Systems & Control Transactions

Research Article - Peer Reviewed Conference Proceeding
ESCAPE 35 - European Symposium on Computer Aided Process Engineering
Ghent, Belgium. 6-9 July 2025
Jan F.M. Van Impe, Grégoire Léonard, Satyajeet S. Bhonsale,
Monika E. Polańska, Filip Logist (Eds.)

PSE PRESS

Recurrent Deep Learning Models for Multi-step Ahead Prediction: Comparison and Evaluation for Real Electrical Submersible Pump (ESP) System.

Vinicius V. Santana*[a], Carine M. Rebello[a], Erbet A. Costa[a], Odilon S. L. Abreu[b], Galdir Reges[b], Téofilo P. G. Mendes[b], Leizer Schnitman[b], Marcos P. Ribeiro[c], Márcio Fontana[b], Idelfonso Nogueira[a]*.

[a] Norwegian University of Science and Technology, Department of Chemical Engineering, Trondheim, Norway
[b] Federal University of Bahia, CTAI, Salvador, Bahia, Brazil
[c] CENPES, Petrobras R&D Center, Brazil
* Corresponding Author: idelfonso.b.d.r.nogueira@ntnu.no, vinicius.santana@ntnu.no

ABSTRACT

Predicting processes' future behavior based on past data is vital for automatic control and dynamic optimization in engineering. Recent advances in deep learning, particularly Artificial Neural Networks, have improved predictions in various engineering fields. Recurrent Neural Networks (RNNs) are well-suited for time series data, as they naturally evolve through dynamic systems with recurrent updates. Despite their high predictive power, RNNs may underperform if their training ignores the model's future application. In Model Predictive Control, for example, the model evolves over time using only current information, relying on its own predictions at later steps. A model trained for one-step-ahead predictions may fail when tasked with multi-step-ahead forecasting in autoregressive mode. This study explores deep recurrent neural network models for predicting critical operational time series of a large-scale Electric Submersible Pump system. We present an innovative training approach, framing the task as a multi-step-ahead prediction problem. Results show that aligning model training with its future use is crucial to ensure real-time performance.

Keywords: Deep Learning, System Identification, Electric Submersible Pumps, Artificial Neural Networks

INTRODUCTION

Models play an important role in chemical and industrial engineering practice. First-principles models describe the energy and mass transport in unit operations through continuous conservation laws, rate, and equilibrium expressions. They are usually capable of providing reliable predictions in a broad range of conditions and have parameters with physical meaning. On the other hand, dynamic high-fidelity models can be tricky to frame correctly and identify, especially for highly nonlinear systems with few observable states [1].

Black-box or data-driven models are often used to circumvent such a problem. In this sense, Artificial Neural Networks (ANNs) are essential for identifying nonlinear dynamic systems in chemical engineering. ANNs have been widely used since the 1980s for nonlinear dynamic system identification, mainly as the nonlinear function for systems identification in the Nonlinear AutoRegressive with Exogenous inputs (NARX) formulation [2], [3]. It is well-known that the NARX formulation is suitable for prediction tasks (short-term horizons) but may perform poorly recursively for long-term predictions (simulation mode).

In contrast, the Nonlinear Output Error (NOE) representation is more appropriate for engineering systems and long-term predictions (simulation) as noise is assumed to be an output additive, i.e., there is a deterministic dynamic of states, and noise is placed only at observables [2], [4]. When the NOE nonlinearity is framed with a neural network, its resemblance with recurrent neural networks (RNNs) is evident [5]. NOE is one of the most important representations for simulation in engineering systems, but it is often misused and comprehended in practice.

In real-time applications requiring multi-step-ahead

predictions and sensor feedback, such as predictive control, choosing the appropriate formulation for a data-driven model plays an essential role in the final predictive capacity and it can be a significant source of confusion. To address these questions, this study assesses the performance of a multi-step ahead predictor designed explicitly for model predictive control (MPC) applications. The predictor employs deep recurrent neural networks (RNNs) and is evaluated using real operating data from an Electrical Submersible Pump (ESP) system deployed in an offshore oil field.

Electric Submersible Pumps (ESPs) are a vital technology in the oil and gas industry, enabling the efficient lifting of fluids from subsea reservoirs to the surface. Their effective operation and control require a comprehensive understanding of the complex interactions between the reservoir, pump, and surface conditions. While models are instrumental for understanding these dynamics, first-principles modeling is particularly challenging due to the complexity of the governing equations. This makes data-driven approaches an appealing alternative for modeling ESP system dynamics.

Several studies have investigated data-driven methods for modeling ESP behavior. For example, Jordanou et al. (2022) developed an Echo State Network (ESN) as a surrogate nonlinear dynamic model using training data from a rigorous mechanistic model [6]. Similarly, Abreu et al. (2024) applied ESNs for zone-based Nonlinear Model Predictive Control (NMPC) of an ESP-lifted oil well to maximize oil production using data from a mechanistic model [7]. In another study, Costa et al. (2024) proposed a deep neural network for system identification and uncertainty quantification, leveraging mechanistic model data [8].

However, to our knowledge, no prior work has demonstrated the predictive capabilities of deep-learning-based, data-driven models using real operational data from ESP systems. The remainder of this text is organized as follows:

The Methodology details the model representation, data collection, model training and hyperparameter tuning strategies. In the Results, the test set performance is evaluated. The conclusions summarize the main findings and points to future research directions.

METHODOLOGY

Recurrent Neural Networks

A recurrent neural network (RNN) is an artificial neural network designed to process sequential information, such as time series data. It has a deterministic h state, and an evolution rule defines how the state changes based on its computed values and input data. This rule is typically a parameterized function $\mathcal{F}(\mathcal{W}_u, \mathcal{W}_x, b_u)$, such as a multilayer perceptron in a standard RNN or a more sophisticated structure in long-short-term memory (LSTM) architectures [9].

Training an RNN involves evolving the states according to the update rule and the current parameters. These states are mapped to outputs matching the dimensions of the observed data through a readout layer ($\mathcal{G}$). The prediction error is computed and used to adjust the parameters, often through gradient descent. This process is illustrated in Figure 1.

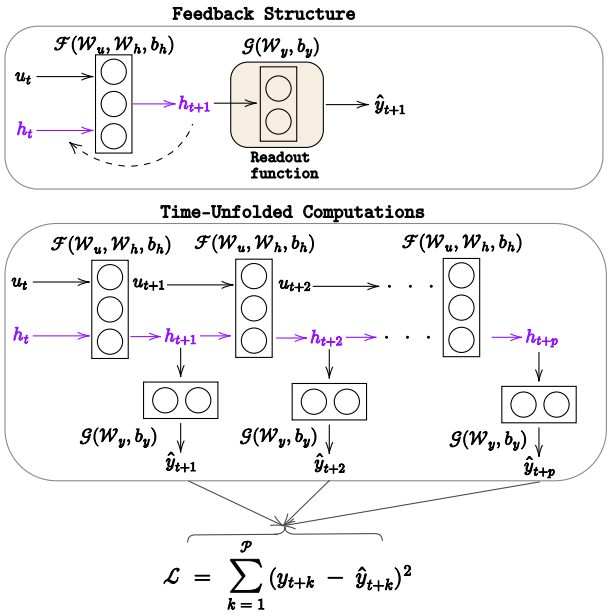

Figure 1. Diagrammatic representation of RNNs computations

However, practical implementations raise several questions: How many times should the states evolve? How many state observations should be used to calculate the error? There is no definitive answer to these questions. The training approach should align with the model's intended application [10]. In this work, we frame the model training as the closest to match a potential final application of model predictive control, as in [10].

Data collection and processing

Identifying nonlinear dynamic models from real process data presents significant challenges. Industrial processes typically operate at fixed set points, and shifting these points to gather dynamic data involves safety and financial risks. Consequently, many studies develop and validate mechanistic models to be later used as a data source instead - a strategy called simulation-assisted identification [10]. However, identifying mechanistic models for industrial-scale ESP systems is particularly challenging.

In this work, real operational data was used to train the models. Only a small subset of the total operational

data - usually during startup and shutdown - proves useful for dynamic model identification. The Figure 2 and the the Figure 3 highlight these regions for the two normalized target variables – TV1, TV2 and the two manipulated variables – MV1, MV2. The variables and time have been normalized to avoid disclosing sensitive information about the data source. The two variables have different levels of noise in the process and were selected to evaluate how robust to noise is the used methodology.

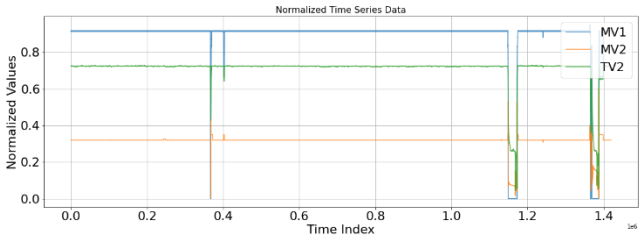

Figure 2. ESP real operational data history for MV1, MV2 and TV2.

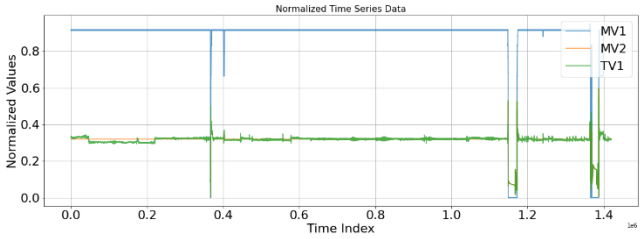

Figure 3. ESP real operational data history for MV1, MV2 and TV1.

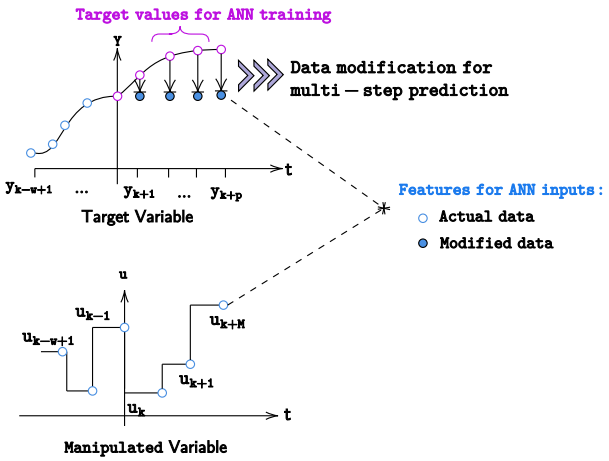

Figure 4. Illustration of the data preprocessing and organization for RNN training targeted at an MPC application. Adpated from [10].

The process data needs to be prepared differently to accommodate the particularities of each representation (NARX, NOE) and the final application for parameter identification (training) and inference. For this work, we are adopting a multi-step ahead prediction strategy that requires a unique data organization and follows the structure proposed in [10] and is depicted in **Figure 4**. The proposed data organization aligns well with the computations required in predictive control strategies, which should favor the model's predictive accuracy when handling this task. In contrast, **Figure 5** illustrates the traditional approach for preparing the data for single-step-ahead predictions. In this setting, the target output is a one-dimensional sequence of points. Multi-step-ahead predictions at inference time require a set of input distributions the model was not trained for.

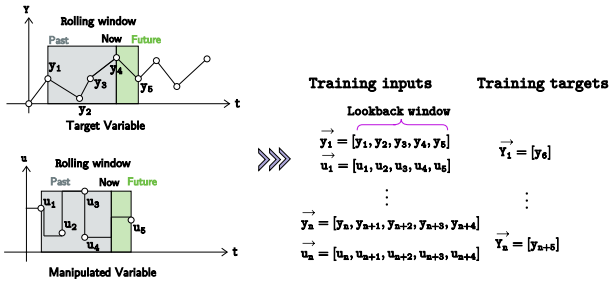

Figure 5. Illustration of the traditional data processing for single-step ahead prediction for training an RNN.

The dataset was split into training, validation, and test sets, ensuring no time overlap. Time series data was processed using a sliding window with a stride of 1, moving forward in time. The total window size was 12-time steps, including features and labels.

For the target variable to be used as features (predictors), 8 of the 12-time steps were actual past observations (white circles with blue contours). At the same time, the remaining 4 were repeated from the last valid observation (filled blue circles). For the manipulated variables (MV1 and MV2), all 12 points were actual observations (white circles with blue contours on the manipulated variable). The target consisted of 4 future actual observations (white circles with pink contours), as illustrated in the **Figure 4**.

This process resulted in a 3-dimensional feature tensor $X \in \mathbb{R}^{n_b \times 12 \times 3}$ where n_b is the number of sequences of size 12 and 3 corresponding to the two MVs and the target variable being predicted. The target is also a 3D tensor $Y \in \mathbb{R}^{n_b \times 4 \times 1}$.

Model Training and Hyperparameter Tunning

The estimation of the predictor's parameters constitutes the model training. One model was trained for each target variable (TVs), forming two multiple-input single-output (MISO) models. The mean squared error is the most common cost function for continuous numerical variables such as those used here.

For the proposed predictor, the cost function to be minimized is here proposed to be written as:

$$J(W,b) = \frac{1}{2m}\sum_{i=1}^{m}\sum_{k=1}^{4}\left(\hat{y}_{i,k},(W,b) - y_{i,k}\right)^2 \qquad (1)$$

i varies from 1 to the number of training sequences m. k varies from 1 to the number of time-steps 4. W and b are the trainable weights and biases. Typically, the number of sequences is chucked into smaller data sets, forming batch sequences for updating the parameters.

In RNN parameter identification, layer-wise gradients are typically computed using Backpropagation Through Time (BPTT) to solve the optimization problem. However, this method often leads to convergence problems in recurrent neural networks due to vanishing or exploding gradients. Over the years, various RNN variants have been proposed to address these issues. Among them, the Long Short-Term Memory (LSTM) variant has been widely adopted to extract long-range patterns in time series data [9]. Therefore, this architecture was used in this work.

Model selection entails defining its architecture before parameter estimation, including the number of layers, neurons, activation functions, and optimizer parameters such as learning rate, momentum (for momentum-based optimizers), decay, epochs, and mini-batch size. The hyperparameter space comprises a set of discrete and continuous variables, making their selection a complex task. One recent approach to select them efficiently is wrapping another learning algorithm (hyper-learner) around the original learning problem to select hyperparameters by monitoring a cost function from a separated dataset (validation data set). The hyper-learner may also have more parameters, but it is simpler to choose than the hyperparameters.

Machine learning practitioners often employ exploratory random and grid search for hyperparameter tuning [11]. However, they are very time-consuming as they allocate the same resources for all sampled hyperparameters. Most recently, it has been possible to find a method called HYPERBAND [12], which has the advantage of being up to 30 times faster in search speed than random grid search. HYPERBAND formulates hyperparameter optimization as a pure-exploration problem where a predefined resource is allocated to randomly sampled configurations.

The neural network's initialization, the training process, and the hyperparameter tuning were implemented using TensorFlow 2.10.1 [13] and Keras [14] tuner version 1.4.7, respectively.

Inference and state initialization

RNNs have a deterministic state variable that evolves according to the update rule (parameterized function $\mathcal{F}(\mathcal{W}_u, \mathcal{W}_x, b_u)$) in **Figure 6**. The initialization of this variable defaults to the null vector in many deep learning frameworks, including TensorFlow. During training with gradient descent, the default policy is that the states are restarted to zero for each batch. It has been shown that this is an effective way of learning the patterns in time series. However, this policy must be modified during real-time inference in a predictive control environment to accommodate the incoming data stream. In this sense, we propose the following state update policy for inference:

Algorithm 1 Neural Network Inference Policy

Input: Time-series data $\mathbf{X} \in \mathbb{R}^{T \times F}$, LSTM model M, warm-up steps W, step size S.
Output: Predictions $\hat{\mathbf{Y}}$, True Values $\mathbf{Y}_{\text{true}}$.
Step 1: Preprocessing
Normalize each feature f:

$$\mathbf{X}_f = \frac{\mathbf{X}_f - \min(\mathbf{X}_f)}{\max(\mathbf{X}_f) - \min(\mathbf{X}_f)}.$$

Reshape data to 3D:

$$\mathbf{X}_{3D} \in \mathbb{R}^{1 \times T \times F}.$$

Step 2: Warm-up
Extract the first W time steps:

$$\mathbf{X}_W = \mathbf{X}_{3D}[:, : W, :].$$

Reset the LSTM model's M state.
Pass $\mathbf{X}_W$ through M to initialize the state:

$$M(\mathbf{X}_W).$$

Save the updated state $\mathbf{S}_W$.
Step 3: Forecasting Loop
Initialize empty sets $\hat{\mathbf{Y}}, \mathbf{Y}_{\text{true}}$.
for $i = 0$ to $\lfloor \frac{T-W}{S} \rfloor - 1$ **do**
  Define start and end indices:

$$\text{start} = W + i \cdot S, \quad \text{end} = W + (i+1) \cdot S.$$

  Extract future data:

$$\mathbf{X}_S = \mathbf{X}_{3D}[:, \text{start} : \text{end}, :].$$

  Apply zero-order hold (ZOH) to the Target Variable with index 3:

$$\mathbf{X}_S[:, t, 2] = \mathbf{X}_{3D}[:, \text{start} + t - 1, 2], \quad \forall t \in \{1, \ldots, S\}.$$

  Predict outputs:

$$\hat{\mathbf{Y}} = M(\mathbf{X}_S).$$

  Save predictions $\hat{\mathbf{Y}}$ and true values:

$$\mathbf{Y}_{\text{true}} = \mathbf{X}_{3D}[:, \text{start} : \text{end}, 3].$$

  Restore the state $\mathbf{S}_W$:

$$\text{set\_lstm\_states}(M, \mathbf{S}_W).$$

  Perform one additional step to update the state:

$$\mathbf{X}_1 = \mathbf{X}_{3D}^{2}[:, \text{end} : \text{end} + 1, :].$$

  if $\mathbf{X}_1 \neq \emptyset$ **then**
    Use $M(\mathbf{X}_1)$ to update the state $\mathbf{S}_W$.
  end if
end for
Output: Predictions $\hat{\mathbf{Y}}$, True Values $\mathbf{Y}_{\text{true}}$.

Figure 6. Algorithm for inference and state initialization.

The algorithm operates in two alternating phases to emulate real-time application behavior. The first phase updates the internal states of the LSTM model using the most recent incoming data, ensuring that the model updates its states based on the latest observation. The second phase performs a 4-time-step-ahead prediction based on the updated state. These two phases are executed alternately across the test set, simulating a real-time scenario where the model continuously processes

new data while concurrently making multi-step future predictions based on the last observation and MV's actual inputs.

Results and Discussion

The hyperparameter search space was limited to the numbers of neurons in the LSTM layer between 60 and 160 and step of 20, the activation function between hyperbolic tangent and rectified linear unit and minibatch size between 16 and 256 with step of 32. The ADAptive Moment estimation (ADAM) optimizer was used with a fixed learning rate of 0.001. The validation data set error was used to determine the best architecture during the search. The hyperband was parametrized with a factor of maximum allocated epochs of 100 and 1 iteration. The best configurations for TV1 and TV2 are displayed in the tables.

Table 1. Hyperparameters for TV1 and TV2 RNNs.

Neurons	Activation	Batch size	Variable
60	tanh	16	TV1
160	tanh	16	TV2

After finding the best hyperparameters, the model was retrained with the selected architecture for 1000 epochs, with early stopping and 100-step patience, by monitoring the validation set to avoid overfitting.

By applying the algorithm in **Figure 6**, a test set mean absolute error of 1.22×10^{-3} for TV1 and 2.3×10^{-3} for TV2.

The **Figure 7** and **Figure 8** show the parity plot and history for TV1. In the parity plot, predictions and actual values seem to be randomly distributed along the diagonal line in Figure. Still, there is a considerable variance due to the noise in the data. In **Figure 8**, it is possible to observe that the predictions follow the observations satisfactorily.

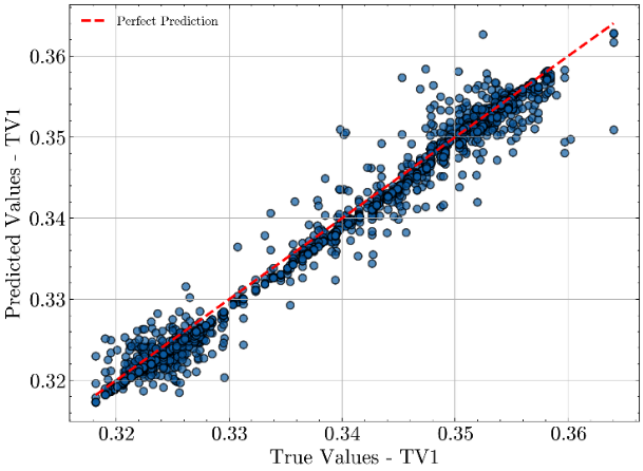

Figure 7. Parity plot for TV1.

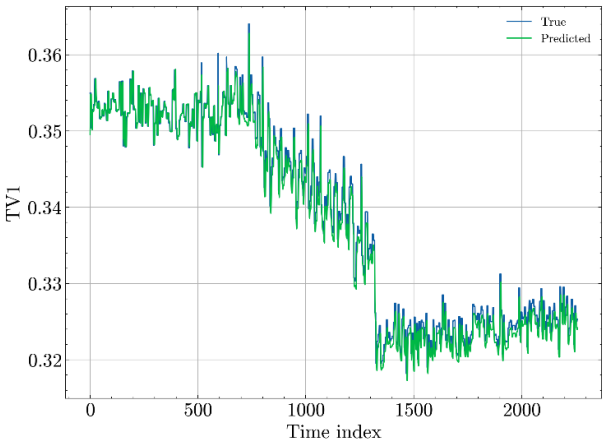

Figure 8. History of actual values and predictions for TV1.

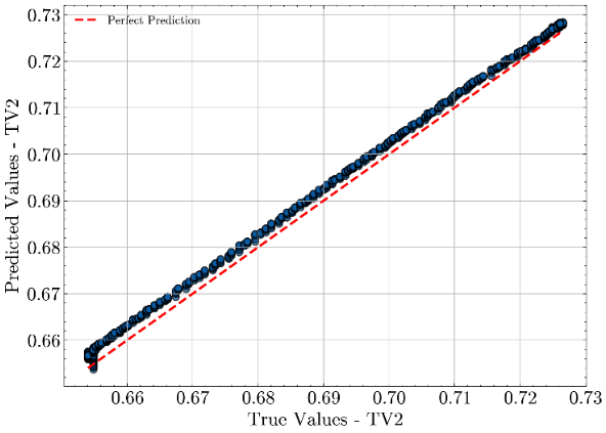

Figure 9. Parity plot for TV2.

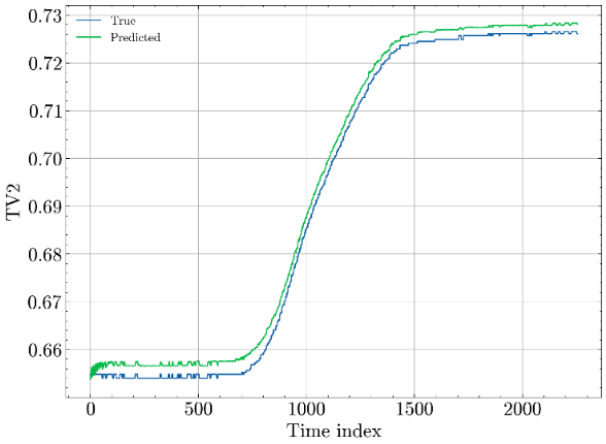

Figure 10. History of actual values and predictions for TV2.

The **Figure 9** and **Figure 10** show the parity plot and history for TV2 for the test set data set. They show that the predictions' dynamic behavior is close to that of the actual values. However, a slight bias prevents a better prediction. The prediction bias is more evident in **Figure 10** where the target variable 2 is displayed over time. However, for a real-time model-based application such as MPC, this mismatch would fit into the minimum requirements, and the additional modeling efforts and data

requirements would be beneficial, but the improvement in control performance may become marginal [15].

CONCLUSIONS

This works investigates the formulation of dynamic system identification for an industrial size electrical submersible pump system using a multi-step ahead predictor targeted at model predictive control. The data processing considers how information is processed in real-time and formulate the model architecture and parameter identification accordingly. It has been shown the strategy provides a satisfactory predictive capacity and further attempts of improving may become marginal in a possible control strategy implementation. Future works might explore the use of new deep learning architectures for processing sequenced information such as Transformers and the introduction of physics constraints of the observed variables for enhanced extrapolation power.

ACKNOWLEDGEMENTS

This work was carried out as a part of SUBPRO-Zero, a Research Centre at the Norwegian University of Science and Technology (project code 100589 and CristinID 2720670). The authors gratefully acknowledge the project support from SUBPRO-Zero, which is financed by major industry partners and NTNU. This publication has been produced with support from the HYDROGENi Research Centre (hydrogeni.no), performed under the Norwegian research program FMETEKN. The authors acknowledge the industry partners in HYDROGENi for their contributions and the Research Council of Norway (333118).

REFERENCES

1. O. Levenspiel, “Modeling in chemical engineering,” 2002. [Online]. Available: www.elsevier.com/locate/ces
2. Q. Zhang, “Nonlinear system identification with output error model through stabilized simulation,” IFAC Proc. Vol., vol. 37, no. 13, pp. 501–506, 2004, doi: 10.1016/S1474-6670(17)31273-9.
3. A. U. Levin and K. S. Narendra, “Recursive identification using feedforward neural networks,” Int. J. Control, 1995, doi: 10.1080/00207179508921916.
4. H. Koivisto, “A Practical Approach to Model Based Neural Network Control Hannu Koivisto,” Tampere University of Technology, Tampere, 1995. [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.113.6407
5. D. M. Himmelblau, “Applications of artificial neural networks in chemical engineering,” Korean J. Chem. Eng., vol. 17, no. 4, pp. 373–392, Jul. 2000, doi: 10.1007/BF02706848.
6. J. P. Jordanou, I. Osnes, S. B. Hernes, E. Camponogara, E. A. Antonelo, and L. Imsland, “Nonlinear Model Predictive Control of Electrical Submersible Pumps based on Echo State Networks,” Adv. Eng. Informatics, vol. 52, no. December 2021, p. 101553, 2022, doi: 10.1016/j.aei.2022.101553.
7. O. S. L. de Abreu, M. P. Ribeiro, B. P. Foresti, L. Schnitman, and M. A. F. Martins, “An implementable zone-based NMPC with Echo State Networks applied to an ESP-lifted oil well for maximum oil production,” 34th Eur. Symp. Comput. Aided Process Eng. / 15th Int. Symp. Process Syst. Eng., vol. 53, pp. 1717–1722, 2024, doi: https://doi.org/10.1016/B978-0-443-28824-1.50287-8.
8. E. Almeida Costa et al., “An uncertainty approach for Electric Submersible Pump modeling through Deep Neural Network,” Heliyon, vol. 10, no. 2, Jan. 2024, doi: 10.1016/j.heliyon.2024.e24047.
9. S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” Neural Comput., 1997, doi: 10.1162/neco.1997.9.8.1735.
10. J. Park, “Hybrid Machine Learning and Physics-Based Modeling Approaches for Process Control and Optimization,” Brigham Young University, USA, 2022.
11. J. Bergstra and Y. Bengio, “Random search for hyper-parameter optimization,” J. Mach. Learn. Res., 2012.
12. L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, “Hyperband: A novel bandit-based approach to hyperparameter optimization,” J. Mach. Learn. Res., vol. 18, pp. 1–52, 2018.
13. M. Abadi et al., “TensorFlow: A system for large-scale machine learning,” in Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016, 2016.
14. F. Chollet, “Keras: The Python Deep Learning library,” Keras.Io, 2015.
15. S. Zhan and A. Chong, “Data requirements and performance evaluation of model predictive control in buildings: A modeling perspective,” Renew. Sustain. Energy Rev., vol. 142, no. February, p. 110835, 2021, doi: 10.1016/j.rser.2021.110835.