Proceedings of Machine Learning Research vol 144:1–11, 2021

# Decoupling dynamics and sampling: RNNs for unevenly sampled data and flexible online predictions

**Signe Moe*** SIGNE.MOE@SINTEF.NO
**Camilla Sterud*** CAMILLA.STERUD@SINTEF.NO
*Dept. of Mathematics and Cybernetics, SINTEF Digital, Oslo, Norway*
**These authors have contributed equally to this work*

## Abstract

Recurrent neural networks (RNNs) incorporate a memory state which makes them suitable for time series analysis. The Linear Antisymmetric RNN (LARNN) is a previously suggested recurrent layer which is proven to ensure long-term memory using a simple structure without gating. The LARNN is based on an ordinary differential equation which is solved using numerical methods with a defined step size variable. In this paper, this step size is related to the sampling frequency of the data used for training and testing of the models. In particular, industrial datasets often consist of measurements that are sampled and analyzed manually or sampled only for sufficiently large changes. This is usually handled by resampling and interpolating to gain a dataset with evenly sampled data. However, in doing so, one has to apply several assumption regarding the nature of the data (e.g. linear interpolation) and valuable information about the dynamics captured by the actual sampling is lost. Furthermore, interpolation is non-causal by nature, and thus poses a challenge in an online setting as future values are not known. By using information about sampling time in the LARNN structure, interpolation is obsolete as the model decouples the dynamics of the sampled system from the sampling regime. Furthermore, the suggested structure enables predictions related to specific times in the future, resulting in updated predictions regardless of whether new measurements are available. The performance of the LARNN is compared to an LSTM on a simulated industrial benchmark system.

**Keywords:** Recurrent Neural Network, Time-series Analysis, Unevenly Sampled Data

## 1. Introduction

Recurrent neural networks (RNNs) are commonly used for modeling sequential data Goodfellow et al. (2016). Although standalone RNNs have fallen somewhat out of fashion in the recent years, they are still keystones in modern architectures, such as models with attention mechanisms Graves et al. (2014); Xu et al. (2015). Their ability to learn semi long-term dependencies is key in for instance language modeling Jozefowicz et al. (2016) and learning physical systems with slow dynamics. The hidden state in a vanilla RNN is defined as

$$\mathbf{h}_t = f(\mathbf{W}\mathbf{h}_{t-1} + \mathbf{V}\mathbf{x}_t + \mathbf{b}), \tag{1}$$

where $\mathbf{h}_i$ is the output and $\mathbf{x}_i$ the input at time $i$, $\mathbf{W}$ and $\mathbf{V}$ are weight matrices, $\mathbf{b}$ is a bias vector and $f(\cdot)$ is a nonlinear activation function. Hence, an RNN generates an output trajectory $\begin{bmatrix}\mathbf{h}_1 & \mathbf{h}_2 & \dots & \mathbf{h}_N\end{bmatrix}^T$ from an initial condition $\mathbf{h}_0$ and an input sequence $\begin{bmatrix}\mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_N\end{bmatrix}^T$. A wide variety of RNNs have been suggested, where the best known are gated recurrent units (GRUs) and long-short-term memory networks (LSTMs).

These RNN variations aim to handle what is known as the exploding and vanishing gradient problem with vanilla RNNs. Large and small gradients indicate oversensitivity and insensitivity to changes in the input and initial condition of the RNN, respectively. Oversensitivity is undesirable as it can lead to input-output instability, while insensitive systems will yield similar trajectories in spite of changes in input and initial condition.

The increasing interest in applying deep learning RNNs for dynamical system and control problems has prompted research on their stability properties and other theoretical guarantees. In Moe et al. (2020) a new recurrent structure referred to as Linear Antisymmetric RNN (LARNN) was presented. In short, the LARNN is based on an ODE which is marginally stable, thereby ensuring what can be considered a consistent, long-term memory. Further details are presented in Section 2.

Relevant to this work are Neural ODEs Chen et al. (2018) and related follow up works Dupont et al. (2019). Here, the the derivative of the hidden state is parameterized using a neural network and the output of the network is computed using a black-box differential equation solver. However, Neural ODEs are not designed to handle systems that are affected by an independent time sequence input. Furthermore, Neural ODEs are suitable for solving continuous time problems, but it is unclear whether they would be a wise choice when dealing with systems subject to conservative sampling regimes.

The contribution of this paper is an extension to the previously suggested LARNN structure to handle unevenly sampled input sequences, which is common for industrial datasets consisting of manual measurements or conservative logging systems. This recurrent regression model is presented in Section 3. These extensions are tested on a simulated industrial benchmark process and compared to an LSTM. Results and discussions are presented in Section 4 and Section 5, respectively. Finally, conclusions are given in Section 6.

## 2. Linear Antisymmetric RNNs

In Moe et al. (2020), the authors suggest a recurrent network structure based the following linear-in-the-state ordinary differential equation (ODE):

$$\dot{\mathbf{h}} = \mathbf{W}_h \mathbf{h} + f(\mathbf{V}\mathbf{x} + \mathbf{b}), \quad \mathbf{W}_h = \mathbf{W} - \mathbf{W}^T \tag{2}$$

Here, $\mathbf{W}_h$ is an antisymmetric matrix, which has eigenvalues on the imaginary axis and a complete set of eigenvectors. As shown in Moe et al. (2020), (2) is stable, such that solutions of the ODE with different initial conditions, but subject to the same input sequence, will neither diverge from each other, nor converge into one another, thereby infinitely conserving the initial condition. This also holds when the solutions have the same initial condition, but are subject to input sequences that differ only in the first time step.

The ODE (2) can be numerically solved using implicit numerical methods, and as the ODE is linear in the state $\mathbf{h}$, the resulting equations can be solved explicitly. In particular, the backward Euler method and the implicit midpoint method are numerically stable for all $\mathbf{W}_h$ and step size $\varepsilon > 0$. Consequently, these methods inspire the following RNN structures:

$$(I - \varepsilon \mathbf{W}_h)\mathbf{h}_{t+1} = \mathbf{h}_t + \varepsilon f(\mathbf{V}\mathbf{x}_t + \mathbf{b}) \tag{3}$$

$$\left(I - \frac{\varepsilon}{2}\mathbf{W}_h\right)\mathbf{h}_{t+1} = \left(I + \frac{\varepsilon}{2}\mathbf{W}_h\right)\mathbf{h}_t + \varepsilon f\left(\mathbf{V}\mathbf{x}_t + \mathbf{b}\right) \tag{4}$$

As demonstrated in Moe et al. (2020), the LARNNs perform better than, or as well as, LSTMs on simulated data, while having stronger stability guarantees. For practical purposes, this ensures a long term memory using a simple network structure without gating.

The step size $\varepsilon$ is a parameter in the numeric integration of (2) and can be viewed as a hyper-parameter which weighs the effects of previous versus current inputs. In Moe et al. (2020), this parameter is tuned in line with other hyperparameters. However, another possible interpretation is that $\varepsilon$ represents the time step in the integration. Backward Euler and the implicit midpoint method are traditionally applied with a fixed step size $\varepsilon$, but other commonly used numerical ODE-solvers such as Dormand-Prince and Adam can adjust the step size throughout the calculation to ensure that the per-step error remains at a given relative and absolute tolerance. Based on this, we suggest an extension to the LARNN structures (3)-(4) suited for analysis of time sequences which are sampled at varying intervals by relating the step size $\varepsilon$ to sampling time.

## 3. Variable step size

Industry data is often sampled in uneven intervals, either because measurements are performed manually by operators or because only sufficiently large changes are logged due to data storage considerations. This is usually handled by resampling and interpolating to gain a dataset with evenly sampled data. However, in doing so, one has to apply some assumption regarding the nature of the data (e.g. linear interpolation) and valuable information about the dynamics captured by the actual sampling is lost. In this work, we show how the LARNN can handle datasets with varying sampling frequency by introducing a varying, time-dependent step length $\varepsilon_t$. Furthermore, we demonstrate that the LARNNs can exploit the inherent information in such datasets to better learn the dynamics of the underlying system.

The step size $\varepsilon$ can be interpreted as how far into the future the ODE solver makes a prediction. As discussed in Moe et al. (2020), (3)-(4) are stable for all $\varepsilon > 0$, which means that the step size of the LARNNs can be adjusted to an appropriate size when making predictions. To accommodate for variable step size, the previously proposed LARNN structures (3) and (4) are modified as follows: In addition to the input sequence $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \dots & x_N \end{bmatrix}^T$, the LARNN is also fed a time step sequence:

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_2 & \varepsilon_3 & \dots & \varepsilon_{N+1} \end{bmatrix}^T = \begin{bmatrix} t_2 - t_1 & t_3 - t_2 & \dots & t_{N+1} - t_N \end{bmatrix}^T. \tag{5}$$

This yields the following, updated versions of (3) and (4):

$$(I - \varepsilon_{t+1} W_h)\mathbf{h}_{t+1} = \mathbf{h}_t + \varepsilon_{t+1} f(V\mathbf{x}_t + \mathbf{b}) \tag{6}$$

$$(I - \frac{\varepsilon_{t+1}}{2} W_h)\mathbf{h}_{t+1} = (I + \frac{\varepsilon_{t+1}}{2} W_h)\mathbf{h}_t + \varepsilon_{t+1} f(V\mathbf{x}_t + \mathbf{b}), \tag{7}$$

where $\varepsilon_i$ can be viewed as the time we are forecasting into the future based on the most recently sampled data. Thus, this modified network structure enables us to choose the exact prediction time at every time step. This approach is particularly interesting as it allows for updated, time-specific predictions ahead in time given the same input sequence, by choosing $\varepsilon_{N+1}$ freely. This approach is relevant for time series regression, where the goal is to predict some $\mathbf{y}_{N+1}(\mathbf{x}_1, \dots, \mathbf{x}_N)$, as well as time series forecasting, i.e. predict $\mathbf{x}_{N+1}$ from $\mathbf{x}_1, \dots, \mathbf{x}_N$. The last case is illustrated in Figure 1.

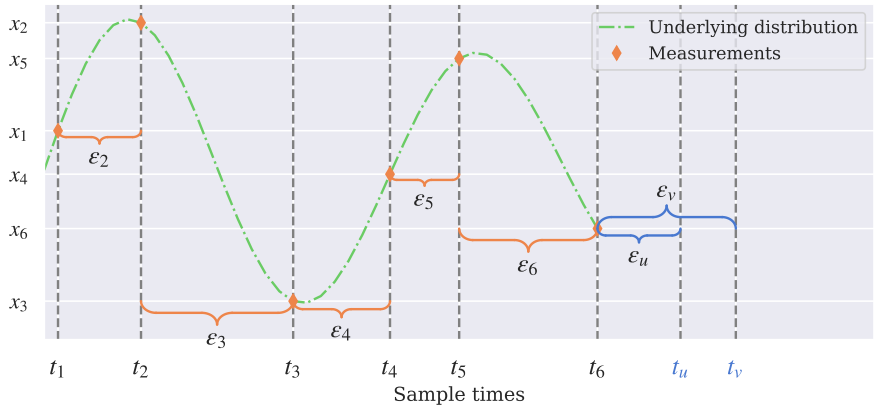



Figure 1: A signal $x$ is sampled at times $t_1$, $t_2$, ..., $t_6$ with varying sample time. The modified LARNN structures in (6) and (7) enable predictions related to specific times in the future, e.g. $t_u$ or $t_v$, based on the same previous measurement sequence $\begin{bmatrix} x_1 & x_2 & \dots & x_6 \end{bmatrix}^T$.

## 4. Implementation and Results

The LARNN structures with varying step size presented in the above section have been implemented in Tensorflow and can be used in line with existing Keras layers and architectures. This section presents results based on a simulated industrial benchmark process and compares training and performance of the LARNN and an LSTM.

### 4.1. Continuous Stirred Tank Reactor

The Continuous Stirred Tank Reactor (CSTR) is a simulated process which is often used as a benchmark system in the literature Maiworm et al. (2018); Manzano et al. (2018). It describes an industrial process of a chemical reaction where a reactant is changed from $A \rightarrow B$ Seborg et al. (2010). The CSTR is illustrated in Figure 2 and the dynamics are given in (8). Numeric values for the model parameters are summarized in Table 1.

The process has two states, namely the concentration of reactant A ($C_A$) and the temperature ($T$). The tank has a continuous inflow of concentration $C_{Af}$ and temperature $T_f$. Furthermore, the reaction in the tank can be affected through a cooling jacket of temperature $T_c$. Concentrations are given in [mol/l] and temperatures in [K]. Note that in the results presented here, $T_c$ is kept constant at 330 K.

$$\begin{aligned} \dot{C}_A(t) &= \frac{q_0}{V}\left(C_{Af}(t) - C_A(t)\right) - k_0 \exp\left(\frac{-E}{RT(t)}\right) C_A(t) \\ \dot{T}(t) &= \frac{q_0}{V}\left(T_f(t) - T(t)\right) - \frac{\Delta H_r k_0}{\rho C_p} \exp\left(\frac{-E}{RT(t)}\right) C_A(t) + \frac{UA}{V\rho C_p}\left(T_c(t) - T(t)\right) \end{aligned} \tag{8}$$

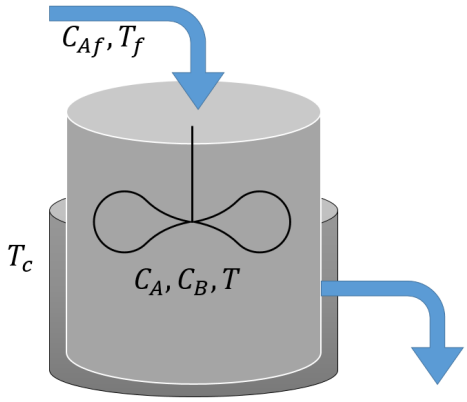



Figure 2: Illustration of the CSTR process and variables.

| Param. | Definition | Value |
|---|---|---|
| $q_0$ | Reactive input flow | 10 l/min |
| $V$ | Liquid volume in the tank | 150 l |
| $k_0$ | Frequency constant | $6{\cdot}10^{10}$ l/min |
| $E/R$ | Arrhenius constant | 9750 K |
| $-\Delta H_r$ | Reaction enthalpy | 10000 J/mol |
| $\rho$ | Density | 1100 g/l |
| $C_p$ | Spesific heat | 0.3 J/(gK) |
| $UA$ | Heat transfer coefficient | 70000 J/(minK) |
| $T_c$ | Temperature of tank cooling jacket | 330 K |

Table 1: CSTR process parameters as used in Maiworm et al. (2018); Manzano et al. (2018).

### 4.2. Simulation Data

Simulations of the CSTR process (8) form the datasets used in this paper. The entire data base consists of 200 individual simulations of the tank. Each simulation spans 50 hours and consists of 180 000 data points of the inflow variables $C_{Af}$ and $T_f$, and the corresponding tank variables $C_A$ and $T$. The initial condition of the tank variables are uniformly distributed according to $C_A(0) \in [0.3, 2]$ and $T(0) \in [335, 380]$. The inflow variables $C_{Af}$ and $T_f$ change throughout a simulation in a manner similar to a step response. The time between steps is given by a Gaussian distribution $\mathcal{N}(120 \text{ min}, 30 \text{ min})$ and the numeric values of the inflow variables are uniformly distributed within the same intervals as the initial condition of the tank. To incorporate a more realistic input, the step response is lowpass-filtered using a $4^{th}$ order Butterworth filter. Data for a single simulations is shown in Figure 3. Note that the first hour for each simulation is discarded to allow the Butterworth filter to stabilize.

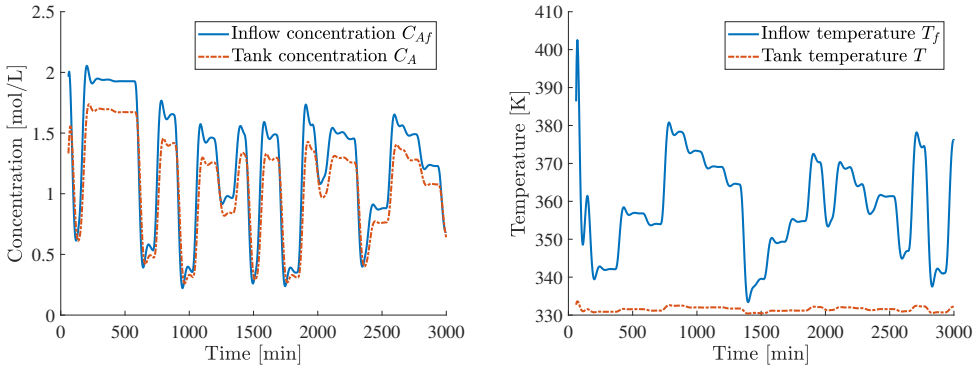



Figure 3: Example of a simulation of the CSTR with inflow variables in blue solid line and tank variables in red dashed line. Concentration to the left and temperature to the right. The inflow variables change according to a smoothened step response at a Gaussian distributed time interval.

### 4.3. Test Scenarios

Three datasets are created based on the simulated CSTR data. Dataset 1 consists of evenly sampled data with a fixed sampling period of 10 min, while in Dataset 2, the samples are taken unevenly anywhere between every 5 min and every 20 min. Dataset 3 is a linear interpolation of Dataset 2, which resamples the data to an even sampling period of 5 min. All datasets are divided into 70% training, 20% validation and 10% test data. The scaled input and output data $\in [0, 1]$ is denoted $\bar{C}_{Af}$, $\bar{T}_f$, $\bar{C}_A$ and $\bar{T}$.

All neural network models consist of a recurrent layer with 20 units, followed by a dense layer with 18 units and ReLU activation, and an output dense layer with two units and linear activation. In the experiments, the goal is to predict $\bar{C}_A$ and $\bar{T}$ at some time $t_{N+1}$, given an input sequence $\mathbf{x}_N = \left[\left[\bar{C}_{Af}(t_1) \quad \bar{T}_f(t_1)\right] \quad \left[\bar{C}_{Af}(t_2) \quad \bar{T}_f(t_2)\right] \quad \dots \quad \left[\bar{C}_{Af}(t_N) \quad \bar{T}_f(t_N)\right]\right]$. In the fixed step case (Dataset 1), $\varepsilon = t_2 - t_1 = t_i - t_{i-1} = 10$ min, while in the variable step case (Dataset 2),

$\boldsymbol{\varepsilon} = \begin{bmatrix}\varepsilon_2 & \varepsilon_1 & \dots & \varepsilon_{N+1}\end{bmatrix} = \begin{bmatrix}t_2 - t_1 & t_3 - t_2 & \dots & t_{N+1} - t_N\end{bmatrix}$. We refer to $N$ as the number of recurrent steps. Details about the training are summarized in Table 2. To evaluate the performance of the various network structures, each model is trained 10 times and the average mean squared error (MSE) is reported for the two scaled target variables $\bar{C}_A$ and $\bar{T}$.

Table 2: Training parameters for the LSTM and LARNN models. *All models were trained until the validation loss did not decrease for 20 epochs, at which point the model with the lowest validation loss was kept.

| Param. | Definition | Value |
|---|---|---|
| $f$ | Activation function | ReLU |
| $N$ | Number of recurrent steps | 10 |
| $L$ | Loss function | Mean squared error |
| $\gamma$ | Learning rate | $1 \times 10^{-3}$ |
| $N_B$ | Training batch size | 196 |
| $N_E$ | Training epochs* | 1000 |

First, an LSTM and the two LARNN models (3)-(4) are trained on Dataset 1, i.e with fixed sampling rate. The results in Table 3 confirm the observation from Moe et al. (2020), that the LSTM and LARNN perform quite equally well.

In the next experiment, the variable step LARNN models (6)-(7) are trained on the unevenly sampled data in Dataset 2. Since existing recurrent structures are designed with evenly sampled data in mind, an LSTM is trained on resampled, linearly interpolated data in Dataset 3 to ensure a fair comparison. When measuring the performance of the LSTM, only the predictions at the non-interpolated times are considered, i.e. the error is calculated based on actual samples and not interpolated values. Results are summarized in Table 4. Note that in order to emulate an online prediction setting during testing, linear interpolation is applied causally, such that the last input value, if not sampled, cannot be interpolated from a future input value. Instead, forward fill is applied to the last available sample. Thus, if at time $N$ a new sample is not available, the input sequence becomes $\mathbf{x}_N = \begin{bmatrix}\begin{bmatrix}\bar{C}_A(t_0) & \bar{T}(t_0)\end{bmatrix} & \dots & \begin{bmatrix}\bar{C}_A(t_{N-1}) & \bar{T}(t_{N-1})\end{bmatrix} & \begin{bmatrix}\bar{C}_A(t_{N-1}) & \bar{T}(t_{N-1})\end{bmatrix}\end{bmatrix}$.

One of the strengths of the LARNNs (6)-(7) is the ability to handle varying step lengths explicitly and exploit this to make predictions related to specific times. Thus, we explore their performance

Table 3: Test MSE for an LSTM and the LARNN models (3)-(4) trained on Dataset 1, i.e. on evenly sampled data. The error is an average of the error of 10 models for each model type.

| Test datset | Model | MSE $\bar{C}_A$ | MSE $\bar{T}$ |
|---|---|---|---|
| Dataset 1 | LSTM | $1.167 \times 10^{-5}$ | $2.469 \times 10^{-5}$ |
| | LARNN midpoint | $1.885 \times 10^{-5}$ | $4.080 \times 10^{-5}$ |
| | LARNN backward | $1.763 \times 10^{-5}$ | $4.325 \times 10^{-5}$ |
| Dataset 2 | LSTM | $8.490 \times 10^{-2}$ | $2.443 \times 10^{-2}$ |
| | LARNN midpoint | $6.663 \times 10^{-2}$ | $5.864 \times 10^{-2}$ |
| | LARNN backward | $1.747 \times 10^{-3}$ | $9.621 \times 10^{-4}$ |

Table 4: Test MSE for an LSTM trained on Dataset 3 and the LARNN models (6)-(7) trained on Dataset 2. The error is an average of the error of 10 models for each model type. *When testing the LSTM on Dataset 2, the sampled data is linearly interpolated causally, i.e. only based on samples taken up to the time of prediction. Forward fill is applied to the last available sample to emulate an online prediction scenario where future values are not known.

| Test dataset | Model | MSE $\bar{C}_A$ | MSE $\bar{T}$ |
|---|---|---|---|
| Dataset 1 | LSTM | $2.250 \times 10^{-3}$ | $5.086 \times 10^{-3}$ |
| | LARNN midpoint | $1.441 \times 10^{-4}$ | $1.970 \times 10^{-4}$ |
| | LARNN backward | $1.030 \times 10^{-4}$ | $1.630 \times 10^{-4}$ |
| Dataset 2 | LSTM* | $2.967 \times 10^{-3}$ | $1.663 \times 10^{-3}$ |
| | LARNN midpoint | $3.038 \times 10^{-4}$ | $4.203 \times 10^{-4}$ |
| | LARNN backward | $5.679 \times 10^{-4}$ | $5.582 \times 10^{-4}$ |

on Dataset 2 with the goal of making a prediction every $2\,\mathrm{min}$ in an online setting. During training, the LARNNs have only been exposed to time steps between $5\,\mathrm{min}$ and $20\,\mathrm{min}$. During testing, the prediction step length at the last recurrent step, $\varepsilon_{N+1}$, is changed such that a prediction is made every $2\,\mathrm{min}$ following an input measurement. A visualization of these results is shown in Figure 4.

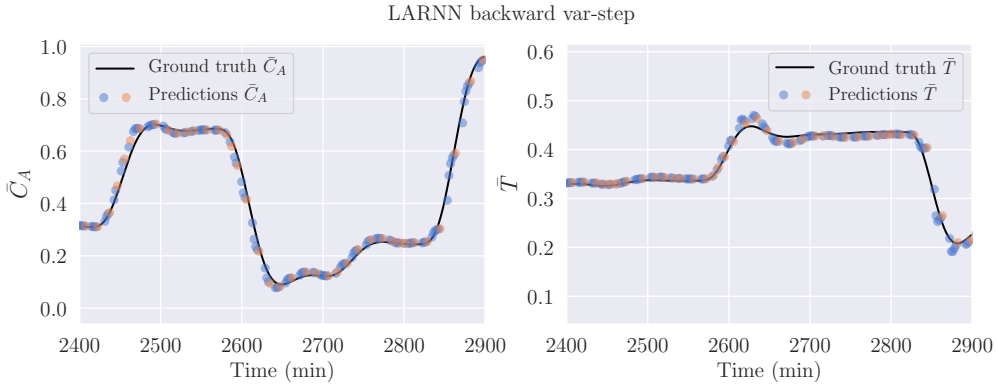
For comparison, we show the corresponding setup for the LSTM trained on Dataset 3. In this case, the neural network bases its prediction only on the input variables and not the corresponding sampling times. As when testing the LSTM on Dataset 3 before, we apply linear interpolation in a causal fashion. As the predictions can only be made every $5\,\mathrm{min}$, predictions are reused once or twice to achieve a resolution of one prediction every other minute, as shown in Figure 5.

## 5. Discussion

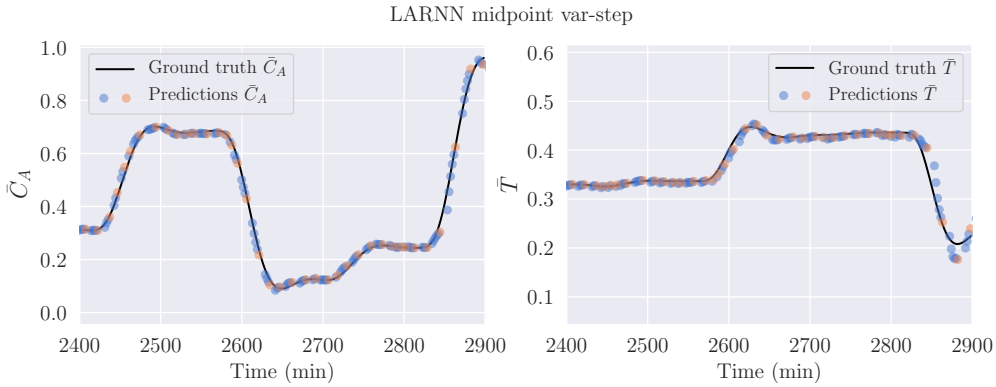
As seen in Table 3, both the LSTM and LARNN models trained on fixed step data experience a decrease in performance when tested on unevenly sampled data. This is particularly true for the midpoint LARNN. This indicates that when training on evenly sampled data, the models become more biased, as they do not differentiate between the sampling scheme and the underlying dynamics. In this sense, the LARNN with variable step length decouples the dynamics of the sampled system from the sampling regime, and hence, learns more of what we are actually interested in learning.

The results presented in the previous section prove that the LARNN architecture is flexible and has the ability to learn based on unevenly sampled data. Table 4 reveals that when the LARNN models are trained on unevenly sampled data, they are still capable of generalizing and making predictions on evenly sampled data, and in fact do so with a higher performance than on the unevenly sampled data. In particular, Figure 4 illustrates how the trained LARNNs are able to make predictions that reflect the behaviour of the underlying system as the prediction step length $\varepsilon_{N+1}$ is varied beyond what the model has seen during training. This shows that the models have learned the dynamics of the system and the effect of varying time between input samples.

In general, all models achieve a low mean squared error during testing. However, the LARNNs achieve lower mean squared errors on the unevenly sampled data than the LSTM trained on interpolated data. Interpolation and the chosen resolution for resampling introduces bias. In addition, time




(*a*) Backward LARNN (6)




(*b*) Midpoint LARNN (7)

Figure 4: A demonstration of online predictions on Dataset 2 made by the backward and midpoint LARNN, respectively. The scaled target variables $\bar{C}_A$ and $\bar{T}$ are shown in the black line and the blue and orange dots are the LARNN predictions. For every input sequence $\mathbf{x}_N$, we make predictions every 2 min into the future by increasing $\varepsilon_{N+1}$ by 2 at a time. Each orange dot marks the times where the input variables for the LARNN prediction model are sampled and $\mathbf{x}_N$ is updated. The LARNN is able to accurately predict the target variables ahead in time based on the last $N$ samples.

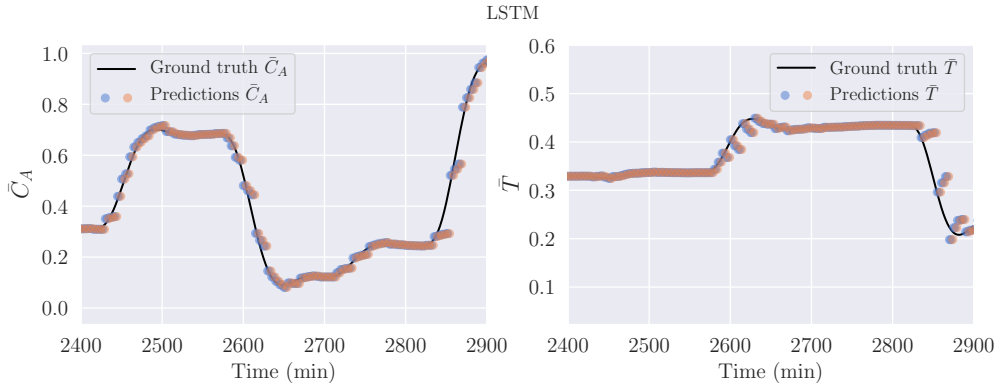



Figure 5: A demonstration of online predictions on Dataset 2 made by the LSTM trained on Dataset 3. As the LSTM cannot be reused for different step lengths, we resample the dataset with linear interpolation in a causal manner, such that it coincides with the $\varepsilon = 5\,\mathrm{m}$ the LSTM has seen during training. In addition, as $\varepsilon = 5\,\mathrm{min}$ and we are interested in making predictions every $2\,\mathrm{min}$, each prediction must be repeated once or twice in order to achieve the desired resolution.

series regression is often intended for online use, as in the case of a soft sensor, which sabotages the interpolation scheme. In such a case, one must default to reuse the last sample until a new sample can be made. The strength of the LARNN is not in achieving a low prediction error compared to an LSTM, but the ability to change its prediction based on the prediction time step.

## 6. Conclusion

The flexible structure of the LARNN enables learning on unevenly sampled data, without introducing the bias of interpolation and choice of resolution. In addition to using the sampled values, the LARNN uses the sampling times both during training and testing. This additional information enables higher generalization as is it independent of a particular sampling scheme. Furthermore, the LARNN models can be used for making predictions online, as the prediction time step can be varied freely. The LARNN achieves better or equal performance to the LSTMs in terms of mean squared error on the test sets, and is vastly more flexible in its application. In the experiments we demonstrated that the LARNNs with variable step length differentiate between the learned dynamics and the prediction step length, indicating that they learn more of the underlying distribution of the continuous system, than when learning with a fixed step length.

## Acknowledgments

This work was supported by the industry partners Borregaard, Elkem, Hydro, Yara and the Research Council of Norway through the project TAPI: Towards Autonomy in Process Industries, project number 294544.

## References

R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/69386f6bb1dfed68692a24c8686939b9-Paper.pdf.

E. Dupont, A. Doucet, and Y. W. Teh. Augmented neural odes. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/21be9a4bd4f81549a9d1d241981cec3c-Paper.pdf.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

A. Graves, G. Wayne, and I. Danihelka. Neural turing machines, 2014.

R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu. Exploring the limits of language modeling, 2016.

M. Maiworm, D. Limon, J. Maria Manzano, and R. Findeisen. Stability of gaussian process learning based output feedback model predictive control. *IFAC-PapersOnLine*, 51(20):455 – 461, 2018. ISSN 2405-8963. doi: https://doi.org/10.1016/j.ifacol.2018.11.047. URL http://www.sciencedirect.com/science/article/pii/S2405896318327058. 6th IFAC Conference on Nonlinear Model Predictive Control NMPC 2018.

J. M. Manzano, D. Limon, D. Muñz de la Peñ, and J. Calliess. Robust data-based model predictive control for nonlinear constrained systems. *IFAC-PapersOnLine*, 51(20):505 – 510, 2018. ISSN 2405-8963. doi: https://doi.org/10.1016/j.ifacol.2018.11.039. URL http://www.sciencedirect.com/science/article/pii/S2405896318326971. 6th IFAC Conference on Nonlinear Model Predictive Control NMPC 2018.

S. Moe, F. Remonato, E. I. Grøtli, and J. T. Gravdahl. Linear antisymmetric recurrent neural networks. In A. M. Bayen, A. Jadbabaie, G. Pappas, P. A. Parrilo, B. Recht, C. Tomlin, and M. Zeilinger, editors, *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, volume 120 of *Proceedings of Machine Learning Research*, pages 170–178, The Cloud, 10–11 Jun 2020. PMLR.

D. E. Seborg, D. A. Mellichamp, T. F. Edgar, and F. J. Doyle. *Process Dynamics and Control*. John Wiley & Sons, 2010. ISBN 9780470128671. URL https://books.google.no/books?id=_PQ42kOvtfwC.

K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057, 2015.