MLME project - group 14

Tim Knittel
Matr.: 225653

Lion Döpp
Matr.: 220082

Birger Vedder-Stute
Matr.: 229018

Akanksh Sukumara
Matr.: 269930

***Abstract*— This work summarizes the application of a NARX (Nonlinear Autoregressive with Exogenous Input) neural network to predict slug flow crystallization. The model accurately predicted process variables and particle sizes. Uncertainty quantification via CQR (conformalized quantile regression) validated the reliability of these predictions.**

I. Introduction

This report summarizes the results of the Machine Learning Project from the summer semester 2025 for group 14.
The topic was to predict data from a slug flow crystallization using a model. A NARX (Nonlinear Autoregressive with Exogenous Input) model was used for this purpose and the uncertainty quantifications were also determined.
In the following, the various tasks required for the finished model are divided into individual sections. The first step is to import and analyze the raw data. The next very important part is preparing the data and dividing it into different products and processes using clustering. This is followed by the NARX model, which was used to determine the predictions for the individual processes. The analysis and classification of the results using quantitative regression followed next. Finally, the work produced is summarized and the results of the model are shown.

II. Data import and Inital Exploration

The project began with importing and exploring the raw process data. This initial step was essential to gain an understanding of the dataset's structure and to identify possible issues such as noise, gaps, or irregularities that could impact model performance.
The given data consists of 98 individual textfiles which contain input-data as seen in the input vector $u = [Q_{PM}, Q_{TM}, Q_{air}, w_{cryst}, c_{in}, T_{PM,in}, T_{TM,in}]$, which includes the flow rates, the solid fraction in the feed and the inlet conditions for concentration and temperatures. The output-data concludes from the liquid and outer tube temperature, the concentration and the particle size distribution shown by the 0.1, 0.5 and 0.9 quantiles. All this is in the output vector $y = [T_{PM}, T_{TM}, c_{PM}, d_{10}, d_{50}, d_{90}]$ [1].
When analyzing the data, we realized that some of the diameter values must be incorrect, as some of them are more than a factor of 200 larger than the previous and later ones.

III. Data Preprocessing

In the context of this project, a multi-stage pipeline for data pre-processing and cluster analysis was implemented in the form of a Python class for the purpose of modularization. The purpose of this implementation was to evaluate the experimental process data efficiently, comprehensively and robustly. The workflow is designed to ensure the quality of the data on the one hand and to generate reproducible and objective groupings (clusters) on the other. These clusters then serve as the basis for further analysis and modelling. For the modeling task, a total of 98 rawdata files, each containing 1000 data points, were available. Thus providing a total of approximately 100.000 data points for analysis. Each of these files is representative of a dynamic process trajectory.

A. Initial Coarse Filtering

Initially, the raw data underwent a preliminary filtration process. It was determined that all files with an unsuitable format and those whose mean value of the particle size distribution parameter d50 was greater than a threshold value of 0.001 were being excluded from further analysis. This value is based on reasonable process-related limits for the targeted particle size, with the purpose of preventing extreme values or incorrect measurements from distorting the cluster formation. Seven of the original 98 data files were discarded due to the presence of errors.

B. Scaling

To enable subsequent classification, all remaining data were merged into a single DataFrame and jointly scaled. This approach ensures that the feature space is standardized across all samples, thus facilitating consistent clustering and further analysis. The StandardScaler from *sklearn.preprocessing* was applied for the purposes of normalization and scaling. This scaler transforms the data by centering the mean μ of each feature to zero and scaling the standard deviation σ to one (Eq: 1). Consequently, the values are standardized with a mean of zero and a standard deviation of one after transformation.

$$x' = \frac{x - \mu}{\sigma} \tag{1}$$

This transformation ensures that all features are equally weighted in the cluster analysis and subsequent steps. The scaler employed in this study is saved for subsequent classification tasks to ensure consistency. The joblib module was utilized to store the data.

C. Clustering

A variety of algorithms are available for unsupervised learning, also known as clustering. Examples include K-means and DBscan. In this project, the HDBSCAN algorithm (Hierarchical Density-Based Spatial Clustering of Applications with

Noise) was implemented to ensure flexibility and precise data traceability. The algorithm has been demonstrated to exhibit several advantages over traditional clustering algorithms. In contrast to the K-Means approach, it does not require a specification of the number of clusters in advance. It is capable of identifying clusters of varying densities and arbitrary shapes. In addition, HDBSCAN demonstrates resilience to noise and outliers due to its capacity to explicitly label points that do not belong to any cluster. These characteristics render HDBSCAN particularly well suited for real-world datasets characterized by intricate structures and heterogeneous density distributions.

The HDBSCAN procedure was initiated with a minimum of 500 data points per cluster and the prediction true function. The minimum number of points was determined in order to prevent the formation of small clusters while ensuring that the total number of points remains below the maximum capacity of a single trajectory. In addition, the HDBSCAN model was trained with the prediction data option enabled to facilitate the subsequent allocation of new data points to existing clusters. This function has two primary applications. Firstly, it enables the allocation of points to the clusters. Secondly, it provides a corresponding probability of occurrence. [2] In this instance as well, the joblib module was utilized for the purpose of storing the cluster specifications. In order to facilitate the visualization of the clusters, a dimensional reduction to N = 2 was conducted using PCA (Principal Component Analysis). [3] The graphical representation (Figure 1) is employed in two distinct capacities: firstly, to ensure the quality control of the cluster formation process, and secondly, to facilitate the comprehension of the distribution and separability of the data groups.

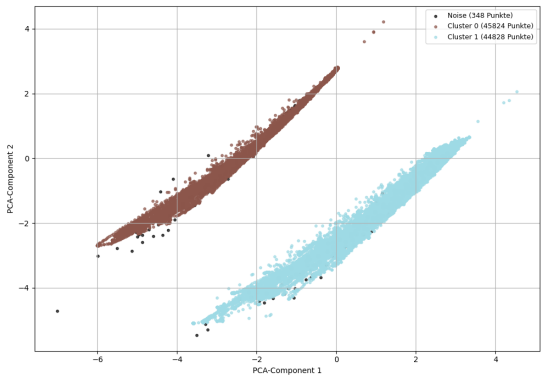

Fig. 1. The resulting clusters after the analysis and the filtering completed.

The presence of two distinct clusters (roughly 45000 data points per cluster) and some noise (roughly 350 data points) was identified (Figure 1). These are referred to as clusters 0 and 1 in the following and the noise as -1. However, the implementation of clustering results in the disruption of the orderly sequence, leading to the amalgamation of the time series and the consequent degradation of the system's dynamics. Instead, each original trajectory is read in individually, scaled, and then classified according to the cluster specifications that were previously collected. This guarantees the maintenance of the time series and the distinction between trajectories. Predictions were only processed further if they included only the two identiefied clusters (0 and 1) or noise (-1), in order to ensure clear separation and comparability. The results were then summarised in clustergroups.

D. Outlier Filtering

In order to achieve better training data, the data set was subjected to outlier removal. Several outlier detection methods were evaluated in this work, including the HDBSCAN-based filter using outlier scores, the Z-score filter, and the Mahalanobis distance. Among these, the Mahalanobis distance proved to be the most effective for identifying outliers in the dataset [4]. In order to enhance the quality of the results, a two-stage outlier adjustment was performed within each cluster. Initially, the Mahalanobis distance was employed to retain the innermost 97 % of the data points with regard to the output variables. The selection of the 97th percentile constitutes a rational compromise, as it ensures the retention of sufficient clean data for modeling while concurrently eliminating extreme values due to noisy data that could compromise the integrity of the training cluster. This sequential approach resulted in the most robust data cleaning and led to improved model performance. In a similar manner, the same approach was used to filter outliers for the inputs, but with only the 99th percentile, because the data was more smooth, resulting in a smaller number of outliers. This approach was selected to eliminate outliers while preserving the dynamics of the system data. The results of the clustering by outlier distance were then visualised and saved as separate files for each cluster, with the objective of enabling modular further processing and targeted analysis (Figure 2).

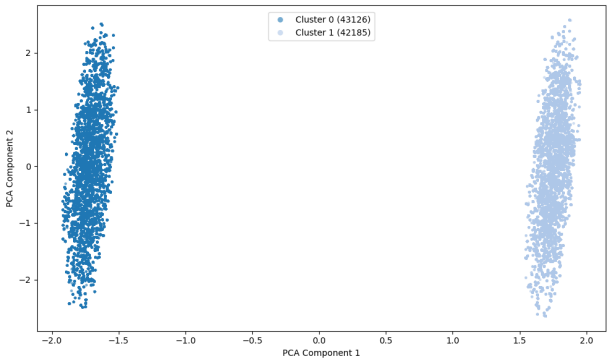

Fig. 2. The resulting clusters after the analysis and the filtering completed.

E. Cluster Classification

The final stage of the process involved a comparison of the classified individual files with the adjusted clusters. The intersection of the data points was determined and saved for each cluster.

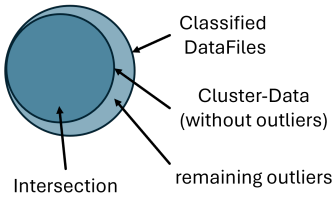

Fig. 3. The process of intersecting the raw data with the cleansed cluster data is undertaken with the objective of removing outliers.

This step ensures that only consistent and outlier-adjusted data points are used in subsequent analyses. The selection of parameters and threshold values was determined with the objective of enhancing data quality and meeting the requirements of robust clustering while maintaining the time-dependent dynamics of the trajectories.

IV. Data Lagging and division into data sets

The cluster-based pre-processing pipeline was followed by the implementation of an additional transformation step, which was specifically designed to prepare the data for the requirements of a NARX model. The processed and outlier-adjusted cluster data sets constituted the foundation for this step, ensuring that only consistent, robust and cluster-specific time series were utilised in the subsequent modelling phase. For each of the identified clusters, all associated time series files were systematically processed. The focal point of this phase was the generation of lagged feature sets. In accordance with the stipulated requirements of the NARX approach, the current values of the exogenous input variables were considered for each time point within each trajectory, in addition to their historical values over a defined lag window. Furthermore, past values of the output variables (the autoregressive component) were incorporated. This comprehensive construction of the feature space enables the NARX model to explicitly incorporate process dynamics, temporal dependencies and feedback effects within each cluster [1]. The lagged input vectors were created by concatenating the most recent lag values of all relevant input variables and the preceding lag-1 values of the outputs. A lag of d=5 time steps was used for the construction of the input features.

$$\begin{aligned} \mathbf{x}_t &= [u_{i,t-l},\ y_{j,t-l}]^T \\ \text{with } i &= 1, \ldots, m = 7; \\ j &= 1, \ldots, p = 6; \\ l &= 1, \ldots, d = 5 \end{aligned} \tag{2}$$

The resulting lagged samples were then paired with the target outputs at each current time step. This approach is designed to ensure that both the immediate and the recent historical context of each data point are captured, as is necessary for dynamic process modelling. In order to prevent information leakage and to preserve the temporal structure of the data, the splitting into training, validation and test sets was carried out on a per-trajectory basis. Complete time series were randomly assigned exclusively to one of these data partitions, with typical ratios of 80 % for training, and 10 % each for validation and testing. This strategy guarantees the maintenance of model evaluation biaslessness and reliable assessment of the generalisation capability with respect to unseen process trajectories.[Zhao] For each cluster and data partition, all lagged samples were consolidated into unified data sets and stored as dedicated files. This facilitates modular access for subsequent modelling steps, including cluster-specific NARX training and evaluation.

V. NARX-Model

To implement the NARX architecture, a feedforward neural network (MLP) was used as described in Bishop [6]. The modeling approach was split according to the clusters obtained in Section III-C. In addition, for each cluster, separate models were trained for the main process variables and the particle size distribution outputs (d_{10}, d_{50}, d_{90}). This specialization helped improve prediction precision for each target set. All models were trained on the full input feature space relevant to each cluster. Training was performed for up to 150 epochs, with early stopping based on validation loss and a patience of 15 epochs.

All model hyperparameters were set individually per cluster and output based on the results obtained during hyperparameter tuning.

A. Hyperparameter Tuning

To select optimal hyperparameters for the NARX model, a tuning was performed for each cluster using the Optuna framework using a Bayesian type algorithm. The search space included lag length (2–5), number of layers (1–4), hidden sizes per layer (8–100), activation function (ReLU or GELU), batch size (64 or 128), learning rate (10^{-5} to 10^{-3}), L_1 (10^{-8}–10^{-5}) and L_2 (10^{-12}–10^{-5}) regularization, dropout (0–0.2), and loss function (MSE or Huber).The optimizer was set to ADAMW. Each tuning run used 100 trials per cluster and output. The best model was selected based on mean absolute error (MAE) on the validation set. These optimal parameters were subsequently used for the final NARX models and are outlined in the table I.

TABLE I
Best NARX model hyperparameters per cluster and output. Cl.: cluster; Out.: output group (main variables or dxx for particle size); Lag: lag window; Hidden Sizes: neurons per hidden layer; Act.: activation function; LR: learning rate.

Cl.	Out.	Lag	Hidden Sizes	Act.	LR
0	main	4	55, 81, 19	gelu	3.4e-4
0	dxx	5	32	relu	1.4e-4
1	main	5	72, 64, 39	gelu	2.0e-4
1	dxx	4	97	relu	2.5e-4

B. Model Performance and Results

Table II compares the developed models' test MAE and MSE (averaged across clusters obtained during closed-loop prediction) to benchmark values for each output [7]. The model matches the benchmarks and outperforms especially

in regard to the particle size distribution. This highlights the prediction capabilities for the closed-loop prediction.

TABLE II
TEST MAE AND MSE FOR EACH OUTPUT, AVERAGED OVER BOTH CLUSTERS IN CLOSED-LOOP (ONE-STEP-AHEAD) PREDICTION, USING GROUND-TRUTH VALUES AS INPUTS. "CLOSED LOOP" BENCHMARK VALUES FROM SCHOUKENS & LJUNG [7] ARE SHOWN FOR COMPARISON. ALL VALUES ARE SHOWN IN SCIENTIFIC NOTATION.

Output	This Work		Benchmark	
	MAE	MSE	MAE	MSE
T_{PM}	1.10e-1	8.10e-2	7.30e-2	1.30e-2
T_{TM}	1.04e-1	7.70e-2	8.76e-2	1.39e-2
c	1.69e-4	2.40e-7	9.70e-5	4.81e-8
d_{10}	1.70e-5	5.10e-10	7.93e-5	1.82e-6
d_{50}	1.56e-5	4.60e-10	6.47e-5	1.43e-6
d_{90}	2.82e-5	1.40e-9	8.00e-5	1.57e-6

Additionally, to illustrate model performance, Figure 4 shows parity plots comparing the model's predictions to the true values on the test set for the main outputs of Cluster 0. The close alignment of points along the diagonal lines indicates strong predictive accuracy across all targets. For the particle sizes the distribution of points is less aligned due to the measuring noise.

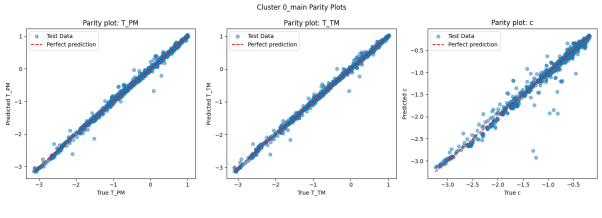

Fig. 4. Parity plots of predicted versus true values for main outputs of Cluster 0 on the test set. The plots demonstrate that the model achieves strong predictive accuracy, with most points clustering near the ideal diagonal.

VI. UNCERTAINTY QUANTIFICATION AND QUANTILE REGRESSION

To estimate predictive uncertainty, we implemented Conformalized Quantile Regression (CQR) based on trained NARX models. First, separate neural networks were trained to predict the lower and upper conditional quantiles (10th and 90th percentiles) of the model output. These quantile regressors were trained using the pinball loss function, which is tailored for quantile estimation [5].

The CQR method was then applied using a hold-out calibration dataset. For each output, residuals between the observed and predicted quantiles on the calibration set were computed. These residuals served to calculate nonconformity scores, which were used to correct the quantile predictions in order to obtain valid prediction intervals with the desired confidence level.

The final prediction intervals were evaluated on the test set in terms of empirical coverage, interval width, and the number of outliers (i.e., test samples falling outside the predicted intervals). The implementation was carried out separately for each output variable, allowing for individual uncertainty quantification.

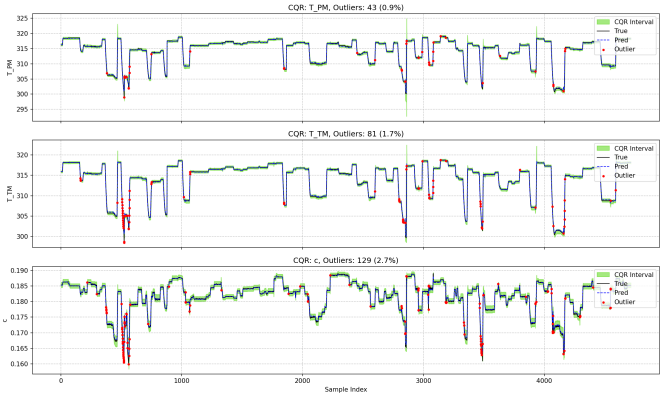

Fig. 5. CQR prediction intervals (90%) for all main outputs in Cluster 0, showing test set predictions versus true values. The green bands indicate the conformalized quantile regression (CQR) intervals, black lines are ground truth, blue dashed lines are predictions, and red dots mark outliers. Results show that the model provides well-calibrated uncertainty estimates, covering the vast majority of test samples with few outliers.

VII. CONCLUSION

This investigation successfully developed a data-driven modeling workflow for slug flow crystallization utilizing NARX neural networks. Unsupervised data preprocessing, including HDBSCAN clustering and Mahalanobis distance-based outlier removal, to ensure high-quality, temporally consistent input for model training.

Cluster-specific NARX models demonstrated predictive accuracy, particularly for particle size distributions, with error metrics outperforming established benchmark values. Uncertainty quantification through Conformalized Quantile Regression provided well-calibrated prediction intervals.

VIII. FUTURE SCOPE

While the current results are promising, several extensions could be pursued:

- **Deployment in Real-time Systems:** Integrating the trained models into live process monitoring systems could support operational decision-making.
- **Adaptability to Other Setups:** Testing the workflow on different crystallization systems or conditions could validate its broader applicability.
- **Improved Interpretability:** Adding explainability tools (e.g., SHAP values) would help identify key process drivers and support engineering understanding.
- **Advanced Uncertainty Techniques:** Exploring Bayesian or ensemble-based models could provide deeper insights into uncertainty under limited or noisy data.

IX. Authors

TABLE III
SECTION AUTHORSHIP OVERVIEW.

Section	Author(s)
Introduction	Birger Vedder-Stute
Data Import and Initial Exploration	Birger Vedder-Stute
Data Preprocessing	Tim Knittel
Clustering	Tim Knittel
Data Lagging and Division	Tim Knittel
NARX-Model	Lion Döpp
Hyperparameter Tuning	Lion Döpp
Model Performance and Results	Lion Döpp
Uncertainty Quantification and Quantile Regression	Akanksh Sukumara
Conclusion and Future Scope	Akanksh Sukumara

References

[1] F. Brabender, "Data-based modeling of slug flow crystallization with uncertainty quantification – Task description," *Machine Learning Methods for Engineers – process automation systems*, 2025.

[2] scikit-learn developers, "sklearn.cluster.HDBSCAN," *scikit-learn: Machine Learning in Python*, [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.HDBSCAN.html#, accessed: 15-Jul-2025.

[3] scikit-learn developers, "sklearn.decomposition.PCA," *scikit-learn: Machine Learning in Python*, [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html#, accessed: 15-Jul-2025.

[4] SciPy developers, "scipy.spatial.distance.mahalanobis," *SciPy v1.11.0 Manual*, [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.mahalanobis.html, accessed: 15-Jul-2025.

[5] I. Bauer, H. Haupt, S. Linner, "Pinball boosting of regression quantiles," *Computational Statistics & Data Analysis*, vol. 200, p. 108027, 2024. DOI: 10.1016/j .csda.2024.108027.

[6] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[7] J. Schoukens and L. Ljung, "Nonlinear System Identification: A User-Oriented Roadmap," *arXiv preprint arXiv:1902.00683* , 2019. Available: https://arxiv.org/abs/1902.00683