathbf{x}(t), \theta_d)}{\partial \theta_d} dt. \tag{10}$$

Detailed proofs can be found elsewhere [35]. Under the deterministic network structure d and parameter θ_d, all integrals for solving $\mathbf{h}(t)$, $\alpha(t)$, and $\partial L/\partial \theta$ can be evaluated. Any numerical methods for solving ODEs can be used here for an approximate solution, including the Euler, Mid-Point, and Runge-Kutta methods.

Generally, an ODE solver with a lower error tolerance increases the frequency for calling differential function d. It leads to more time consumption but results in a higher accuracy. This guideline is also tenable when we construct a neural ODE network to fit sequential datasets. The detailed comparisons of the time cost and accuracy are shown in Section V.

It is worth investigating the definition of a suitable structure of d. The most intuitive solution is to employ a basic neural network to estimate the derivative that is named non-stationary model.

Non-Stationary:

$$d(\mathbf{h}(t), \mathbf{x}(t), \theta_d) = \text{MLP}(\mathbf{h}(t), \mathbf{x}(t), \theta_d) \tag{11}$$

where MLP$(\cdot)$ denotes a multi-layer perceptron. The combination of a non-stationary system with an ODE solver has a strong similarity to residual connections, which have been widely used in other advanced deep networks [35].

In the field of stochastic process analysis, non-stationary systems are a stochastic process with a mean and covariance that vary with respect to time [40]. Differencing [41] is an effective way to make non-stationary time series stationary by eliminating trend and seasonality. Generally, a thickening system has strong trends in the underflow concentration, mud pressure, and other core variables. The thickening system is an approximation of non-stationary systems, indicating that the differencing operation can improve the fitting accuracy. In (11), the derivative module intrinsically learns the first-order difference of the hidden states in the latent space. In contrast to the operation of differencing the system outputs directly, a model that differences the hidden states has an equivalent or stronger ability to represent a non-stationary system that is of first or even higher order. However, the non-stationary system (11) also suffers from a severe problem when handling long-term prediction tasks. To solve an ODE over long intervals, repetitive accumulation in a CT range can lead to a significant magnitude increase of the hidden states. Consequently, the estimation error will grow accordingly, resulting in the difficulties in achieving accurate system output from the decoder.

Therefore, we devise another derivative module, namely, the stationary system, to handle the long-term prediction problem. In particular, we have

Stationary:

$$d(\mathbf{h}(t), \mathbf{x}(t), \theta_d) = \frac{1}{\mu_t}(\text{GRU}(\mathbf{h}(t), \mathbf{x}(t), \theta_d) - \mathbf{h}(t)) \tag{12}$$

where GRU denotes a gated recurrent unit.

In a stationary system, GRU$(\mathbf{h}(t), \mathbf{x}(t), \theta_d)$ determines a target based on the current external input $\mathbf{x}(t)$ and hidden state $\mathbf{h}(t)$. The factor μ_t regularizes the speed toward the target. Specifically, the outputs from the GRU are limited to $(-1, 1)$ according to the network standard. Regardless of how much time has passed, the state $\mathbf{h}(t)$ that is sent to the decoder module is stable in the range of the GRU's output. This property significantly contributes to the stability of a stationary model in a long-term prediction task.

In a stationary system, we use a GRU to construct the derivative module because it has a strong ability to carry long-time information. In a non-stationary system, consecutive accumulations diffuse the hidden state in an unconstrained range. Thus, we employ the MLP to learn the first-order difference $\dot{\mathbf{h}}(t)$ directly under $\mathbf{h}(t)$ and $\mathbf{x}(t)$.

C. Parallel Spline Interpolation

Note that the calculation of $\dot{\mathbf{h}}(t)$ in (11) and (12) depends on the external input $\mathbf{x}(t)$, which may not exist in our dataset. External input sequences $\mathbf{X}_f^M$ in the training data are discrete, while the computation of the ODE needs $\mathbf{x}(t)$ in a CT range. Before each forward pass of the network, it is necessary to interpolate the external inputs to the continuous form. Deep networks are typically trained in mini-batches to take advantage of efficient data-parallel graphics processing unit (GPU) operations. Therefore, we implement a parallel spline interpolation mechanism on top of PyTorch, which is a well-known deep learning framework.

In our dataset, the external input data are evenly sampled, thereby simplifying the implementation of parallel interpolation. To simplify the explanation, we assume that the dimension of the external input is equal to 1.

We begin with specific input sequences organized in batch $\mathbf{X} = [\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^m]$, where m is the batch size and $\mathbf{x}^i = [x_1^i, x_2^i, \ldots, x_M^i]$ is an independent input sequence consisting of M sampled data in discrete steps. We define a time interval $[0, T]$ to represent the M steps and for any given time index t in the interval with constraint $0 \le t \le T$. In the M steps, the nearest integer index in the left-hand side of t is $k = [tM/T]$. The nth-order spline interpolation can be implemented by finding matrix $\mathbf{A}$ according to the following equation:

$$\mathbf{A} \cdot \begin{bmatrix} k^0 & \cdots & k^n \\ (k+1)^0 & \cdots & (k+1)^n \\ \vdots & \ddots & \vdots \\ (k+n)^0 & \cdots & (k+n)^n \end{bmatrix} = \begin{bmatrix} x_k^1 & \cdots & x_k^m \\ x_{k+1}^1 & \cdots & x_{k+1}^m \\ \vdots & \ddots & \vdots \\ x_{k+n}^1 & \cdots & x_{k+n}^m \end{bmatrix}. \tag{13}$$

The interpolated inputs in a batch at time t are obtained as follows:

$$\left[x^1(t), x^2(t), \ldots, x^m(t)\right] = \left(\left[1, \frac{tM}{T}, \cdots, \left(\frac{tM}{T}\right)^n\right]\mathbf{A}\right). \tag{14}$$

The parallel multiplication of the matrix can be effectively implemented in the deep learning framework.

D. State Decoder

The state decoder mechanism is essentially a fully connected network. We therefore have the following equation to represent the output,

$$\hat{\mathbf{y}}(t) = \mathbf{V}^T \tanh(\mathbf{W}\mathbf{h}_t + \mathbf{b}_w) + \mathbf{b}_v. \tag{15}$$

Compared with other state space models that only employ a single matrix for decoding, the nonlinear decoder is chosen because the accumulative form in (11) causes the range of the input $\mathbf{h}(t)$ to be non-deterministic. The activation function $\tanh(\cdot)$ constrains the output of the decoder to a rational range.

E. Model Training

Since all of the operations of the ODE solver in our model are smooth and differentiable, we can train the complete model by the standard back-propagation algorithm with the loss function defined as follows:

$$O\left(\hat{\mathbf{Y}}^M, \mathbf{Y}^M\right) = \frac{1}{M}\sum_{i=1}^{M} |\mathbf{y}_i - \hat{\mathbf{y}}_i|^2. \tag{16}$$

To prepare the tuples $(\mathbf{X}_p^N, \mathbf{Y}_p^N, \mathbf{X}_f^M, \mathbf{Y}_f^M)$ for training models from the training dataset, a sliding window of size $N+M$ moves along the real sequential data. When the window reaches position i, four sequences ($\mathbf{X}_p^N = \mathbf{X}[i : i+N]$, $\mathbf{Y}_p^N = \mathbf{Y}[i : i+N]$, $\mathbf{X}_f^M = \mathbf{X}[i+N : i+N+M]$, $\mathbf{Y}_f^M = \mathbf{Y}[i+N : i+N+M]$) are collected as a piece of data for training. We set the moving size of sliding windows as $D = 1$ for generating training, validation, and test datasets. In the validation and test datasets, the size of sliding window changes to $N+L$, in which L represents the length of the predicted sequence (which may not be equal to M). The model is trained by feeding the only training dataset and successively validated and evaluated on different validation and test datasets with specific L. The detailed procedure for constructing datasets is illustrated in Fig. 3.

V. Experimental Results

This section presents experimental results for the proposed method on the dataset of real thickening systems. We mainly investigate three issues: RQ1: What are the advantages of employing a CT deep sequential network with a high-accuracy ODE solver for modeling a thickening system? RQ2: What are the pros and cons of using stationary and non-stationary systems in prediction tasks? RQ3: How do the different interpolation methods and sequential encoder affect the accuracy of the proposed CT model? We first describe the dataset, the hyper-parameters of the model, and the training and test configurations. We then present the detailed experimental results.

A. Thickening System Dataset

For our experiments, the dataset was collected from the paste thickener manufactured by the FLSmidth from the NFC Africa Mining PLC, Zambian Copperbelt Province. Fig. 4 illustrates two identical thickeners in our experiments. They are used to concentrate copper tailings to produce paste in the backfilling station. Both devices operate in the closed-loop mode with PID controllers.

Some key technical parameters of the studied thickener are listed in Table I.

The measured data are sampled evenly with two-minute intervals from May 2018 to February 2019. A short piece of original dataset is shown in Table II. The collected data come from seven monitoring sensors just as the defined $\mathbf{y}(k)$ and $\mathbf{x}(k)$ in Section III. After deleting the records corresponding to the time when the system was out of service, there are 24 673 pieces of data remaining.

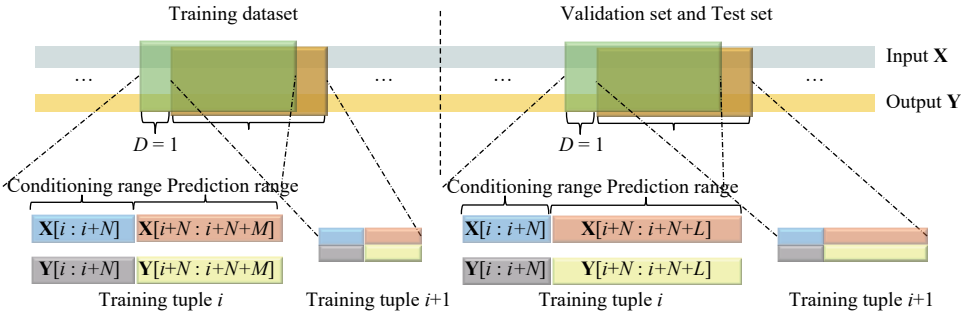


Fig. 3. Illustration of the process of building both the training, validation, and test datasets. An independent data tuple for training or testing is composed of four vector sequences. $\mathbf{X}[i:i+N]$ and $\mathbf{Y}[i:i+N]$ represent the historical trajectories in conditional range. $\mathbf{X}[i+N:i+N+M]$ and $\mathbf{X}[i+N:i+N+L]$ represent the inputs sequences, which have equivalent length with predicted sequences. $\mathbf{Y}[i+N:i+N+M]$ and $\mathbf{Y}[i+N:i+N+L]$ represent the real system outputs. The former is utilized to generate optimized loss in training and the latter is only used in testing and validation phase for evaluating the accuracy of prediction.

(a) Top view of the paste thickener (b) Upward view of feeding pipes

Fig. 4. The figures illustrate two identical paste thickeners in our experimental mining station, including one primary and one alternate thickener. Both devices operate in closed-loop mode with proportional-integral-derivative (PID) controllers.

TABLE I
SOME KEY TECHNICAL PARAMETERS OF THE THICKENER

Parameter	Value
Diameter	18 m
Height	22.27 m
Height of straight cylinder	10.06 m
Height of zone	7.1 m
Maximum capacity	2983 m^3
Overflow level (from ground)	15.3 m
Maximum underflow speed	260 m^3/h
Maximum feeding speed	1500 m^3/h

We employ the first 70% of the entire dataset to train the model. In the remaining 30% portion, the first 15% is used for validation to determine the best training epochs, and the other 15% is the test dataset for evaluating the model accuracy. By splitting and building the inputs-outputs sequential tuples according to Fig. 3, there are 17 131 tuples left for training and 3561, 3421, 3121 tuples left for testing and validation for $L = 60, 200, 500$, respectively. All the datasets are normalized to standard normal distributions with a unified mean and variance before the training and test phases.

B. Experimental Setup

We use the mini-batch stochastic gradient descent (SGD) with the Adam optimizer [42] to train the models. The batch size is 512, and the learning rate is 0.001 with an exponential decay. The decay rate is 0.95, and the period for decay is 10 epochs. The size of the hidden state $\mathbf{h}(t)$ in ODE is 32. The RNN encoder module has a single hidden layer and the size is equal to 32 that is consistent with the size of hidden state $\mathbf{h}(t)$. The size of the hidden layer in state decoder is 64. In both the adaptive ODE solvers, the time for solving the ordinary differential equations increases if we reduce the tolerance of approximate error. For balancing the time cost and accuracy, we set the relative tolerance to 1E–4 and absolute tolerance to 1E–5 in all of experiments.

During the training procedure, the length of the historical sequences denoted by N is 80, and the length of predicted outputs denoted by M is 60. The best-performing model in the validation dataset is chosen for further evaluation with the test dataset. The training and test phases were performed on a single Nvidia V100 GPU. The implementation uses the PyTorch framework. We define the CT range as $0 \le t \le M\delta_t$ for given discrete integral indices $[0,1,\ldots,M]$. The time interval δ_t of the adjacent data points is set to 0.1. Accordingly, the normalized factor μ_t in (12) is also set to 0.1. When we use the Euler approximation to solve an ODE equation in a stationary system, the predicted hidden state in the next time step is equal to the output of the GRU cell corresponding to the discrete-time system:

$$\begin{aligned}\mathbf{h}(t+\delta_t) &= \mathbf{h}(t) + \delta_t \cdot \frac{\mathrm{GRU}(\mathbf{h}(t), x(t), \theta_d) - \mathbf{h}(t)}{\mu_t} \\ &= \mathrm{GRU}(\mathbf{h}(t), x(t), \theta_d). \end{aligned} \tag{17}$$

We use the averaged root relative squared error (RRSE) and mean squared error (MSE) of the underflow concentration to evaluate the performance of the different models. The RRSE is defined by the Equation (18) with the prediction length L:

$$\mathrm{RRSE} = \sqrt{\sum_{j=1}^{L} \frac{e_j^2}{\left(\hat{y}_j - \bar{y}\right)^2}}, \quad e_j = \hat{y}_j - y_j. \tag{18}$$

The RRSE can be interpreted as the normalized root mean squared (RMS) error.

C. Results and Discussion

1) Main Results

We investigate the influence of the types of ODE solvers

TABLE II
A TABULAR EXAMPLE OF PASTE THICKENING SYSTEM DATASET

Collected timestamp	Feed flow rate	Feed concentration	Mud pressure	Rake speed	Flocculant flow rate	Underflow rate	Underflow concentration
2018/5/9 10:20	164.47	16.47	18.41	500.58	4.30	58.96	59.72
2018/5/9 10:22	169.21	15.51	17.99	500.16	4.06	61.56	58.88
2018/5/9 10:24	141.78	15.30	16.41	500.56	4.06	59.97	59.26
2018/5/9 10:26	305.67	25.31	16.11	500.99	4.07	59.46	58.77
2018/5/9 10:28	328.70	28.28	16.43	501.42	4.43	59.68	59.43
2018/5/9 10:30	323.96	25.90	17.11	501.56	4.40	61.40	60.09

and system types. We select four ODE solvers: Euler, Mid-Point, fourth-order Runge-Kutta (RK4), Dormand-Prince (Dopri5) [35], and 3-order Bogacki-Shampine (Bosh) [43]. We investigate the performances of those ODE solvers in both non-stationary and stationary systems. To make a trade-off between the model accuracy and time consumption, we set the relative tolerance to 1E–4 and the absolute tolerance to 1E–5. Moreover, we also consider the discrete-time deep sequential model for the state space (DT-State-Space), the attention-based Seq2Seq model (Attention-Seq2Seq) [5], and Transformer [44] for comparison. The DT-State-Space [45] model employs a parameterized per-time-series linear state space model based on a recurrent neural network (RNN) to forecast the probabilistic time series. The sizes of state space and RNN hidden layer are set to 16 and 32, respectively. The hyperparameters setting of Transformer and Attention-Seq2Seq are kept with the original literature.

We conduct three groups of experiments to investigate the RRSEs, MSEs, and time consumption of models with prediction lengths of $L = 60$, 200, and 500.

a) Comparison of proposed and other baseline models:

We first examine the performance of the Attention-Seq2Seq model, the DT-State-Space model, and Transformer, which are defined in discrete-time settings in Table III. Although they perform competitively, better than the proposed models with the Euler ODE solver, they perform worse than the models with high-order ODE solvers, especially for long-term predictions. The results also indicate that employing a CT model is consistent with the features of the CT evolution in thickening systems, thereby improving the prediction accuracy.

b) Comparison of different ODE solvers:

We analyze the comparisons of different ODE solvers respectively from stationary system and non-stationary system. When the derivative module is defined as a non-stationary system and we only focus on short-term prediction with $L = 60$, we find that the Euler method achieves relatively higher RRSE and MSE values (i.e., poorer prediction performances) than the other four ODE solvers, though it has a much lower time consumption than the other solvers. As the simplest method for solving ODEs, the Euler method evaluates the derivative network only once between two adjacent time points. Meanwhile, the Mid-Point and RK4 methods have higher prediction accuracies than the Euler method, since they evaluate the derivative network two and four times, respectively, between two adjacent time points. Moreover, the Dopri5 and Bosh methods achieve better accuracies, though they have higher time consumptions. Dopri5 performs slightly better than Bosh. As adaptive methods in the Runge-Kutta family, the Dopri5 and Bosh methods ensure that the output is within a given tolerance of the true solution. Their time consumptions for solving an ODE equation increase as the accuracy tolerance is decreased.

Strangely, with the increase in the prediction length, we find that the accuracies of non-stationary models crash gradually and the degradation of Euler is slightly lower than the others. The reason of this inconsistent phenomenon is that non-stationary system brings accumulative errors in long-term predictions. High-order ODE solvers evaluate the derivative module more times recursively, which brings more accumulative errors. Not only do the high-order solvers not improved the accuracies of non-stationary system in long-term predictions, they made the accuracies worse.

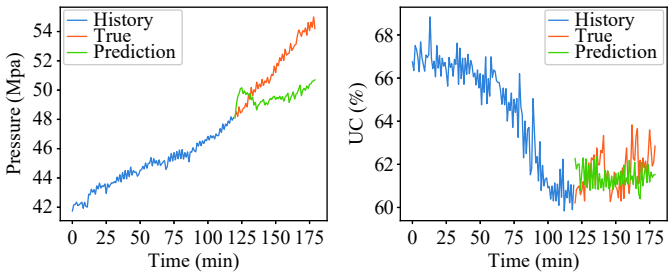
When the derivative module is switched to a stationary system. It is worth mentioning that the time consumption for the two adaptive methods, Bosh and Dopri5, to solve an ODE equation significantly increases. We do not list the accuracies of the Dopri5 and Bosh for the stationary systems in Table III because the extremely slow speed makes the method ineffective for practical applications. According to the comparison of the ODE solvers, the high-order ODE solver, such as RK4, results in lower fitting errors than the low-order methods while requiring more time to evaluate the ODE equations intensively.

c) Comparison of stationary models and non-stationary models:

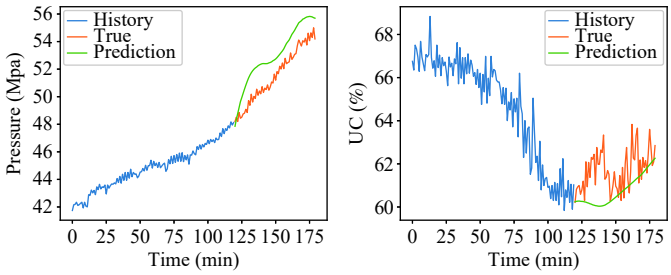
For comparing the distinctions between stationary models and non-stationary models more intuitively, we further visualize the prediction performance of the non-stationary and stationary systems with different ODE solvers. Fig. 5 depicts the predicted sequences of the non-stationary and stationary systems with different ODE solvers for the short-term prediction task with $L = 60$. The results show that the non-stationary models outperform the stationary models in short-term prediction tasks. The estimated sequences from the non-stationary models are slightly closer to the real system output than those from the stationary models. The learning process of a non-stationary system is essentially equivalent to differencing the hidden state and employing the MLP network to learn the relatively stationary first-order difference. Furthermore, the non-stationary models can predict the system outputs smoothly because the non-stationary structure limits

TABLE III
ROOT RELATIVE SQUARED ERROR (RRSE), MEAN SQUARED ERROR (MSE), AND TIME CONSUMPTION OF PREDICTED UNDERFLOW CONCENTRATION

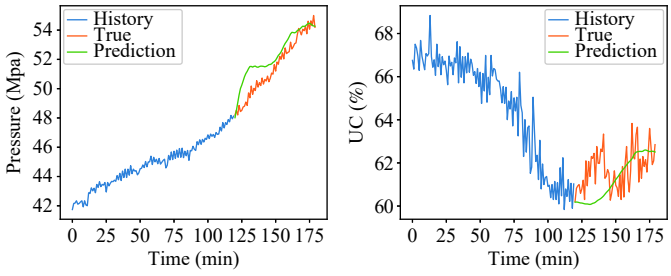
Models		$L = 60$ (120 min)			$L = 200$ (400 min)			$L = 500$ (1000 min)		
		RRSE	MSE	Time (s)	RRSE	MSE	Time (s)	RRSE	MSE	Time (s)
Non-stationary system	Euler	3.18	9.07	1.71	5.09	80.25	3.81	3.95	152.21	4.65
	Mid-Point	3.10	8.95	3.23	5.24	80.29	7.36	4.16	172.43	9.15
	RK4	3.10	8.97	6.95	5.24	83.90	14.82	4.16	172.64	18.76
	Bosh	3.08	8.57	12.8	5.84	84.60	19.0	4.61	172.39	24.75
	Dopri5	**2.83**	**6.40**	9.63	5.31	84.60	13.8	4.19	175.39	25.75
Stationary system	Euler	3.18	9.06	1.63	3.75	34.78	3.58	1.63	37.77	4.66
	Mid-Point	3.18	9.08	3.22	3.73	34.64	7.17	1.62	38.36	9.3
	RK4	3.18	8.96	6.80	**3.58**	**32.90**	15.17	**1.61**	**34.88**	18.66
	Bosh	N/A	N/A	>50	N/A	N/A	>200	N/A	N/A	>3000
	Dopri5	N/A	N/A	>50	N/A	N/A	>200	N/A	N/A	>3000
Attention-Seq2Seq [5]		3.13	8.97	0.41	4.02	33.90	0.41	1.82	40.53	0.42
DT-State-Space [45]		3.22	9.36	0.06	4.69	41.11	0.07	3.35	45.64	0.08
Transformer [44]		3.16	8.36	0.02	3.99	40.23	0.02	2.55	44.23	0.03



(a) Stationary system with fourth-order Runge-Kutta (RK4) ODE solver



(b) Non-stationary system with RK4 ODE solver



(c) Non-stationary system with Dormand-Prince (Dopri5) ODE solver

Fig. 5. In the short-term prediction task with $L = 60$, the non-stationary models output more stable and accurate sequences than the stationary models.

the hidden states to only changing in a continuous and slow manner. This constraint is consistent with the properties of a slow thickening system that shrinks the searching space of the model parameters to prevent overfitting.

Fig. 6 presents the experimental results of a long-term prediction task with $L = 200$ (similar results with $L = 500$ can be found in Table III). The tabular results in Table III demonstrate that the RRSE and MSE of non-stationary models are much higher than those of the stationary ones in long-term prediction task, which is consistent with the graphical results. Compared to the excellent results of non-stationary models in Fig. 5, Fig. 6(a) shows that the prediction accuracy for the non-stationary system decays significantly, and the predicted outputs deviate from the true outputs gradually with the increase in the prediction length. However, the predicted results of stationary models are stabilized and closed to the true system outputs, which confirms the excellent accuracies of stationary models in the long-term prediction problem. The structure of non-stationary ODE leads to the hidden state in progressive evolution that is unconstrained and gradually expanding. Although we embed a tanh function for the decoder network to restrict the final prediction of the underflow concentration and pressure to a rational range, it is impossible for the decoder module to learn an effective mapping function from an extremely large hidden state space to the system output space. Similarly, Fig. 6 also demonstrates that high-order ODE solvers, such as the 4th-order Runge-Kutta, still perform slightly better than the low-order solvers, such as Euler, in long-term prediction.

We conduct five other groups of experiments with different values of the prediction length to evaluate the prediction performance (i.e., the MSE) of the underflow concentration and ground-truth for both stationary and non-stationary systems. The results in Fig. 7 show that the non-stationary system performs better than the stationary system in the short-term prediction task (e.g., $L < 100$), although the stationary system outperforms the non-stationary system in long-term prediction tasks. For example, when L exceeds 120, the errors from the non-stationary system increase with the predicted

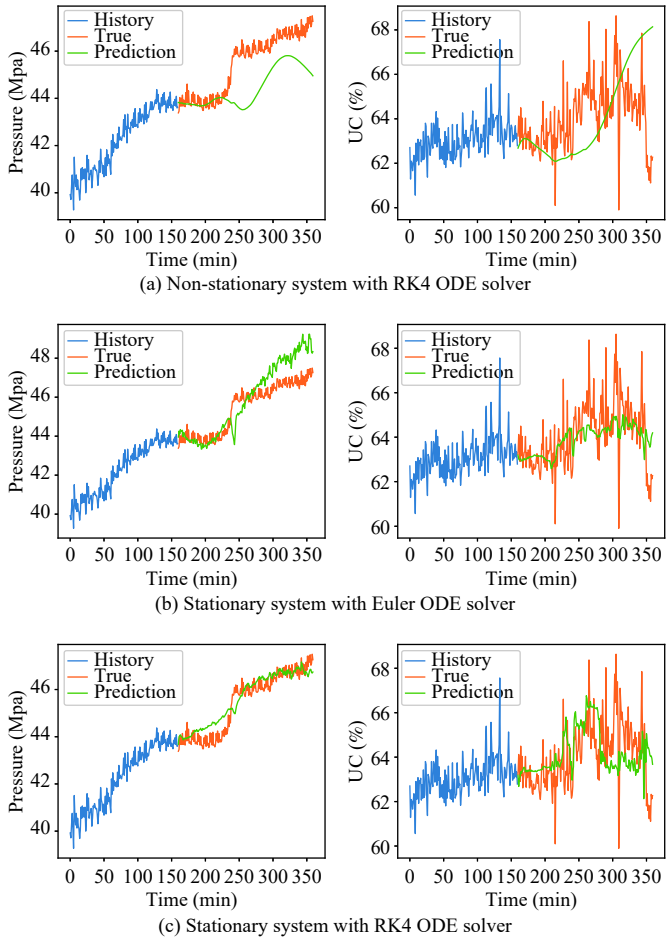


Fig. 6. In the long-term prediction task with $L = 200$, Fig. 6(a) illustrates that non-stationary models only performed well in the early time horizon. In the late horizon, the predicted sequences deviated from the true system outputs significantly compared with those of the stationary models, which are shown in Figs. 6(b) and 6(c). The results also indicate that the models with high-order ODE solvers performed better than the models with low-order ODE solvers in long-term prediction tasks.

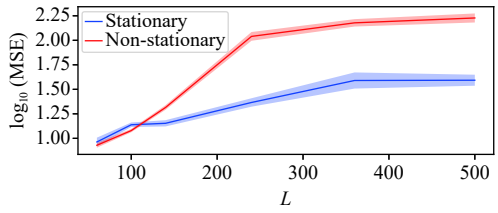


Fig. 7. Predicted length L affected the accuracy ($\log_{10}$ MSE $\pm 2\sigma$, computed across five runs) of predicted underflow concentration for both stationary and non-stationary systems.

length, while the stationary system significantly stabilizes the accumulative errors in the long-term prediction.

2) Experiments for Evaluating Interpolation Order

We next investigate the effect of the interpolation method on the prediction accuracy. We test four spline interpolation methods with different orders and compare the prediction accuracies on test datasets. The results in Table IV demonstrate that the higher-order interpolations slightly outperform the lower-order ones. This proves that the system input of the thickening system is a non-linear complex process, and that the information from external inputs is essential for predicting the outputs of the system. Higher-order spline interpolations exploit more correlational features from adjacent inputs and interpolate the empty area with a better accuracy than the lower-order interpolations.

3) Ablation Experiments for Studying System Time Delays and Improvements From Sequential Encoder

Finally, we investigate the significance of introducing the sequential encoder to confront the system time delay. We investigate the influence of N on the model accuracy. Specially, when N is set to 1, the sequential encoder is replaced by a neural network with one hidden layer that encodes the system output $\mathbf{x}(k-1)$ in a single time step to the initial hidden state $\mathbf{h}(t_0)$. When N is set to 0, the initial state $\mathbf{h}(t_0)$ is a learnable or zero vector [19] that had no relationship with historical system trajectories. We examine the different choices of N in experiments with $L = 60$, 200, and 500, respectively. In the experiments with $L = 60$, the derivative module is set to be a non-stationary system with an MLP cell. We change it to a stationary system with a GRU cell when $L = 200$ and 500. The ODE solver is the fourth-order Runge-Kutta solver for all of the models.

The results shown in Table V demonstrate that the introduction of the sequential encoder to extract features from the historical sequence leads to better performance than those in the cases with $N = 1$ or 0. The intuitive explanation is that the predicted output sequences have strong statistical correlations with historical system trajectories. The optimal length of the encoded sequence is approximately $N = 80$, which is consistent with our prior experience of 2–3-h time delays in thickening systems. When the length of input sequence exceeds the optimal value, the accuracy slightly decreases.

Intuitively, the short-term prediction task benefits more from historical system trajectories than the long-term prediction task. When the length of the predicted sequence increases, the advantage brought by the sequential encoder also decreases. In the task with $L = 500$, the profit of employing sequential encoder decreases obviously.

VI. Conclusions

This paper focuses on the prediction of the outputs of a thickening system based on deep neural sequence models. We introduce a CT network composed of a sequential encoder, a state decoder, and a derivative module, with internal computation processes including interpolation and a differential ordinary differential equation solver, to describe the complex dynamics of a thickening system. Experiments on datasets from real thickening systems demonstrate that the introduction of the sequential encoder and parallel cubic spline interpolation play a crucial role in our model architecture. We conducted extensive experiments to evaluate the proposed models for both stationary and non-stationary systems with different ODE solvers. The results show that the non-stationary system outperforms the stationary system for short-term prediction tasks. However, the non-stationary

TABLE IV
ACCURACY COMPARISONS WITH DIFFERENT ORDERS OF INTERPOLATIONS

Models	$L = 60$ (120 min)		$L = 200$ (400 min)		$L = 500$ (1000 min)	
	RRSE	MSE	RRSE	MSE	RRSE	MSE
Cubic	**3.083**	**8.565**	**3.581**	32.90	1.615	**34.88**
Quadratic	3.097	8.993	3.593	**32.585**	**1.613**	36.741
Slinear	3.098	8.999	3.763	33.530	1.627	37.778
Zero	3.115	9.050	3.791	33.585	1.628	37.695

TABLE V
ACCURACY COMPARISONS WITH DIFFERENT METHODS FOR GENERATING THE INITIAL HIDDEN STATE $\mathbf{h}(t_0)$

N	$L = 60$ (120 min)		$L = 200$ (400 min)		$L = 500$ (1000 min)	
	RRSE	MSE	RRSE	MSE	RRSE	MSE
160	3.11	9.08	**3.56**	34.13	1.61	35.88
80	**3.10**	**8.97**	3.58	**32.92**	**1.61**	**34.88**
40	3.19	8.99	3.65	36.07	1.71	41.26
1	4.06	10.71	4.97	51.09	1.77	63.56
$N = 0$ with learnable $\mathbf{h}(t_0)$	5.26	20.68	4.84	58.68	1.77	63.91
$N = 0$ with $\mathbf{h}(t_0) = \mathbf{0}$	5.26	23.11	5.84	64.49	1.77	63.53

model suffers from the accumulation of errors from the incremental calculation, thereby leading to inferior results in long-term prediction tasks. This demonstrates that the model with the non-stationary system is more suitable for being embedded in a model-based feedback controller (e.g., MPC controller) while the stationary system avoids this problem and performs better in long-term prediction tasks. Therefore, the model with the stationary system is a better choice when a stable and robust identified system is required to predict long-term sequences (e.g., simulations or controller testing).

In the industrial data processing field, it is a common requirement to process unevenly spaced data. Although the dataset employed in this paper is sampled evenly, we can extend our method to deal with uneven data naturally by adjusting the time intervals. This extension deserves further experimental verification in future work. Another promising research direction is to extend the method to probabilistic generative models and perturbed time-varying models [46] for determining the unknown sampling noise and uncertainty in thickening systems. Moreover, it is worth investigating our method for other dynamical industrial systems.

REFERENCES

[1] S. Yin, Y. Shao, A. Wu, H. Wang, X. Liu, and Y. Wang, "A systematic review of paste technology in metal mines for cleaner production in china," *Journal of Cleaner Production*, vol. 247, Article No. 119590, 2020.

[2] Z.-L. Yuan, R.-Z. He, C. Yao, J. Li, X.-J. Ban, and X.-R. Li, "Online reinforcement learning control algorithm for concentration of thickener underflow," *Acta Automatica Sinica*, vol. 45, pp. 1–15, 2019.

[3] H. Li, A. Wu, and H. Wang, "Evaluation of short-term strength development of cemented backfill with varying sulphide contents and the use of additives," *Journal of Environmental Management*, vol. 239, pp. 279–286, 2019.

[4] C. K. Tan, J. Bao, and G. Bickert, "A study on model predictive control in paste thickeners with rake torque constraint," *Minerals Engineering*, vol. 105, pp. 52–62, 2017.

[5] F. Núnez, S. Langarica, Díaz, M. Torres, and J. C. Salas, "Neural network-based model predictive control of a paste thickener over an industrial internet platform," *IEEE Trans. Industrial Informatics*, vol. 16, no. 4, pp. 2859–2867, 2019.

[6] Z. Yuan, J. Hu, D. Wu, and X. Ban, "A dual-attention recurrent neural network method for deep cone thickener underflow concentration prediction," *Sensors* (*Switzerland*), vol. 20, no. 5, pp. 1–18, 2020.

[7] A. Wu, Z. Ruan, R. Bürger, S. Yin, J. Wang, and Y. Wang, "Optimization of flocculation and settling parameters of tailings slurry by response surface methodology," *Minerals Engineering*, vol. 156, Article No. 106488, 2020.

[8] E. K. Larsson and T. Söderström, "Identification of continuous-time AR processes from unevenly sampled data," *Automatica*, vol. 38, no. 4, pp. 709–718, 2002.

[9] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Computational Intelligence and Neuroscience*, vol. 2018, PP. 1–13, 2018.

[10] B. Ma, X. Wei, C. Liu, X. Ban, H. Huang, H. Wang, W. Xue, S. Wu, M. Gao, Q. Shen, "Data augmentation in microscopic images for material data mining," *NPJ Computational Materials*, vol. 6, no. 1, pp. 1–9, 2020.

[11] B. Ma, Y. Zhu, X. Yin, X. Ban, H. Huang, and M. Mukeshimana, "Sesffuse: An unsupervised deep model for multi-focus image fusion," *Neural Computing and Applications*, vol. 33, no. 11, pp. 5793–5804, 2021.

[12] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *IEEE Computational Intelligence Magazine*, vol. 13, no. 3, pp. 55–75, 2018.

[13] H. Liu, I. Chatterjee, M. Zhou, X. S. Lu, and A. Abusorrah, "Aspectbased sentiment analysis: A survey of deep learning methods," *IEEE Trans. Computational Social Systems*, vol. 7, no. 6, pp. 1358–1375, 2020.

[14] A. E. Essien and C. Giannetti, "A deep learning model for smart manufacturing using convolutional LSTM neural network autoencoders," *IEEE Trans. Industrial Informatics*, vol. 16, no. 9, pp. 6069–6078, 2020.

[15] J. Liu, N. Wu, Y. Qiao, and Z. Li, "Short-term traffic flow forecasting

using ensemble approach based on deep belief networks," *IEEE Trans. Intelligent Transportation Systems*, pp. 1–14, 2020. DOI: 10.1109/TITS.2020.3011700

[16] J. Fei and L. Liu, "Real-time nonlinear model predictive control of active power filter using self-feedback recurrent fuzzy neural network estimator," *IEEE Trans. Industrial Electronics*, pp. 1–1, 2021. DOI: 10.1109/TIE.2021.3106007

[17] D. A. Neu, J. Lahann, and Fettke, "A systematic literature review on state-of-the-art deep learning methods for process prediction," *Artificial Intelligence Review*, pp. 1–27, 2021. DOI: 10.1007/s10462-021-09960-8

[18] H. Li, G. Hu, J. Li, and M. Zhou, "Intelligent fault diagnosis for large-scale rotating machines using binarized deep neural networks and random forests," *IEEE Trans. Automation Science and Engineering*, pp. 1–11, 2021. DOI: 10.1109/TASE.2020.3048056

[19] T. Demeester, "System identification with time-aware neural sequence models," in *Proc. AAAI Conf. Artificial Intelligence*, vol. 34, no. 4, 2020, pp. 3757–3764.

[20] C. K. Tan, R. Setiawan, J. Bao, and G. Bickert, "Studies on parameter estimation and model predictive control of paste thickeners," *Journal of Process Control*, vol. 28, pp. 1–8, 2015.

[21] J. I. Langlois and A. Cipriano, "Dynamic modeling and simulation of tailing thickener units for the development of control strategies," *Minerals Engineering*, vol. 131, pp. 131–139, 2019.

[22] X. Luo, M. Zhou, S. Li, and M. Shang, "An inherently nonnegative latent factor model for high-dimensional and sparse matrices from industrial applications," *IEEE Trans. Industrial Informatics*, vol. 14, no. 5, pp. 2011–2022, 2017.

[23] X. Luo, M. Zhou, S. Li, Y. Xia, Z.-H. You, Q. Zhu, and H. Leung, "Incorporation of efficient second-order solvers into latent factor models for accurate prediction of missing qos data," *IEEE Transactions on Cybernetics*, vol. 48, no. 4, pp. 1216–1228, 2017.

[24] D. Wu, X. Luo, M. Shang, Y. He, G. Wang, and X. Wu, "A datacharacteristic-aware latent factor model for web services QoS prediction," *IEEE Trans. Knowledge and Data Engineering*, 2020.

[25] M. Kebria, A. Khosravi, S. M. Salaken, and S. Nahavandi, "Deep imitation learning for autonomous vehicles based on convolutional neural networks," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 1, pp. 82–95, 2020.

[26] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, 2017.

[27] T.-Y. Chai, "Development directions of industrial artificial intelligence," *Acta Automatica Sinica*, vol. 46, no. 10, pp. 2005–2012, 2020.

[28] L. Jay, L. Xiang, X. Yuan-Ming, Y. Shaojie, and S. Ke-Yi, "Recent advances and prospects in industrial AI and ap-plications," *Acta Automatica Sinica*, vol. 46, no. 10, pp. 2031–2044, 2020.

[29] D. Wu, H. Wang, and R. Seidu, "Collaborative analysis for computational risk in urban water supply systems," in *Proc. 28th ACM Int. Conf. Information and Knowledge Management*, 2019, pp. 2297–2300.

[30] D. Wu, H. Wang, H. Mohammed, and R. Seidu, "Quality risk analysis for sustainable smart water supply using data perception," *IEEE Trans. Sustainable Computing*, vol. 5, no. 3, pp. 377–388, 2020.

[31] J. Zhou, H.-N. Dai, and H. Wang, "Lightweight convolution neural networks for mobile edge computing in transportation cyber physical systems," *ACM Trans. Intelligent Systems and Technology* (*TIST*), vol. 10, no. 6, pp. 1–20, 2019.

[32] H.-N. Dai, R. C.-W. Wong, H. Wang, Z. Zheng, and A. V. Vasilakos, "Big data analytics for large-scale wireless networks: Challenges and opportunities," *ACM Comput. Surv.*, vol. 52, no. 5, pp. 99:1–99:36, 2019.

[33] R. Oulhiq, K. Benjelloun, Y. Kali, and M. Saad, "Identification and control of an industrial thickener using historical data," in *Proc. 18th Int. Multi-Conf. on Systems, Signals Devices (SSD)*, 2021, pp. 915–920.

[34] Di az, J. C. Salas, A. Cipriano, and F. Núnez, "Random forest model? Predictive control for paste thickening" *Minerals Engineering*, vol. 163, Article No. 106760, 2021.

[35] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," in *Proc. Advances in Neural Information Processing Systems 31*, Curran Associates, Inc., 2018, pp. 6571–6583.

[36] R. J. Weiss, J. Chorowski, N. Jaitly, Y. Wu, and Z. Chen, "Sequence-to-sequence models can directly translate foreign speech," in *Proc. Interspeech 2017*, pp. 2625–2629, 2017.

[37] M. GEVERS, "A personal view of the development of system identification: A 30-year journey through an exciting field," *IEEE Control Systems Magazine*, vol. 26, no. 6, pp. 93–105, 2006.

[38] T. Chai, Y. Jia, H. Li, and H. Wang, "An intelligent switching control for a mixed separation thickener process," *Control Engineering Practice*, vol. 57, pp. 61–71, 2016.

[39] B. Kim and M. S. Klima, "Development and application of a dynamic model for hindered-settling column separations," *Minerals Engineering*, vol. 17, no. 3, pp. 403–410, 2004.

[40] M. Guidolin and M. Pedio, "Chapter 4-unit roots and cointegration," in *Essentials of Time Series for Financial Applications*, M. Guidolin and M. Pedio, Eds. USA: Academic Press, 2018, pp. 113–149.

[41] F. Christoffersen, "Forecasting non-stationary economic time series," *Journal of the American Statistical Association*, vol. 96, no. 453, pp. 347–347, 2001.

[42] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv: 1412.6980*, 2014.

[43] Bogacki and L. F. Shampine, "A 3(2) pair of Runge-Kutta formulas," *Applied Mathematics Letters*, vol. 2, no. 4, pp. 321–325, 1989.

[44] S. Wu, X. Xiao, Q. Ding, P. Zhao, Y. Wei, and J. Huang, "Adversarial sparse transformer for time series forecasting," *Advances in Neural Information Processing Systems*, vol. 33, no. NeurIPS, pp. 17105–17115, 2020.

[45] S. S. Rangapuram, M. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski, "Deep state space models for time series forecasting," *Advances in Neural Information Processing Systems*, vol. 2018-Decem, no. NeurIPS, pp. 7785–7794, 2018.

[46] H. Lu, L. Jin, X. Luo, B. Liao, D. Guo, and L. Xiao, "RNN for solving perturbed time-varying underdetermined linear system with double bound limits on residual errors and state variables," *IEEE Trans. Industrial Informatics*, vol. 15, no. 11, pp. 5931–5942, 2019.

Zhaolin Yuan received the B.S. degree in computer science from University of Science and Technology Beijing, China, in 2017. He is now a Ph.D. candidate at the School of Computer and Communication Engineering, University of Science and Technology Beijing. His current research interests include complex industrial system, time series prediction and modeling, reinforcement learning.

Xiaorui Li received the B.E. degree from the Department of Information and Communication Engineering, University of Science and Technology Beijing, China in 2020. Currently he is a Ph.D. candidate at University of Science and Technology Beijing, China. His research interests include time series forecasting, deep learning and process control.

Di Wu received the B.S. degree from the Department of Automation, Beijing Institute of Technology, China in 2004, the Ph.D. degree in pattern recognition and intelligent system from the School of Automation, Beijing Institute of Technology in 2010. She worked as Post-doc/Lecturer with the Department of Computer and Communication Engineering, the University of Science and Technology Beijing, China. Currently she is a Ph.D. candidate in data science at Norwegian University of Science and Technology. Her research interests include large scale composable simulation system, artificial life, and data analysis in industrial applications such as water supply systems. She has already published over 35 papers for international journals and conferences.

Xiaojuan Ban received the Ph.D. degree from the University of Science and Technology Beijing, Beijing, in 2003. She is currently a Ph.D. Supervisor with the University of Science and Technology Beijing (USTB). Her current research interests include artificial intelligence, natural human-computer interactions, and 3D visualization. She has authored more than 300 articles. She is also the Managing Director of the Chinese Association for Artificial Intelligence (CAAI). She is also a member of the standing committee of the human-computer interaction specialty and the theoretical computer science specialty in the China Computer Society (CCF). She has received the New Century Excellent Talent of the Ministry of Education.

Nai-Qi Wu (Follow, IEEE) received the B.S. degree in electrical engineering from Anhui University of Technology, China, in 1982, the M.S. and Ph.D. degrees in systems engineering both from Xi'an Jiaotong University, Xi'an, China in 1985 and 1988, respectively. From 1988 to 1995, he was with Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang, China, and from 1995 to 1998, with Shantou University, Shantou, China. He moved to Guangdong University of Technology, Guangzhou, China in 1998. He joined Macau University of Science and Technology, Taipa, China in 2013. He was a Visiting Professor at Arizona State University, Tempe, USA, in 1999; New Jersey Institute of Technology, Newark, USA, in 2004; University of Technology of Troyes, Troyes, France, from 2007 to 2009; and Evry University, Evry, France, from 2010 to 2011. He is currently a Chair Professor at the Institute of Systems Engineering, Macau University of Science and Technology, Taipa, China. His research interests include production planning and scheduling, manufacturing system modeling and control, discrete event systems, Petri net theory and applications, intelligent transportation systems, and energy systems. He is the Author or Coauthor of one book, five book chapters, and 180+ peer-reviewed journal papers. He was an Associate Editor of the *IEEE Transactions on Systems*, *Man*, *& Cybernetics*, *Part C*, *IEEE Transactions on Automation Science and Engineering*, *IEEE Transactions on Systems*, *Man*, *& Cybernetics*: *Systems*, and Editor in Chief of *Industrial Engineering Journal*, and is an Associate Editor of *Information Sciences*.

Hong-Ning Dai (Senior Member, IEEE) received the Ph.D. degree in computer science and engineering from Department of Computer Science and Engineering, the Chinese University of Hong Kong. He is currently with Department of Computing and Decision Sciences at Lingnan University, Hong Kong, China as an Associate Professor. His current research interests include the internet of things, big data, and blockchain technology. He has served as Associate Editors/Editors for *IEEE Transactions on Industrial Informatics*, *IEEE Systems Journal*, *IEEE Access*, *Ad Hoc Networks*, and *Connection Science*. He is also a Senior Member of Association for Computing Machinery (ACM).

Hao Wang (Senior Member, IEEE) received the B.Eng. and Ph.D. degrees, both in computer science and engineering, from South China University of Technology, China in 2000 and 2006, respectively. He is an Associate Professor in the Department of Computer Science, Norwegian University of Science & Technology, Norway. His research interests include big data analytics, industrial internet of things, high performance computing, and safety-critical systems. He has published 170+ papers in reputable international journals and conferences including *IEEE Transactions on Knowledge and Data Engineering, IEEE Transactions on Industrial Electronics, Transactions on Industrial Informatics, Internet of Things Journal, Transactions on Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on Geoscience and Remote Sensing* and *ACM Computing Surveys*. He serves as the Editorial Board Member and Guest Editors for several international journals. He served as a TPC Co-Chair for *IEEE Transactions on Computational Social Systems 2020, IEEE International Conference on Computer and Information Technology 2017, International Conference on Big Data Intelligence and Computing 2015*, and Reviewers for many prestigious journals and conferences. He is a Senior Member of IEEE and a Member of ACM. He is the Chair for Sub TC on Healthcare in IEEE IES Technical Committee on Industrial Informatics.