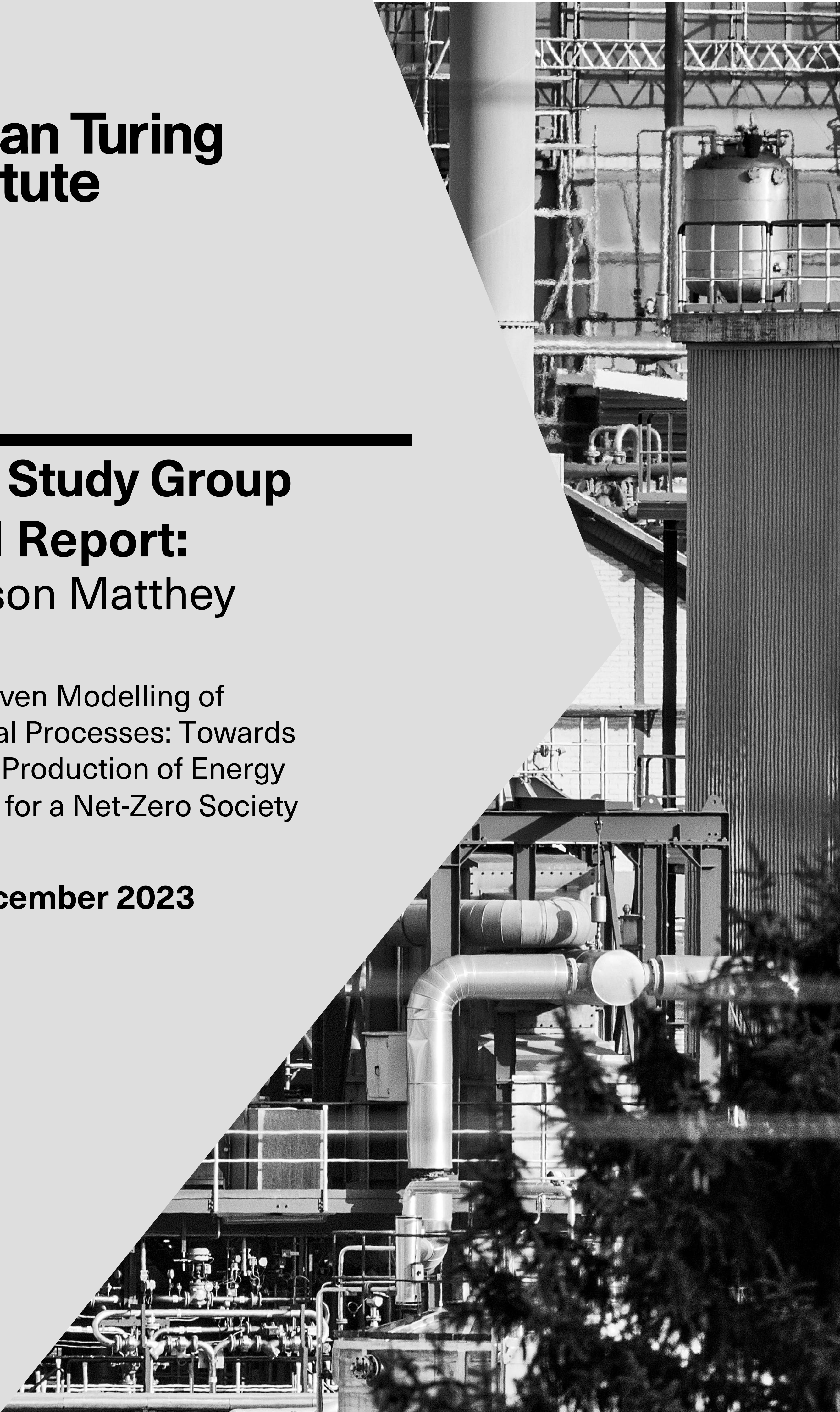
The Alan Turing Institute

Data Study Group Final Report: Johnson Matthey

Data Driven Modelling of Chemical Processes: Towards Flexible Production of Energy Carriers for a Net-Zero Society

4-8 December 2023



https://doi.org/10.5281/zenodo.13847810

Contents

1 Executive summary

1.1 Challenge overview

Johnson Matthey (JM), an FTSE 250 company, is a global leader in speciality chemicals and sustainable technologies with a 200+ year history of using advanced metals chemistry to tackle the world's biggest challenges. JM continually seeks to expand on products and services which provide sustainability benefits through the positive impact they have on the environment, resource efficiency and human health. For instance, JM are interested in various new manufacturing processes that help build a sustainable society, such as power-to-X, power-to-fuels, and green hydrogen processes. This challenge mainly pertains to the flexible operation of these processes: we seek to understand how they can be operated with a time-changing feedstock such as electricity.

Flexibility in the energy system is increasingly important in the context of the modern energy landscape, given the integration of renewable energy sources, the decentralisation of energy production, and the evolving demand patterns of consumers. Moreover, increasing flexibility from all sources contributes to the overall stability and reliability of the energy grid by providing tools and mechanisms (e.g. demand response, energy storage, and flexible generation) to match supply and demand in real-time, reducing the risk of blackouts and other grid disturbances. One promising avenue towards increased flexibility on the demand side is by scheduling the production of green fuels, such as electrolysis-based hydrogen, which can be converted to chemical carriers like methanol or ammonia. This paradigm can also be expanded to include the storage of excess fuel and/or its combustion as a source of flexible generation.

Nevertheless, this strategy of flexible production requires a considerable deviation from conventional operational practices, where processes are generally designed for and operated around some steady state(s). In fact, process control systems are generally designed and tuned to reject disturbances, or to maintain a steady state, rather than to enable quick changes in process operation [15]. The main challenge is balancing between agile response, e.g. changes in feedstock availability, while remaining within safe operating limits and meeting product specifications.

1.2 Main objectives

Given the above context, this project studies how data-driven modelling can play a crucial role in understanding process dynamics. Specifically, chemical manufacturing processes (and their associated control systems) continuously record operating data during routine operations. Can we leverage this operating data recorded by a process to understand its capacity for flexible operation?

Therefore, the central aim of this challenge is to construct a dynamic model that can predict the behaviour of a chemical process in response to changes in inputs and disturbances. We focus on the case of methanol production, as it exemplifies the power-to-fuels setting (using green hydrogen as the primary feed to produce methanol).

Towards this goal, we outline three main objectives:

1. Determine the importance of features/variables to identify which recorded measurements are necessary to represent a flexible manufacturing process.
2. Assess the degree to which the process dynamics can be learned from recorded dynamic operating data, given these features.
3. Use uncertainty quantification techniques to understand the degree to which the predictions of a data-driven model can be trusted, and when the model should be updated.

1.3 Approach

Our approach comprises the following: first we perform data analyses including pre-processing and dimensionality reduction by principal component analysis (PCA). These analyses help determine what model inputs/features should be used. Next, we investigate the performance of several data-driven dynamic models which include dynamic mode decomposition (DMD), long short-term memory (LSTM) networks, and nonlinear autoregressive exogenous (NARX) models. We furthermore study the use of conformal prediction to quantify the uncertainty of the model predictions.

1.4 Main conclusions

The study's outcomes effectively address the central objectives, with key findings summarised under each question:

Feature Selection Approaches: The analysis encompassed 1131 time-indexed variables, underscoring the importance of feature selection and dimensionality reduction.

- The principal component analysis (PCA) results in Figure 4 show that the first 40 principal components can capture the majority ($\sim$90%) of the variation within the dataset.

  Due to the unsupervised nature of PCA, this analysis cannot determine how much variability in the 1117 state variables can be explained by the 14 manipulated variables. Thus, this result only provides a general understanding rather than reducing the number of prediction inputs.

- Based on the findings from our exploratory data analysis, we observed significant correlations between various elements of our time-series data, as detailed in correlation Table 1. These correlations, particularly with the output variables, underscore the intricate inter-dependencies within the dataset. The presence of a high correlation among these variables suggests the potential for leveraging advanced analytical methods that capitalise on these relationships. While specific methodologies such as convolutional long short-term memory (ConvLSTM) offer promising avenues for harnessing these correlations, it's crucial to emphasise that the primary takeaway from our analysis is the substantial correlation within the dataset. Such insights pave the way for employing sophisticated techniques that can effectively utilise the inherent correlations in multivariate time-series analysis.

Model Implementation: Several dynamic models were implemented as summarised below.

- The application of dynamic mode decomposition (DMD) to the entire time-series dataset necessitates a re-evaluation of how to address the dimensionality challenges inherent in handling data matrices. Conversely, when implemented on a per-sample basis with separate treatment of state and control variables, DMD encounters difficulties in accurately reconstructing the original dynamics.

- The multi-input-multi-output Long short-term memory (LSTM) model gives relatively accurate predictions when trained on noise-free data. Furthermore, the prediction performance is reasonable even when trained on noisy data. The LSTM model can give accurate multistep-ahead predictions, tested up to 5-step ahead here. Nevertheless, the prediction error can accumulate with the increasing number of prediction steps.
- Nonlinear autoregressive exogenous (NARX) models are trained on noise-free and noisy datasets to predict the test datasets in order to evaluate their generalisation and predictive performance on the unseen datasets. The models perform well, with normalised RMSE values for noise-free and noisy test datasets of 0.01 and 0.024, respectively.
- Comparing the root mean squared error (RMSE) computed for test datasets for NARX and LSTM, it is observed that NARX exhibited better predictive performance compared to those of LSTM.

1.5 Limitations

A primary limitation of this study is the use of data from a single, simulated chemical process, with a simulated (and perhaps idealised) noise model. We did not investigate common process practicalities such as missing measurements, sensor faults, or process drift. Furthermore, we were limited by time and computational resources to fully investigate the use of larger model structures, most notably the use of transformer models.

2 Data overview

2.1 Dataset description

The data for this challenge comprised simulated data from a simplified dynamic model of a methanol synthesis plant. This setting was selected because methanol is, (i) a key platform molecule that can be synthesised directly from carbon dioxide and hydrogen, and (ii) a common intermediate in e-fuels production. Moreover, although the process has relatively simple chemistry, it has interesting process dynamics, with stable and unstable operating regions which produce very different behaviour.

A Pandas HDF5 datastore was provided by the team at Johnson Matthey, containing in total 344 DataFrames. Each DataFrame is a time-indexed, tabulated dataset, naturally interpreted as a multivariate time series. Each DataFrame represents a single experiment involving the perturbation of 14 manipulated variables. These variables are ramped at a variable rate from the 0.2-hour time step, and the experiment is designed to continue until the two-hour mark, with measurements recorded 100 times per hour. This results in a total of 201 data points per experiment (including the zero time point, $t = 0[h]$).

Measurements include elapsed solution time, the manipulated variable values, a collection of measurements which would be typical for this type of plant, and a larger set of variables which normally are not or cannot be measured but which may be informative.

Not all simulations reach a new steady state; some simulations terminate prematurely, but for reasons which would cause a plant shutdown (such as reaction extinction or column flooding). This unstable behaviour results in a variable length for each experiment—they range from zero to 201 data points.

In total, each time series comprises 1131 variables, with 14 being manipulated and the rest representing the process state. Alongside this "clean" data, realistic measurement noise models are provided. In addition to the 14 manipulated variables, 8 state variables were identified to be of particular importance (the tags of the variables in the dataset are given in parentheses):

(A) The mass flow of the methanol distillate stream (streamsref15a.fm)

(B) The methanol concentration in the methanol distillate stream (streamsref15a.zmnmeoh)

(C) The circulator volumetric flow rate at the inlet (blockscirc.fvin)

(D) Reactor top temperature (streamsr1.t)

(E) The knockout pot holdup (blocksko1.level)

(F) The reboiler holdup (blocksrefcol.stage76.level)

(G) The methanol slip in the bottoms stream (streamsref20a.zmnmeoh)

(H) Reactor bottom temperature (streamsr6.t)

Henceforth, these variables shall be referred to as the **quantities of interest (QoI)**. The manipulated variables can be known at all times, giving a somewhat unusual predictive situation when compared with a classic time series dataset; this isn't a direct forecasting problem defined by the previous state alone. Rather, it could be treated as being closer to a regression problem, attempting to recover a future state for the non-manipulated variables given a known future state for a subset (i.e. the manipulated variables).

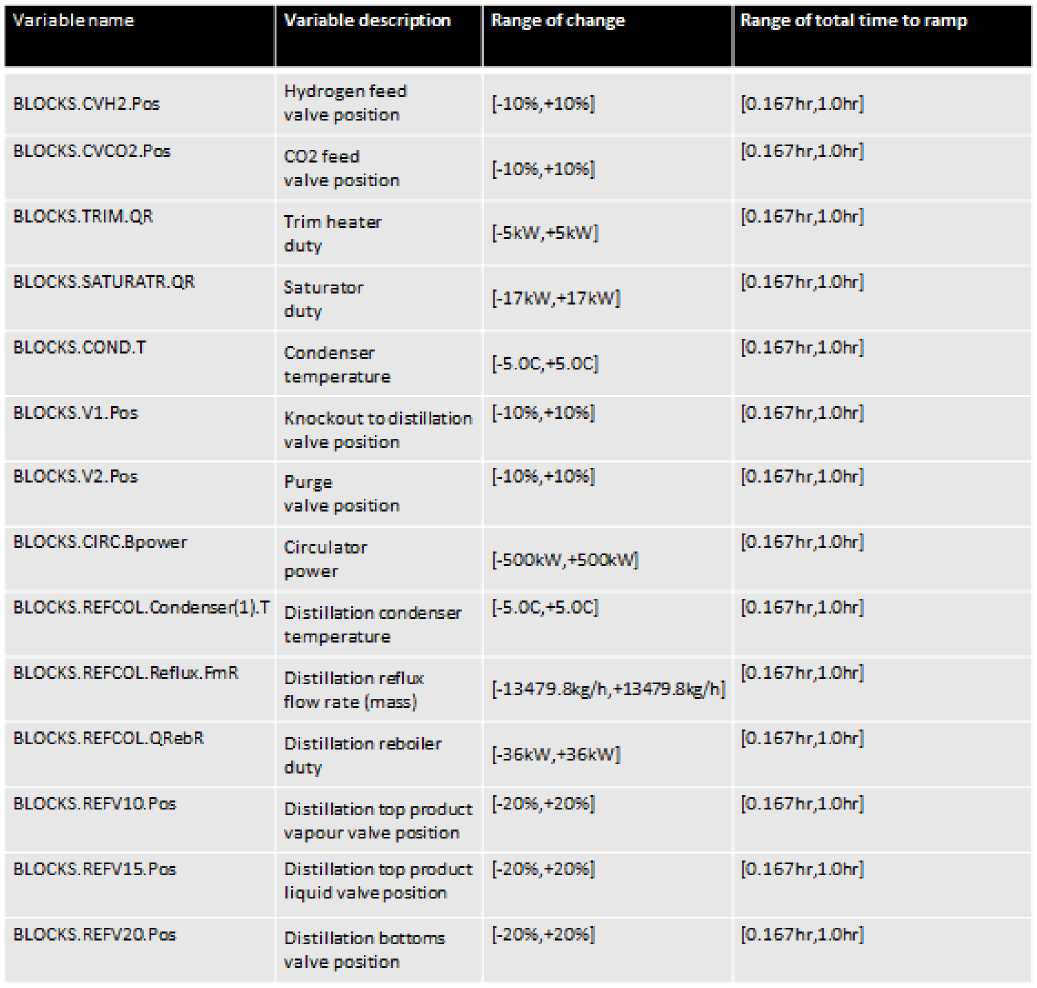
The experiments were generated using a Latin hypercube design, following the maximum projection (MaxPro) criterion [5]. The ranges of variable changes from the initial state are depicted in Figure 1.

The variables in the dataset are categorised into six tiers based on their availability (e.g., typically measured, sometimes measured, possible to measure), as labelled by the challenge owner. Out of these, 80 variables are commonly available across the groups. The distribution of these variable types is illustrated in Figure 3. The top 40 variables that exhibit the greatest variation in the data among the 1131 variables are visualised in Figure 2.

2.2 Data quality issues

Once the data were extracted from the data store, two key issues regarding their quality were identified:

- Two of the 344 experiments contained no data, indicating that they had not been run. This issue was resolved by removing these experiments from the dataset. However, we are unsure what the intended experiments are, so the impact of this omission on the overall richness of the dataset remains unclear.
- The base dataset is without noise. This is not necessarily a problem



Variable name	Variable description	Range of change	Range of total time to ramp
BLOCKS.CVH2.Pos	Hydrogen feed valve position	[-10%,+10%]	[0.167hr,1.0hr]
BLOCKS.CVCO2.Pos	CO2 feed valve position	[-10%,+10%]	[0.167hr,1.0hr]
BLOCKS.TRIM.QR	Trim heater duty	[-5kW,+5kW]	[0.167hr,1.0hr]
BLOCKS.SATURATR.QR	Saturator duty	[-17kW,+17kW]	[0.167hr,1.0hr]
BLOCKS.COND.T	Condenser temperature	[-5.0C,+5.0C]	[0.167hr,1.0hr]
BLOCKS.V1.Pos	Knockout to distillation valve position	[-10%,+10%]	[0.167hr,1.0hr]
BLOCKS.V2.Pos	Purge valve position	[-10%,+10%]	[0.167hr,1.0hr]
BLOCKS.CIRC.Bpower	Circulator power	[-500kW,+500kW]	[0.167hr,1.0hr]
BLOCKS.REFCOL.Condenser(1).T	Distillation condenser temperature	[-5.0C,+5.0C]	[0.167hr,1.0hr]
BLOCKS.REFCOL.Reflux.FmR	Distillation reflux flow rate (mass)	[-13479.8kg/h,+13479.8kg/h]	[0.167hr,1.0hr]
BLOCKS.REFCOL.QRebR	Distillation reboiler duty	[-36kW,+36kW]	[0.167hr,1.0hr]
BLOCKS.REFV10.Pos	Distillation top product vapour valve position	[-20%,+20%]	[0.167hr,1.0hr]
BLOCKS.REFV15.Pos	Distillation top product liquid valve position	[-20%,+20%]	[0.167hr,1.0hr]
BLOCKS.REFV20.Pos	Distillation bottoms valve position	[-20%,+20%]	[0.167hr,1.0hr]

Figure 1: Ranges of the manipulated variables found in the dataset of simulated experiments.

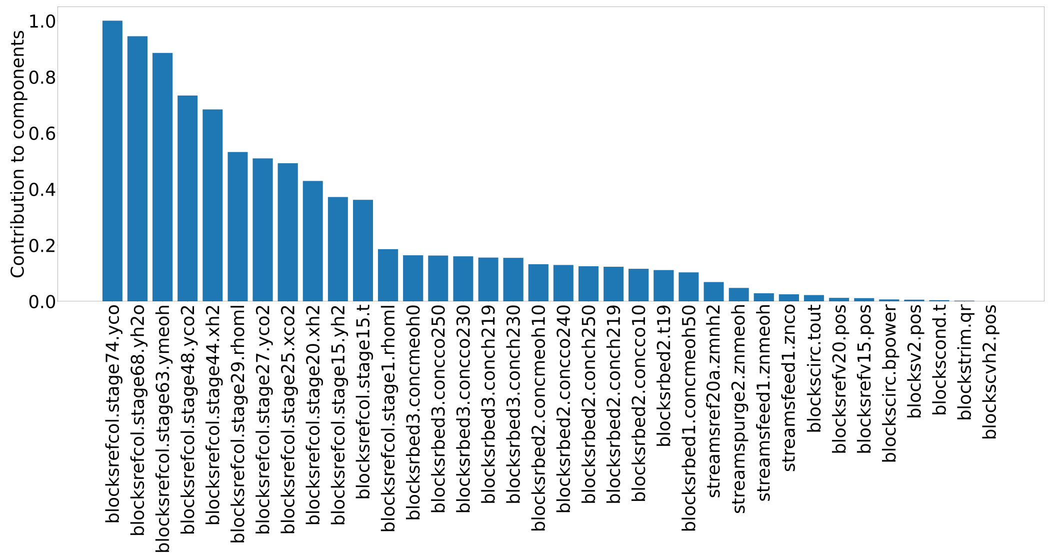


Figure 2: The 40 variables exhibiting the greatest variation in the data among the 1131 variables.

in and of itself; simple normal random noise profiles were provided. A noisy dataset was generated using a noising script and by creating a new, analogous data store containing noisy data.

3 Data exploration and pre-processing

This section describes our initial exploratory data analysis and pre-processing techniques. The feature processing steps are exploratory, though could be used to engineer a reduced set of features in the future. The train/test split is applied in the subsequent analyses, and the sliding window technique, as discussed in Section 3.3, is applied for time series models.

3.1 Feature processing

Overall, the data set for this study includes a total of 1131 time-indexed variables for analysis. As such, it is important that features should be analysed, particularly from the standpoint of whether dimensionality

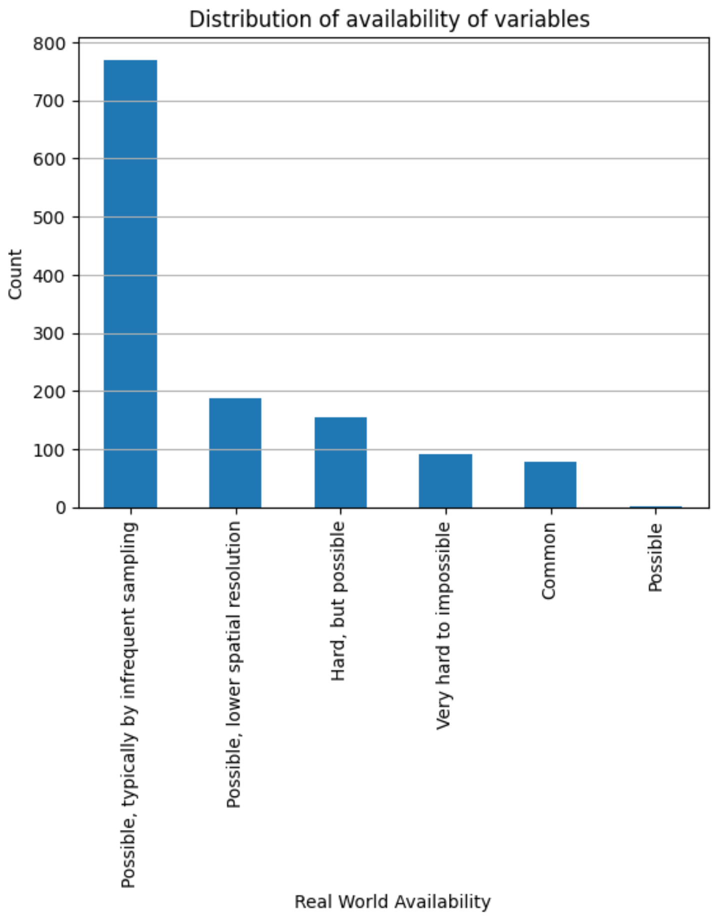


Figure 3: Distribution of state variables categorised into six availability groups. This can be used to inform which variables should be prioritised into the state variables when modelling.

reduction can be employed, e.g. how much correlation lies among the measured variables. Having fewer dimensions to consider implies less computational expense for equivalent models and perhaps improved orthogonality/independence among variables. Additionally, selecting features itself provides insight into the most important aspects of the underlying dynamic response exhibited by the process.

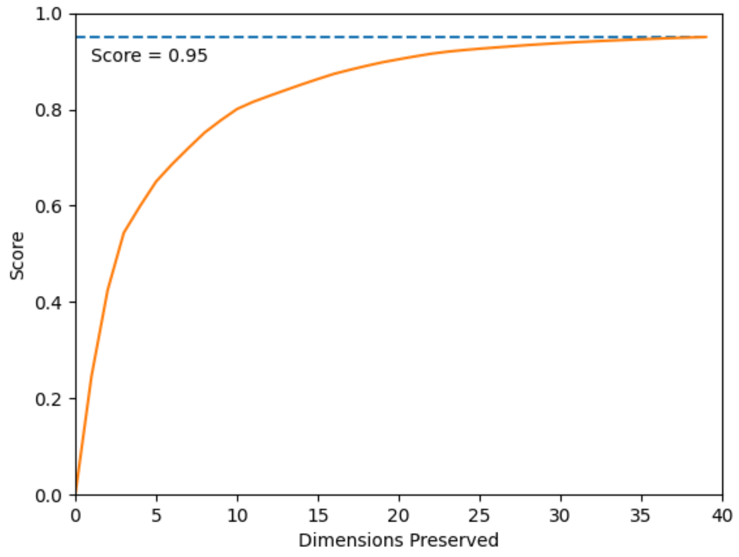
There are a wide variety of techniques for dimensionality reduction, feature selection, and feature engineering, ranging from relatively simple linear methods to much more advanced and non-linear approaches. This report is not intended as a comprehensive study of feature selection, so only a small selection of methods have been considered: principal component analysis (PCA) and correlation analysis of all variables with QoI variables. The interested reader is referred to [7, 18] for more complete discussions of dimensionality reduction and feature selection, respectively.

3.1.1 Principal component analysis

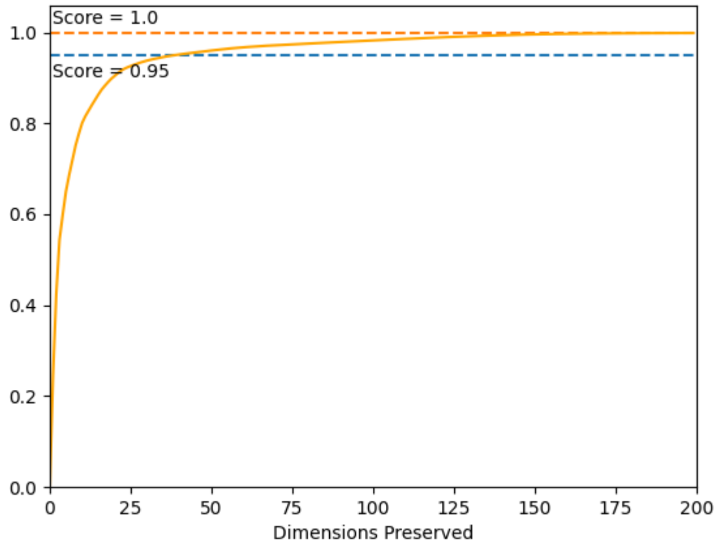
Principal component analysis, or PCA, composes the data into "principal components," by seeking a linear and orthogonal projection of the data that maximises the remaining variance in the data for each new component (essentially by rotation). PCA is an appealing method for two reasons:

- **Cheaper to compute than correlations:** The computation of pairwise correlations among 1131 features results in a large, combinatorial number of correlations. This not only demands significant computational resources but also extends the processing time substantially. PCA is significantly cheaper by comparison.
- **Simple to interpret, even with high-dimensional data:** Interpreting a high volume of pairwise correlations presents substantial challenges. There is a significant risk of missing out on potentially important correlations due to the very large size of the correlation results. By comparison, PCA generates components with simple-to-interpret loading vectors.

Figure 4 shows that a significant portion ($\sim$95%) of the variance in the time series data can be explained by the first 40 principal components. This projection represents a sizeable reduction in



(a) Retaining 40 features



(b) Retaining 200 features

Figure 4: Scores found during PCA for various numbers of principal components. The first 40 principal components correspond to an explained variance ratio exceeding 95%, while 200 principal components capture essentially 100% of the observed variance.

dimensionality—impressive for a linear method on data which is generated by a non-linear process. In many situations, the 95% can be sufficient, but due to the non-linearity of the response, some important behaviour(s) can still be lost from this approximation so we should endeavour to capture more variance. To this end, it can be seen that using 200 principal components captures essentially 100% of the variance and still represents an approximate 80% reduction in the dimensionality of the data. Further dimensionality reduction could potentially be achieved in the nonlinear time series data using non-linear techniques [17].

3.1.2 Correlation analysis of all variables with QoI variables

We perform a correlation analysis of the Quantities of Interest (QoI) variables, introduced in subsection 2.1, against all other variables in the dataset. This analysis is crucial for identifying strong relationships and potential predictors in the dataset.

Table 1 summarises the number of features that exhibit significant correlations with the QoI variables. These correlations are categorised into three thresholds: greater than 0.6, 0.65, and 0.7. Additionally, the 'Slice[16-82]' column in the table specifically highlights the count of highly correlated variables within a crucial subset, or "slice," of the data, providing insights into which variables are most relevant in this specific slice. This slice corresponds to variables that are commonly measured and thus especially likely to be available in reality.

A more detailed visual representation of these correlations is provided in Figures 20–27 in the Appendix. Each bar chart in the Appendix illustrates the absolute correlation values between one QoI variable and other features, sorted in descending order. This visualisation aids in understanding the strength and significance of important pairwise correlations, offering a clearer view of the intricate relationships and dependencies among the variables. Such insights are instrumental for informed data analysis and effective model building.

Table 1: Number of features with correlations exceeding 0.6, 0.65, and 0.7 with QoI variables. The Slice[16-82] refers to the data slice where common variables, which are available and accessible, are located, and the column shows the number of the highly correlated variables in this slice.

Feature	Corr > 0.6	Corr > 0.65	Corr > 0.7	Slice[16-82]
streams.ref15a.fm	1	1	1	1
streamsref15a.zmnmeoh	170	161	54	4
blockscirc.fvin	240	159	55	10
streamsr1.t	31	24	22	6
blocksko1.level	1	1	1	1
blocksrefcol.stage76.level	79	18	16	3
streamsref20a.zmnmeoh	236	132	66	3
streamsr6.t	11	8	6	4

3.2 Train/Test data split

Two versions of the dataset are provided: one without measurement noise and one with random noise representative of real-world instrumentation added by the challenge owner.

The 342 experiments and their associated data (without added noise) were manually partitioned into training (denoted as 'Train_no_noise' and testing (denoted as 'Test_no_noise') datasets. These data are split such that the training and testing datasets contain similar numbers of incomplete experimental runs. An approximate 80%/20% train/test split was sought, resulting in 68 of the 342 experiments being selected as the testing dataset. This test dataset is retained as an out-of-sample test set; at no point is it available to any models during training phases.

The motivation of this is simple: it is important to establish whether the underlying dynamic phenomena have been captured in a generalisable way. Additionally, out-of-sample testing represents a much greater challenge than a validation test set drawn from the sample training pool of data. An identical splitting procedure was followed for the noisy dataset. They are denoted as 'Train_noise' and 'Test_noise' in this report.

3.3 Sliding window approach

The *sliding window* technique involves forming a 'window' of preceding data points at each time instance where predictions are desired. This window comprises a predetermined number of historical observations of the target (i.e., the auto-regressive component) and exogenous variables (i.e., external inputs), as illustrated in Figure 5.

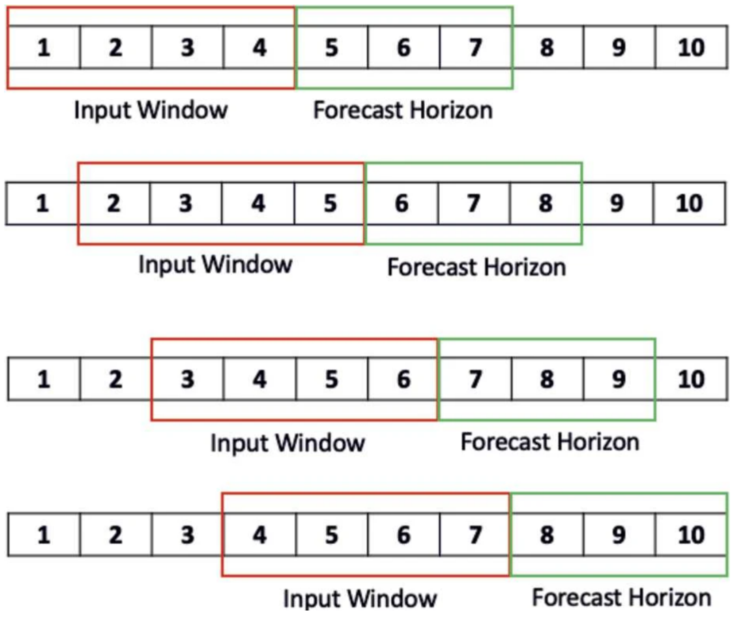


Figure 5: Depiction of the sliding window technique across a time series of length 10, with each index symbolising a discrete time step, adapted from [2]. The input window progresses one step incrementally, creating distinct input-output sets for forecasting at each step of the sequence.

The size of the window that 'slides' across the time series is a modelling decision, which would be ideally selected by some hyper-parameter tuning method such as Bayesian optimisation or through insight into the problem domain of the process. However, for this initial study, a window size of 10 was manually chosen, as 6 minutes of data were heuristically judged to be a sufficiently long window to observe dynamics for the response.

The prediction horizon can also be a variable length and, in practice, initial modelling focuses on the one-step ahead situation, i.e., making predictions for $t+1$. We can define a sliding window with respect to a non-linear model, f for one-step ahead as follows. Given a time series of inputs $\{u(t)\}$ and outputs $\{y(t)\}$:

$$\hat{y}(t+1) = f\left(y(t), y(t-1), \ldots, y(t-n+1), u(t), u(t-1), \ldots, u(t-n+1)\right),$$

where $\hat{y}(t+1)$ is the predicted output at time $t+1$, $y(t), y(t-1), \ldots, y(t-n+1)$ are the past outputs within the window, $u(t), u(t-1), \ldots, u(t-n+1)$ are the past inputs within the window, and n is selected window width.

For multi-step-ahead predictions, the function can be extended iteratively using predicted values of the output:

$$\hat{y}(t+k) = f\left(\hat{y}(t+k-1), \ldots, \hat{y}(t+k-n), u(t+k-1), \ldots, u(t+k-n)\right).$$

The value of k (i.e., k-step-ahead prediction) in this study has been tested up to 5. This seemed a reasonably long time frame relative to the observed state over which to make predictions in this study.

4 Methodology

Experiments conducted as part of this study attempt to make predictions of forecast conditions based on previous states, a problem which can be expressed formally thus:

Let $y_{t:t_0} = (y_t, y_{t+1}, \ldots, y_{t_0})$ be a time series of d-dimensional observations $y_t, \ldots, y_{t_0} \in \mathbb{R}^d$ that start at time step t and end at time step t_0. A multi-horizon time series forecast predicts future values,

$$\hat{y}_{(t_0+1):(t_0+H)} = (\hat{y}_{t_0+1}, \ldots, \hat{y}_{t_0+H}) \in \mathbb{R}^{H \times d}, \qquad (1)$$

given the history of observed values $y_{1:t_0}$. Here, H is the number of steps to be predicted (the prediction horizon).

For the process presented in this study, our focus is restricted to the most important variables with respect to optimal operation. As such, while all of the states are used as input data, prediction is only undertaken for the designated output variables. Let f define a function that behaves as our predictive model for the time series data,

$$f : \mathbb{R}^{H \times d} \to \mathbb{R}^{M \times n}$$

With this division of the process variables, we can specify the prediction task as estimating the behaviour of the output variables, $y'_i \in \mathbb{R}^n$ from the state variables $x_i \in \mathbb{R}^{d-n}$:

$$\hat{y'}_{(t_0+1):(t_0+H)} = f(\hat{y}_{t_0+1}, \ldots, \hat{y}_{t_0+H}; \hat{x}_{t_0+1}, \ldots, \hat{x}_{t_0+H}) \in \mathbb{R}^{H \times d} \quad (2)$$

In general, mathematical models can be evaluated and compared based on metrics such as their prediction accuracy, i.e., quantifying the errors between $\hat{y}$ and y. A popular metric is the root mean squared error, or RMSE. Moreover, models can be evaluated based on their ability to quantify uncertainty; however, this is less straightforward as not all models may include uncertainty estimates.

4.1 Dynamic models

4.1.1 Dynamic mode decomposition

Dynamic Mode Decomposition (DMD) is a data-driven, equation-free method for extracting key dynamic characteristics from measured data in complex systems. It reveals insights such as unstable growth modes, resonance, and spectral properties, making it valuable for understanding system behaviour in numerical simulations or experiments. The earliest development of the DMD was in the context of solving fluid dynamics problems [12, 14].

DMD relies on proper orthogonal decomposition (POD), leveraging the computationally efficient singular value decomposition (SVD). Unlike SVD and POD, DMD offers a modal decomposition, where each mode encompasses spatially correlated structures exhibiting consistent linear behaviour(s) over time. Hence, DMD not only facilitates dimensionality reduction through a reduced set of modes but also furnishes a model depicting the temporal evolution of these modes.

Specifically, DMD constructs the best-fit linear operator $\mathbf{A} \in \mathbb{R}^{n \times n}$ that approximately describes the dynamics of a high-dimensional, nonlinear, spatio-temporal system from pairs of measurements $\boldsymbol{x}_k \in \mathbb{R}^n$ and $\boldsymbol{x}_{k+1} \in \mathbb{R}^n$, such that

$$\boldsymbol{x}_{k+1} \approx \mathbf{A}\boldsymbol{x}_k, \tag{3}$$

where k indicates the temporal iteration from a discrete dynamical system, and n denotes the number of state variables.

Various adaptations of DMD have been proposed [13] to enhance its versatility across diverse systems. In addressing the specific challenge at hand, featuring both control and state variables, the application of DMD with control [6, 10] has proven valuable. This suggests potential avenues for future research, exploring additional extensions of DMD to further tackle the complexities posed by the problem.

4.1.2 Dynamic mode decomposition with control

Dynamic mode decomposition with control, or DMDc, is an extension of DMD that integrates dimensionality reduction and model discovery, specifically designed for systems involving actuation or control inputs [6, 10]. Notably, DMDc exhibits compatibility with model predictive control, showcasing its versatility in addressing complex dynamic systems.

Specifically, DMDc is an algorithm designed to identify optimal linear operators $\mathbf{A}$ and $\mathbf{B}$ that approximately satisfy the equation:

$$\boldsymbol{x}_{k+1} \approx \mathbf{A}\boldsymbol{x}_k + \mathbf{B}\boldsymbol{u}_k, \tag{4}$$

where $\boldsymbol{x}_k \in \mathbb{R}^n$ represents a vector of state variables and $\boldsymbol{u}_k \in \mathbb{R}^l$ is a vector of control variables, where l is the number of control variables. The operator $\mathbf{A} \in \mathbb{R}^{n\times n}$ characterises the inherent dynamics of the system, while $\mathbf{B} \in \mathbb{R}^{n\times l}$ captures dynamics influenced by control.

As input, DMDc necessitates three data matrices derived from m time snapshots of state and control vectors:

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \boldsymbol{x}_1 & \boldsymbol{x}_2 & \dots & \boldsymbol{x}_{m-1} \\ | & | & & | \end{bmatrix}, \tag{5a}$$

$$\mathbf{X}' = \begin{bmatrix} | & | & & | \\ \boldsymbol{x}_2 & \boldsymbol{x}_3 & \dots & \boldsymbol{x}_{m} \\ | & | & & | \end{bmatrix}, \tag{5b}$$

$$\mathbf{U} = \begin{bmatrix} | & | & & | \\ \boldsymbol{u}_1 & \boldsymbol{u}_2 & \dots & \boldsymbol{u}_{m-1} \\ | & | & & | \end{bmatrix}. \tag{5c}$$

Expressed in terms of these data matrices, equation (4) can be re-written as:

$$\mathbf{X}' \approx \mathbf{AX} + \mathbf{BU}. \tag{6}$$

Once the matrices $\mathbf{A}$ and $\mathbf{B}$ are determined, predictions of future state variables are generated given values of the control inputs. Additionally, conducting a modal analysis of matrix $\mathbf{A}$—achieved through its singular value decomposition—offers insights into the key variables essential for describing the time evolution of the process' inherent dynamics.

4.1.3 Long short-term memory (LSTM) neural networks

Long Short-Term Memory (LSTM) networks [4], illustrated in Figure 6, are a type of recurrent neural network (RNN) particularly useful for learning order dependence in sequence prediction problems. While LSTMs are effective at capturing long-range dependencies, they can be computationally intensive and may suffer from over-fitting, especially with smaller datasets. For a more intuitive explanation of LSTM networks and their applications, readers are referred to the blog post [9].

We apply the above sliding window of size 10 to the time series data for both training and testing. As before, this window is composed of both the control variables and the past history values for the output variables. The LSTM model is structured with three layers, each consisting of 50 units. The first two LSTM layers are configured to return sequences, thus passing the whole sequence of predictions to the next layer. The final

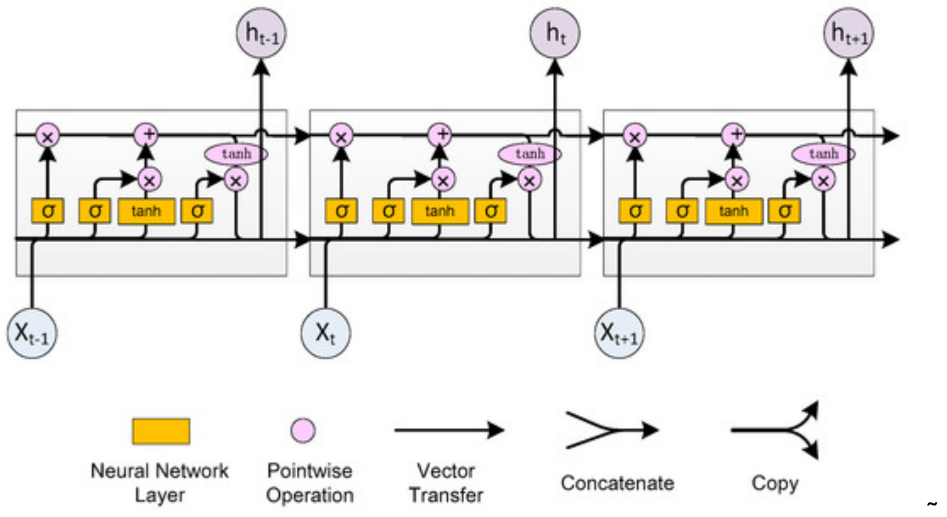


Figure 6: Illustration of LSTM cell processing sequence data at different time steps (t-1, t, t+1), reproduced from [20].

LSTM layer does not return sequences and is fed directly into a dense layer. The dense layer, with 16 units, serves as the output layer of the model, corresponding to the structured prediction task, i.e., predicting values for the 8 output variables. The entire model is trained with the Adam optimiser and mean squared error loss function.

4.1.4 Non-linear auto-regressive exogenous (NARX) models

The Non-linear Auto-regressive exogenous (NARX) model is a powerful tool for modelling complex non-linear processes influenced by external inputs. Figure 7 shows the architecture of a NARX model.

The network consists of several layers: an input layer, some hidden layer(s), and an output layer. In a NARX model, the input layer receives the time-lagged values of both the external inputs and the output state. The hidden layer then captures the non-linear relationships between inputs and prior state values. Finally, the output layer produces the prediction for the current time step. Specifically, the NARX model can be mathematically represented as:

$$y(t) = f\left(y(t-1), y(t-2), \ldots, y(t-n_y), x(t), x(t-1), \ldots, x(t-n_x)\right),$$

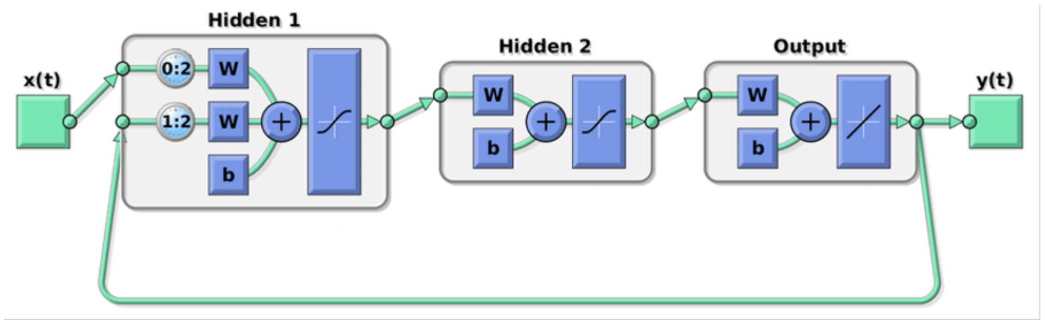


Figure 7: The structure of a NARX model. Here, $x(t)$ are the inputs in the process at time t and $y(t)$ are the outputs at time t. Reproduced from [11].

where $y(t)$ is the current output, $x(t)$ is the current external input, f represents the non-linear mapping function learned by the network, n_y is the number of auto-regressive terms (past output values), and n_x is the number of exogenous input terms.

The hidden layer output at time t for the NARX network is computed as follows. At time t, the hidden layer output for the NARX network, denoted as $M_i(t)$, is calculated by combining both past input values and past network outputs, which are then fed through an activation function f_1. This process is mathematically represented as:

$$M_i(t) = f_1\left(\sum_{r=0}^{n_x} w_{ir}x(t-r) + \sum_{l=1}^{n_y} w_{il}y(t-l) + a_i\right), \tag{7}$$

where w_{ir} and w_{il} are the weights assigned to past inputs and outputs respectively, and a_i is a bias term. The network's final output at time t, $\hat{y}_j(t)$, is then determined by applying a second activation function f_2 to the weighted sum of the hidden layer outputs plus a bias term b_j, which can be expressed as:

$$\hat{y}_j(t) = f_2\left(\sum_{i=1}^{n_h} w_{ji}M_i(t) + b_j\right), \tag{8}$$

where w_{ji} is the weight corresponding to the i-th hidden neuron's contribution to the j-th output. The model's performance is evaluated using the root mean square error (RMSE) loss function.

In our preliminary investigation, we use the NARX model with 14 input variables and a single output variable. While the initial model demonstrated promising predictive capabilities, it struggled to accurately capture the discontinuities between multiple time series; specifically, the transitions between the end (‘tail’) of one trajectory and the beginning (‘head’) of the next. This issue was partially mitigated using the sliding-window approach.

The NARX model is then trained for the eight QoI outputs. We employ a hidden layer composed of 64 neurons employing the ReLU activation function. In addition, the NARX model is configured to provide predictions from one to five steps ahead, allowing for direct comparison with the LSTM model.

4.1.5 Transformers

Transformers are a type of neural network architecture introduced in the groundbreaking paper, “Attention Is All You Need” [19]. They revolutionised natural language processing tasks and are promising for various sequential prediction problems. There are three core parts central to the architecture of transformer networks. First is the self-attention mechanism which allows the model to apply weighting across the input sequence when making predictions. This mechanism enables relationships among different elements within the sequence to be captured, without the need to rely on recurrence or convolution. Next is the encoder-decoder structure where the encoder processes the input into representations, while the decoder generates output from these representations. The final feature is positional encoding. As transformers lack inherent information about sequence order, positional encodings are added to the input embeddings to provide the model with a sequence structure.

However, standard transformer models are unlikely to be a perfect fit to the problem in this challenge. The attention mechanism gives every variable in a time step the same attention, disregarding any relative importance. This means the model focuses mainly on changes across the time domain, so it might miss out on important changes between the state variables within a time step. For instance, it might not capture cases of stark contrast in state variable directions. Also, when we’re dealing with time series data, attention scales poorly. As the series grows, the computational cost can grow quickly, depending on the specific architecture [16]. This makes it tricky to handle long sequences of data.

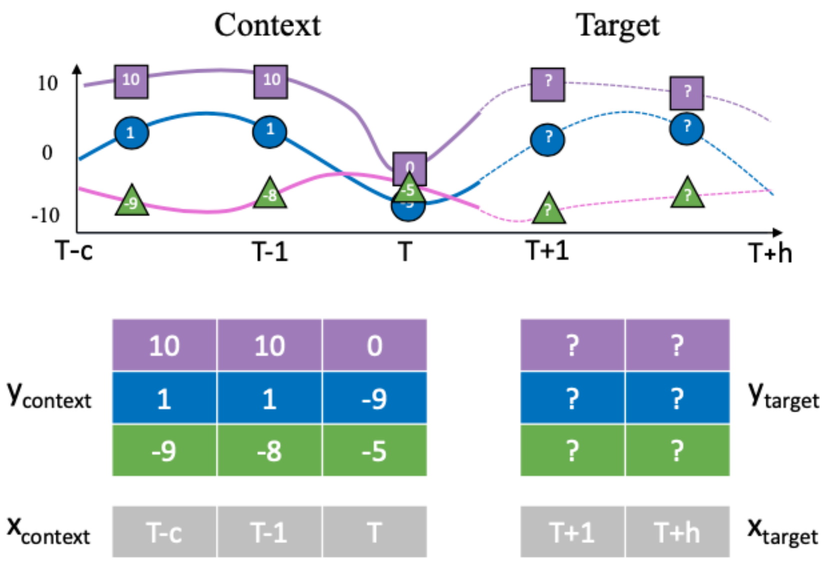
If the basic transformer model is directly applied to this problem, it might be assumed that performance is similar to LSTM as a result of focusing largely on the time domain. Given transformer models are computationally expensive, it is important that the limitations of the base architecture are first addressed. To this end, learnable positional encoding and ProbSparse Attention are proposed, as they allow the transformer model to be more flexible and better exploit sequential ordering information. This will enable the attention layer to calculate weights and probabilities using the most important data points only.

While many new transformer architectures such as LogTrans, Pyraformer, and FEDformer incorporate the above modifications, a Spacetimeformer model has been used here due to the open-source availability of code. Spacetimeformer [3], addresses the above problems by translating the problem of multivariate forecasting into a "spatio-temporal" sequence formulation, where each transformer input token represents the value of a single variable at a given time. Long-range transformers can then learn interactions between space, time, and variable information jointly along this extended sequence. As such this model is built to handle multivariate "sequence-to-sequence" problems such as time series forecasting, as shown in Figure 8.

In this architecture, handling long multi-variate sequences with attention can become unwieldy, especially with large amounts of data. To simplify this, we introduce "local" attention within each layer of the encoder and decoder. This means that each piece of information (each variable in this case) pays attention to its own timeline first and then looks at the bigger picture, considering every piece of information across the entire dataset. This improves efficiency of learning, by encouraging the local spacio-temporal phenomena to be considered first, as shown in Figure 9.

Transformers offer a number of advantages for the problem at hand:

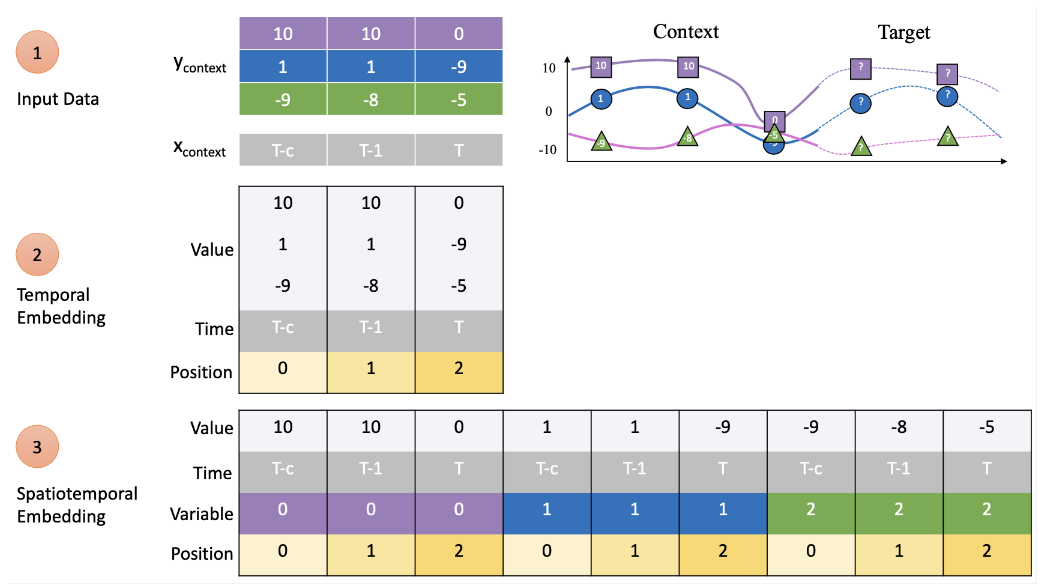
- They are well-suited to capturing long-range dependencies in sequences due to their self-attention mechanism.



	T-c	T-1	T
$y_{context}$	10	10	0
	1	1	-9
	-9	-8	-5

	T+1	T+h
y_{target}	?	?
	?	?
	?	?

Figure 8: Conceptual depiction of Spacetimeformer models [3]. Reproduced from https://github.com/QData/spacetimeformer under the MIT licence.



Value	10	10	0	1	1	-9	-9	-8	-5
Time	T-c	T-1	T	T-c	T-1	T	T-c	T-1	T
Variable	0	0	0	1	1	1	2	2	2
Position	0	1	2	0	1	2	0	1	2

Figure 9: Conceptual depiction of spatio-temporal sequence[3]. Reproduced from https://github.com/QData/spacetimeformer under the MIT licence)

- Parallelism is possible with a transformer network, increasing computational efficiency and speeding the training procedure.
- Transformer networks offer better inter-task generality from the attention mechanism.
- Attention weights can provide improved interpretability compared to recurrent networks as they provide clear insight into the relative importance of state variables within a sequence.

Of course, like all model classes, transformers suffer from their own set of limitations, namely the increased computational resource requirements when compared with LSTMs with equivalent sequence lengths. Additionally, many transformer networks require adaptation to a specific problem, which is time-consuming.

4.2 Quantifying uncertainty using conformal predictions

The conformal prediction method provides a model-agnostic way to quantify prediction uncertainty in the dynamic models. It can be computed quite easily and allows direct assessment as to whether predictions are likely to be trustworthy. The next two subsections define the procedure of conformal prediction.

4.2.1 Conformal Prediction

For a given significance level (error rate) α, the goal of **Conformal Prediction (CP)** is to return a prediction region Γ_α that is guaranteed to contain the true value with probability of at least $(1-\alpha)$. In regression problems such as time series forecasting, CP is modified to work inductively using an additional calibration set, which could be treated as a validation set in machine learning architecture and an underlying model—an approach called **Inductive Conformal Prediction (ICP)**.

4.2.2 Inductive conformal prediction

Given a set of observations $D = \{(x^{(i)}, y^{(i)})\}_{i=1}^{l}$ and a new example $x^{(l+1)}$, the ICP procedure returns a prediction interval Γ_α such that the property of validity is satisfied:

Property 1. (Validity) Under the exchangeability assumption [1], any conformal predictor will return the prediction region $\Gamma_\alpha(x^{(i)})$ such that the probability of error $y^{(l+1)} \notin \Gamma_\alpha(x^{(l+1)})$ is not greater than α. Alternatively:

$$P[y^{(l+1)} \in \Gamma_\alpha(x^{(l+1)})|D] \geq 1-\alpha. \tag{3}$$

The conformal prediction framework is distribution-free (i.e., it does not have any assumptions on the distribution of the underlying data D), and applies to any underlying predictive model as long as the exchangeability assumption is satisfied:

Assumption 1. (Exchangeability) In a dataset of l observations $\{(x^{(i)}, y^{(i)})\}_{i=1}^{l}$, any of its $l!$ permutations are equi-probable. Note that independent identically distributed (or i.i.d.) observations satisfy this assumption. Practically speaking, this assumption states that the order of observations for a group of outcomes does not impact the probability of said outcomes.

The inductive variant of CP operates by splitting the training set into the proper training set of size n and a calibration set of size m: $D = D_{\text{train}} \cup D_{\text{cal}}$. The proper training set is used to train the underlying (auxiliary) model M, and the calibration set is used to obtain the nonconformity scores, which measure how unusual the given example is compared to previously observed data. While CP guarantees validity for any non-conformity score (including a random number generator), the most commonly used non-conformity score in regression is of the form

$$R_i = A(D, (x^{(i)}, y^{(i)})) = \Delta(M(x^{(i)}|D), y^{(i)}), \tag{4}$$

where Δ is some distance metric. When $\Delta(\hat{y}, y) = |\hat{y} - y|$, the non-conformity score $R_i = |\hat{y}^{(i)} - y^{(i)}|$ corresponds to the residual error between the prediction of the underlying model and the true label. The resulting empirical non-conformity score distribution $\{R_i\}_{i=1}^{l}$ is used to compute a critical non-conformity score $\hat{\varepsilon}$, which corresponds to the $\lceil (m+1)(1-\alpha) \rceil$-th smallest residual. For a new example $x^{(l+1)}$, the prediction interval is then:

$$\Gamma_\alpha(x^{(l+1)}) = [\hat{y}^{(l+1)} - \hat{\varepsilon}, \hat{y}^{(l+1)} + \hat{\varepsilon}], \tag{5}$$

with $\hat{y}^{(l+1)} = M(x^{(l+1)})$.

5 Results

5.1 Dynamic mode decomposition

The implementation of the DMDc algorithm for the specified challenge is approached in two distinct ways. Firstly, by considering all time series samples collectively, as discussed in Section 5.1.1. Secondly, by analysing the predictability of DMDc on a per-sample basis, detailed in Section 5.1.2. In both approaches, the state and control variables are segregated. Furthermore, the DMDc algorithm is implemented using the PyDMD Python package (available at https://github.com/PyDMD/PyDMD, accessed in December 2023).

5.1.1 DMDc using all samples

The dataset comprises over three hundred samples of time series data. Despite the wealth of information available, there is limited or no existing literature on effectively handling multiple samples of time series data within the context of DMDc. To address this gap, two distinct approaches were developed. The first approach entails horizontally concatenating the time-series data from all samples to construct comprehensive data matrices, as illustrated below:

$$\mathbf{X} = \begin{bmatrix} | & | & & | & | & | & & | & & | \\ \boldsymbol{x}_1^{(1)} & \boldsymbol{x}_2^{(1)} & \dots & \boldsymbol{x}_{m-1}^{(1)} & \boldsymbol{x}_1^{(2)} & \boldsymbol{x}_2^{(2)} & \dots & \boldsymbol{x}_{m-1}^{(2)} & \dots & \boldsymbol{x}_{m-1}^{(M)} \\ | & | & & | & | & | & & | & & | \end{bmatrix}, \tag{9a}$$

$$\mathbf{X}' = \begin{bmatrix} | & | & & | & | & | & & | & & | \\ \boldsymbol{x}_2^{(1)} & \boldsymbol{x}_3^{(1)} & \dots & \boldsymbol{x}_{m}^{(1)} & \boldsymbol{x}_2^{(2)} & \boldsymbol{x}_3^{(2)} & \dots & \boldsymbol{x}_{m}^{(2)} & \dots & \boldsymbol{x}_{m}^{(M)} \\ | & | & & | & | & | & & | & & | \end{bmatrix}, \tag{9b}$$

$$\mathbf{U} = \begin{bmatrix} | & | & & | & | & | & & | & & | \\ \boldsymbol{u}_1^{(1)} & \boldsymbol{u}_2^{(1)} & \dots & \boldsymbol{u}_{m-1}^{(1)} & \boldsymbol{u}_1^{(2)} & \boldsymbol{u}_2^{(2)} & \dots & \boldsymbol{u}_{m-1}^{(2)} & \dots & \boldsymbol{u}_{m-1}^{(M)} \\ | & | & & | & | & | & & | & & | \end{bmatrix}. \tag{9c}$$

Here M represents the number of time-series samples.

This method posed compatibility challenges with pyDMD, as the library expects the available data to originate from a single sample. More precisely, it mandates the input of all data necessary to construct the matrices $\mathbf{X}$ and $\mathbf{X}'$, which pyDMD subsequently generates automatically. To address this challenge, one potential solution is to hard-code the algorithm. However, an additional complexity arises in data processing due to the uneven time evolution of different time-series samples. This implies that the values of m at each M might not necessarily be consistent.

Considering the time constraints, an alternative approach to handle multiple samples was explored. This method involved vertically appending the time-series data from all samples to construct the data matrices, as illustrated below:

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \boldsymbol{x}_1^{(1)} & \boldsymbol{x}_2^{(1)} & \dots & \boldsymbol{x}_{m-1}^{(1)} \\ | & | & & | \\ \boldsymbol{x}_1^{(2)} & \boldsymbol{x}_2^{(2)} & \dots & \boldsymbol{x}_{m-1}^{(2)} \\ | & | & & | \\ & \dots & & \\ | & | & & | \\ \boldsymbol{x}_1^{(M)} & \boldsymbol{x}_2^{(M)} & \dots & \boldsymbol{x}_{m-1}^{(M)} \\ | & | & & | \end{bmatrix}, \mathbf{X}' = \begin{bmatrix} | & | & & | \\ \boldsymbol{x}_2^{(1)} & \boldsymbol{x}_3^{(1)} & \dots & \boldsymbol{x}_{m}^{(1)} \\ | & | & & | \\ \boldsymbol{x}_2^{(2)} & \boldsymbol{x}_3^{(2)} & \dots & \boldsymbol{x}_{m}^{(2)} \\ | & | & & | \\ & \dots & & \\ | & | & & | \\ \boldsymbol{x}_2^{(M)} & \boldsymbol{x}_3^{(M)} & \dots & \boldsymbol{x}_{m}^{(M)} \\ | & | & & | \end{bmatrix}, \tag{10a}$$

$$\mathbf{U} = \begin{bmatrix} | & | & & | \\ \boldsymbol{u}_1^{(1)} & \boldsymbol{u}_2^{(1)} & \dots & \boldsymbol{u}_{m-1}^{(1)} \\ | & | & & | \\ \boldsymbol{u}_1^{(2)} & \boldsymbol{u}_2^{(2)} & \dots & \boldsymbol{u}_{m-1}^{(2)} \\ | & | & & | \\ & \dots & & \\ | & | & & | \\ \boldsymbol{u}_1^{(M)} & \boldsymbol{u}_2^{(M)} & \dots & \boldsymbol{u}_{m-1}^{(M)} \\ | & | & & | \end{bmatrix}. \tag{10b}$$

This approach introduces challenges with the resulting dimensions of matrices $\mathbf{A}$ and $\mathbf{B}$. Instead of $\mathbf{A}$ being an $n \times n$ matrix, as expected, it manifests as $(n \times M) \times (n \times M)$, where n represents the number of state variables. Similarly, the matrix $\mathbf{B}$ takes the form of $(n \times M) \times (l \times M)$, where l denotes the number of control variables. The applicability of such dimensions to new (test) data becomes unclear, especially when the number of samples in the test set is generally smaller than the training set.

5.1.2 DMDc using independent samples

As a response to the challenges outlined earlier, the DMDc algorithm was applied to individual samples one at a time. The primary objective was to conduct a modal analysis on the matrix $\mathbf{A}$ to explore the presence of POD modes in each sample. While some modal analysis was conducted before using the matrix $\mathbf{A}$ to reconstruct the data employed in its calculation (the training data). It became apparent at this stage that the reconstruction was notably deficient. In other words, $\mathbf{A}$ failed to capture the underlying dynamics. An illustrative example of this unsuccessful reconstruction is presented for four randomly chosen state variables in Figure 10. The four predicted state variable trajectories all ‘blow up’ towards positive or negative infinity after approximately 0.3 hours.

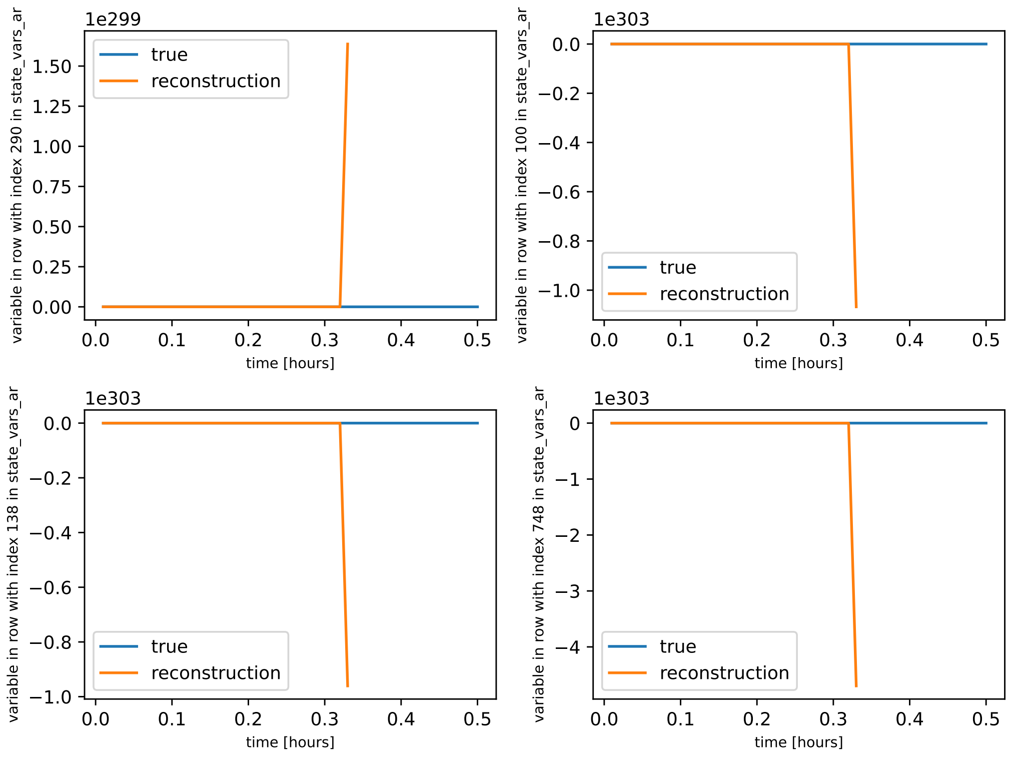


Figure 10: Comparison between the original data (labelled ‘true’) and the reconstruction obtained using the DMDc model for four randomly chosen state variables. The scale of the y-axis masks the fact that at no point does the reconstructed data overlap with the original.

5.2 Long short-term memory networks

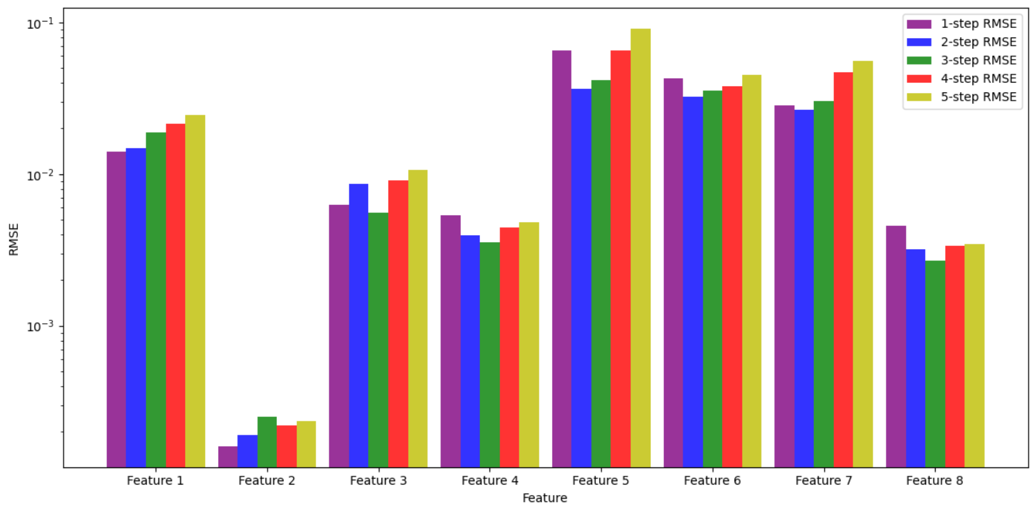


Figure 11: Estimation of RMSE of each QoI with different prediction horizons from the LSTM model.

5.2.1 Impact of measurement noise

The robustness of the LSTM model against measurement noise is important when looking to extend the application of the model beyond simulation data, which is clean. To examine robustness, the LSTM model is trained on both the ‘Train_no_noise’ and ‘Train_noise’ datasets for one-step ahead predictions. The model trained on noise-free data is then used as a direct benchmark for performance by providing a best-case RMSE. Figure 11 compares the RMSE between the models trained on clean and noisy data. For conciseness here we only show predictions for one representative variable, which is ‘streamsref15a.fm’. The LSTM predictions are shown in Figure 12 and Figure 13.

5.2.2 Impact of prediction horizon length

One-step ahead predictions, while valuable in many scenarios, are limiting in general. Instead, greater value can be gained from predicting multiple steps ahead. Due to the inherent nature of the sequence

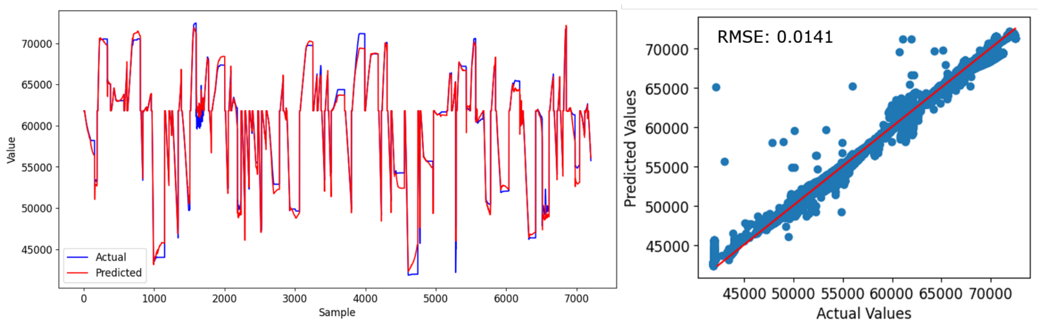


Figure 12: Predictions of LSTM model trained on clean data vs actual values for variable streamsref15a.fm. Left: time series plot. Right: pairwise comparison.

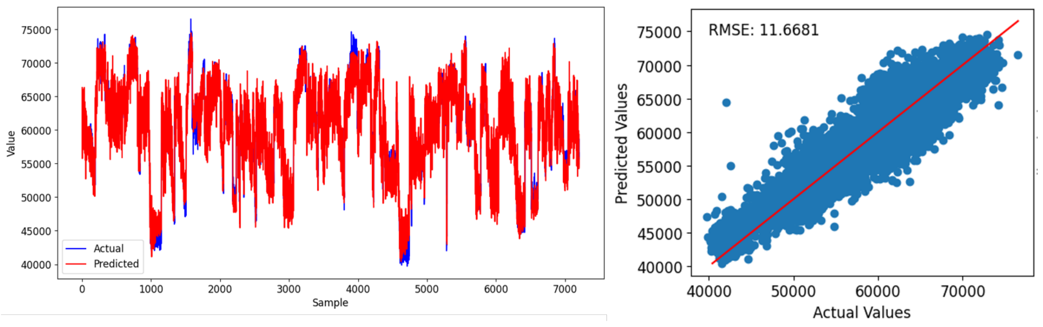


Figure 13: Predictions of LSTM model trained on noisy data vs actual values for variable streamsref15a.fm. Left: time series plot. Right: pairwise comparison.

prediction, the prediction uncertainty propagates with the increase in the number of prediction steps. As such, the accumulation of the uncertainty can seriously impact the prediction accuracy of the LSTM model. Therefore, it is important to test the reliability of the model predictions as we increase the length of the prediction horizon. In this work, this test is carried out by increasing the number of prediction steps from 1 step all the way up to 5 steps. In this case, noise free data is used for all model training and testing. The test results are shown in Figure 11.

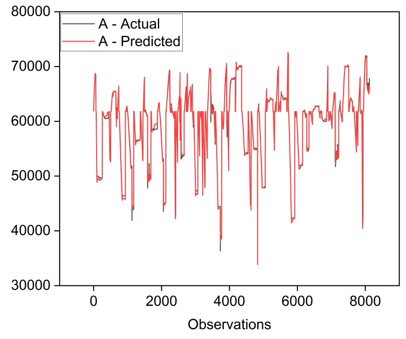
Interestingly, the RMSE does not increase significantly for all of the QoI features as the prediction horizon grows. Features 2,4 and 8 in particular exhibit lower RMSE. Overall, across the QoI variables, the increase is generally acceptable and the largest RMSE (normalised) is below 0.1.

5.3 Non-linear auto-regressive exogenous (NARX) models

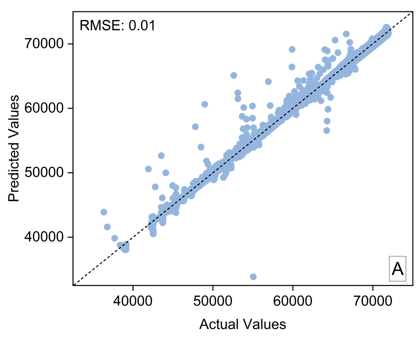
The performance of the multi-input, multi-output (MIMO) NARX model with respect to predictions of the QoI variables is shown in Figure 14. The one-step ahead predictions against the noise-free test dataset are relatively accurate in general, reflected by the un-normalised RMSE of 593. This RMSE value grows to 1609 against the noisy test dataset, as, intuitively, some of the dynamics are liable to be obscured by the introduction of a noise process.

5.3.1 Impact of prediction horizon length

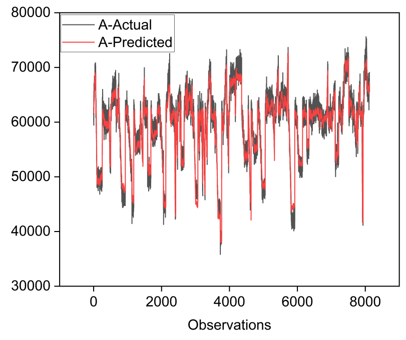
We next investigate the impact of a five-step prediction horizon on the NARX model predictions for the eight output features. The RMSE for these predictions is shown in Figure 15. It is observed that prediction errors for the test dataset generally increase as the prediction horizon increases from one to five. As the prediction horizon increases, the error accumulation can lead to higher prediction errors.



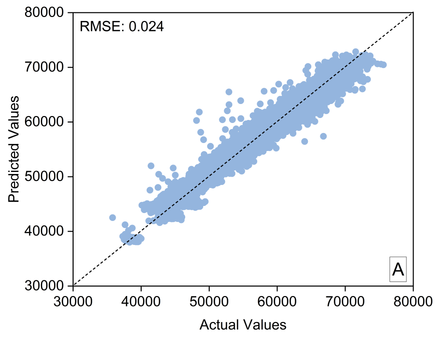
(a) NARX model predictions against clean data.



(b) Pairwise plots of the NARX model predictions against clean data.



(c) NARX model predictions against noisy data.



(d) Pairwise plots of the NARX model predictions against noisy data.

Figure 14: Performance of NARX models trained on clean and noisy datasets for QoI variable ‘A.’

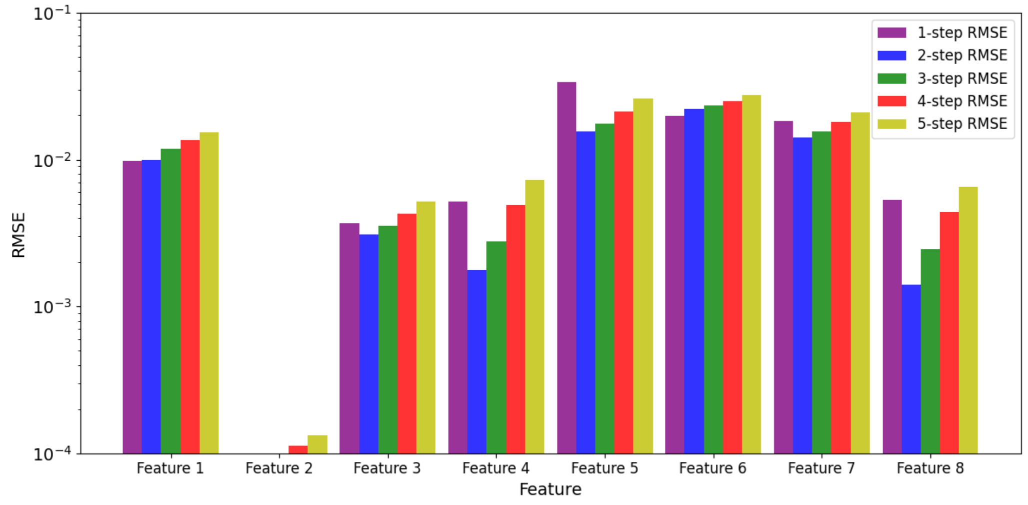


Figure 15: Effect of the length of the 5-steps forward prediction horizon

5.4 Transformers

We faced challenges when implementing a custom dataset and debugging the open-source model for the transformer architecture. The development of a dataset tailored to the model's expectations demanded a meticulous restructuring of data layers, aligning with the hierarchical complexities of the model architecture. The prediction task is set to use the previous 10 values to predict the next 3 values to align with other methods, and the loss calculation and evaluation metrics change accordingly. The initial findings, based on a fraction of the complete dataset of 12 control variables and 8 important variables, hint at promising similarities with LSTM-based models, reflecting normalised MAE (0.053) and MSE (0.033) metrics that foster an optimistic outlook for potential modifications and optimisations.

Besides, deploying the model for full-scale usage encountered unexpected difficulties, p rimarily c entred a round t he computational demands posed by the model architecture. Even with considerable reductions in model complexity (1.8 million parameters)—halving encoding and decoding layers, and minimising attention heads from 10 to 1—the utilisation of GPU resources remained relatively high. This constrained the ability to explore the full potential of the model and impeded seamless deployment within the resource limitations of this study.

Interestingly, among these challenges, an interesting discovery surfaced. The utilisation of a sliding window methodology aligned seamlessly with the requirements of the spacetimeformer model. This approach facilitated the generation of valid input-output pairs, mirroring the model's expectations regarding the context and target points within the dataset. This fortuitous alignment holds promise for the model's compatibility with such data structuring techniques, potentially offering a pathway for further exploration and optimisation in future iterations.

5.5 Conformal Prediction for Uncertainty Quantification

We apply the conformal prediction technique on the trained NARX model to create prediction bounds for the test dataset. The prediction bounds are constructed using 95% confidence intervals for two selected, representative features, which we term 'feature 2' and 'feature 5' here. Snippets of the conformal prediction based bounds for the two features are depicted in Figures 16 and 17.

In our analysis, we apply conformal prediction to time series data, targeting multiple features with a focus on achieving a 95% confidence level in our predictions. The resulting prediction intervals, visualised sequentially, effectively enclose the true values and thus offer a robust and reliable measure of prediction uncertainty. Notably, these intervals were dynamically adjusted around the predicted values, reflecting the inherent variability in the time series. The acceptance rates, indicating the proportion of true values falling within these intervals, varied across different features, ranging from approximately 88% to 95%. This variation underscores the diverse nature of our data and the efficacy of conformal prediction in providing meaningful, confidence-driven insights into our time-based forecasts.

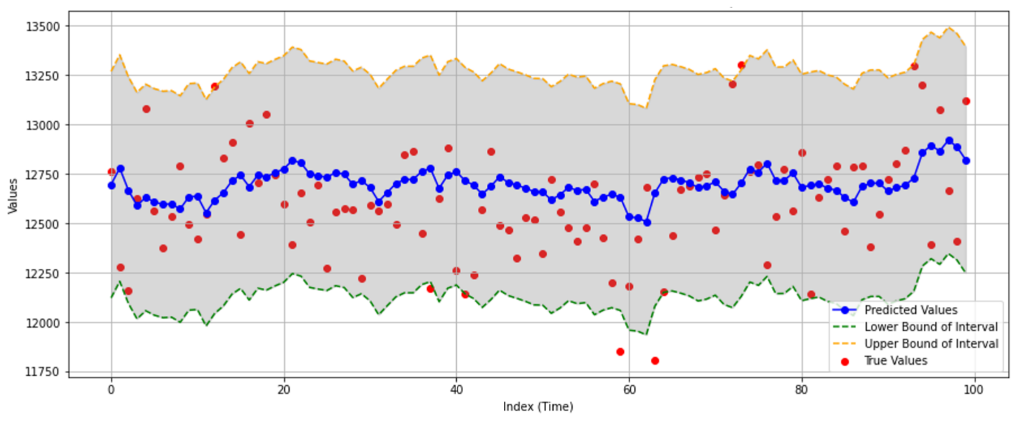


Figure 16: Conformal prediction intervals and true values for ‘feature 2.’

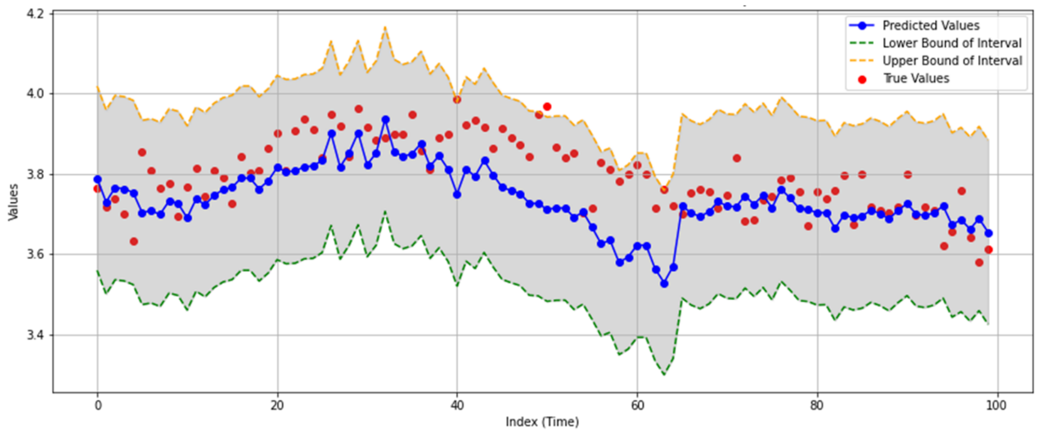


Figure 17: Conformal prediction intervals and true values for ‘feature 5.’

6 Discussion

The DMDc models did not perform well on the given dataset. The failure of DMDc to effectively reconstruct the data may be attributed to several potential factors. Firstly, the presence of strongly non-linear dynamics in some state variables could be a contributing factor, as illustrated in Figure 18.

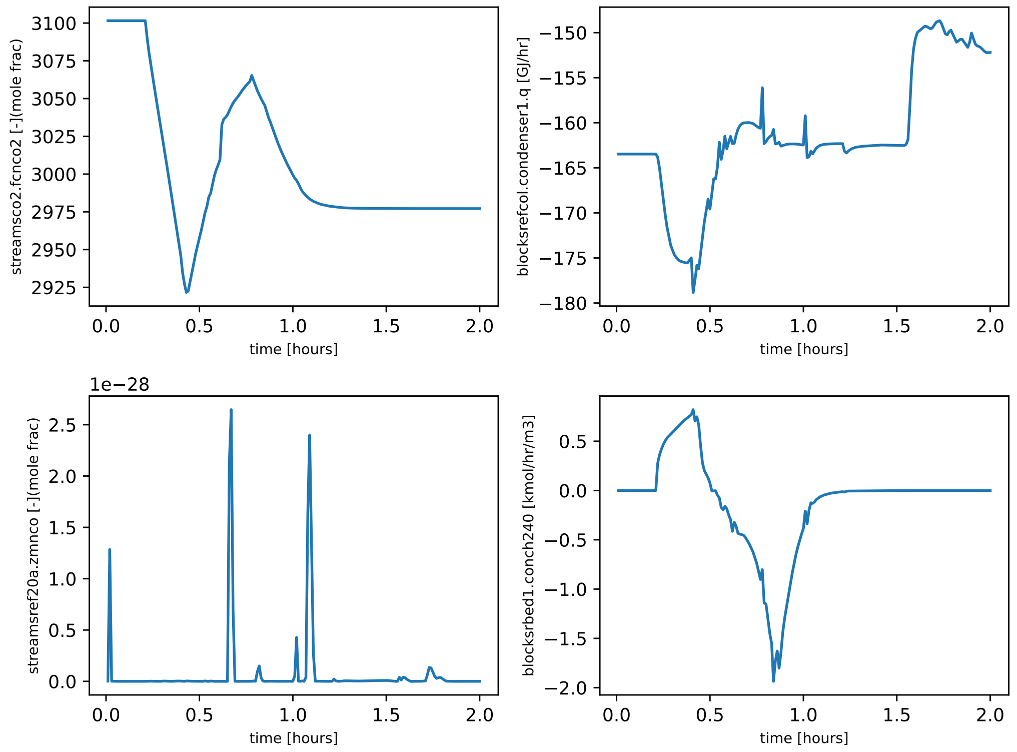


Figure 18: The time evolution of a small sample of the state variables that behave in a strongly nonlinear fashion from the experiment titled "appeal-branch-cabinet-bargain".

Notably, these variables exhibit intricate and non-linear dynamics. In contrast, Figure 19 presents the smoother time evolution of control variables, implying a more gradual progression.

Secondly, there arises a question regarding the precise definition of

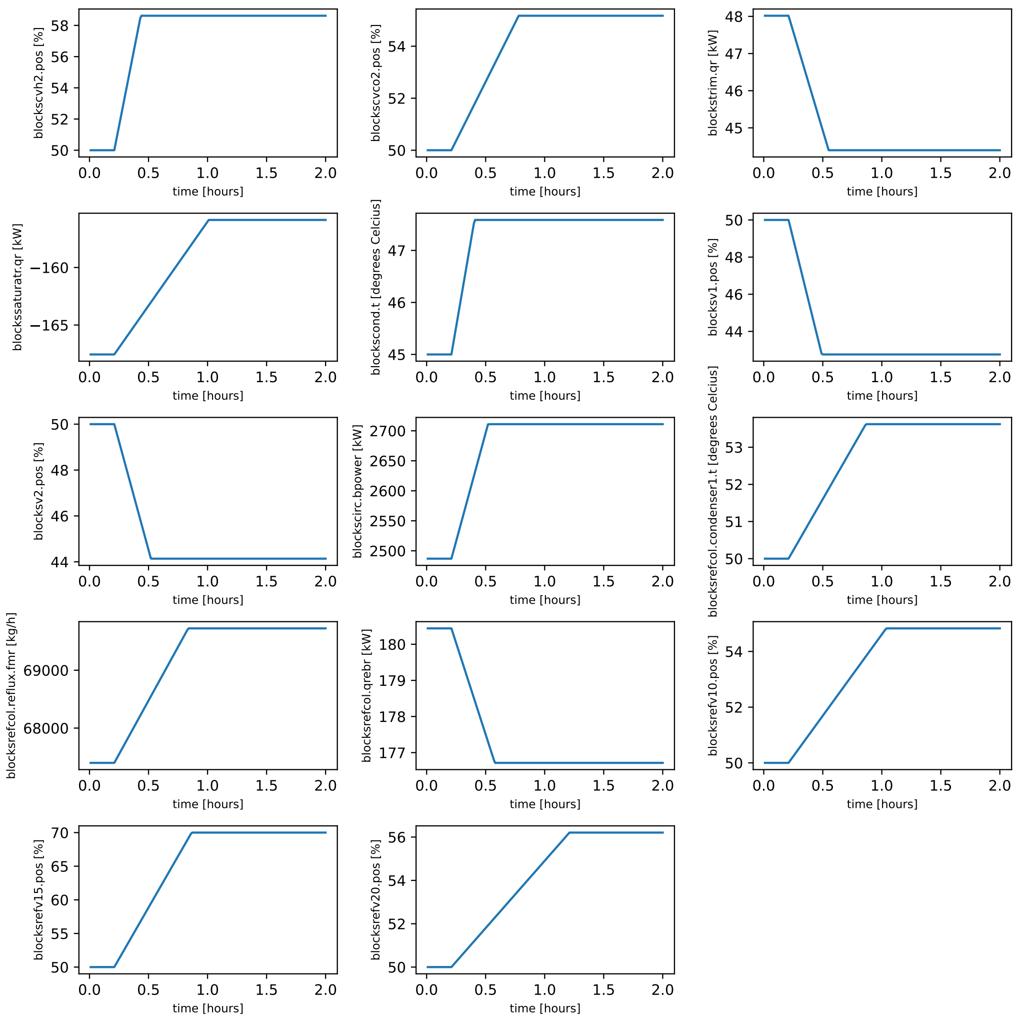


Figure 19: The time evolution of the manipulated variables from the experiment titled ”appeal-branch-cabinet-bargain”.

system control. It is plausible that the control variables might be better conceptualised as manipulated variables in an otherwise open-loop dynamical system. Given the presence of feedback mechanisms, it becomes conceivable that the state variables may influence the system, effectively acting as control inputs. This intricate interplay could pose a considerable challenge for DMDc to handle effectively.

On the other hand, as seen in Figure 12, the LSTM model produces good predictions, especially when trained on the clean dataset, showing it is capable of multi-input-multi-output predictions for this complex dynamic chemical process. Moreover, the model still achieves reasonable predictions even when trained on noisy data, as shown in Figure 13. This suggests the LSTM model can be applied for industrial chemical process modelling in real-life scenarios, and it has the potential to be further embedded for process control and optimisation.

The NARX models performed similarly perform well, and our results suggest they may have an advantage in terms of RMSE when the data included are noisy. These two modelling frameworks therefore provide promising directions towards modelling the process dynamics. Conformal prediction methods can be used to provide uncertainty estimates.

6.1 Recommendations and future work

In this study we investigate several dynamic models for representing chemical process dynamics. While we found that NARX models perform very accurately, the model compactness should also be considered with developing high-level control and optimisation applications. For example, a next step of this study could be to enable intelligent production scheduling given changes in the availability of upstream green hydrogen.

Some potential directions for future work based on the directions of this study are summarised below:

- **DMDc:** A promising future direction for addressing the challenge at hand with DMDc involves manually enforcing the data matrices to encompass all time-series samples, particularly focusing on horizontal stacking. In the context of per-sample analysis, exploring alternative versions of DMD, such as higher-order DMD, could be beneficial. Additionally, further investigation is warranted to better understand the organisation of state and control variables and explore potential inter-dependencies among these variables.
- **Uncertainty quantification:** In the realm of nonlinear dynamical systems, exploring the application of time-dependent polynomial

chaos expansion (PCE) can be valuable, as it is a widely used technique for functional approximation-based uncertainty quantification. Additionally, PCE can be employed for higher-level tasks, such as variance-based global sensitivity analysis.

- **LSTM:** In this work, a vanilla LSTM model is trained for all the tests. The model may be further improved by incorporating with proper hyperparameter optimisation, such as by using Bayesian optimisation.
- **Transformers:** This study focuses on evaluating the feasibility of a spacetimeformer model through preliminary tests. However, the model's performance was constrained by limited GPU resources. To enhance its capabilities, the immediate next phase includes the creation of a custom dataset aligned with the specific requirements of the challenge owner (JM) and fine-tuning of hyperparameters to adapt the open-source model. The anticipated outcomes include longer sliding window lengths, enhanced prediction horizons, and refined spatial relationships within more input variables rather than focusing on the controllable ones, aiming for a more robust model.
- **Integrating FC-ConvLSTM for multivariate time-series analysis:** In our exploratory data analysis (data processing), we identified significant correlations between time-series elements, as detailed in our correlation Table 1. To further exploit these correlations and address multi-collinearity, we propose the integration of a fully connected convolutional long short-term memory (FC-ConvLSTM) network for future research. The FC-ConvLSTM's ability to efficiently capture both spatial and temporal dependencies makes it an ideal choice for handling the complexities of multivariate time series data, especially in scenarios with high inter-series correlations.
- **SHAP analysis:** Understanding model decisions is vital in machine learning, particularly for complex, black-box models of multivariate time series data, such as LSTM neural networks. While these neural network models excel in processing sequential data, their intricate structures pose challenges for interpretability.

  SHapley Additive exPlanations (SHAP) offer a solution, grounded in cooperative game theory [8]. SHAP values, which distribute credit among features based on their contribution to a prediction, adhere to principles of fairness and consistency. However, applying SHAP to LSTMs is not straightforward due to the sequential nature of the data and the internal complexity of these models. As such, it has not been possible in the course of the data study to generate SHAP on the LSTM models as initially desired. Despite these challenges, finding an approach for conducting SHAP analysis on LSTMs is likely to be invaluable. It not only aids in understanding the model's decision-making process but also highlights potential areas for improvement, thereby enhancing the model's reliability and applicability.

7 Team members

Team members (alphabetical by last name):

Waqar Muhammad Ashraf is a PhD student at the Department of Chemical Engineering, University College London, UK. His research is centred around data-driven modelling and optimisation for the energy systems supporting the net-zero goal.

Maryam Khaksar Ghalati is a research associate at the School of Engineering, University of Leicester, focusing on data-driven and mathematical modelling for materials and manufacturing processes.

Matthew Jones is a PhD student in the Department of Physics and Astronomy at the University of Sheffield. His research is centred around modelling the phase separation of polymer blends using both theory-driven and data-driven techniques.

Sharana Kumar Shivanand is a postdoctoral research associate at The Alan Turing Institute. His research combines uncertainty quantification, machine learning, and computational physics to tackle challenges in computational material science and mechanics.

Maria Luisa Taccari is currently pursuing her PhD at the University of Leeds, focusing on the development of deep learning surrogate models in groundwater flow simulation. Maria was the technical facilitator for this challenge.

Kelly Tallau is an Industrial Data Scientist with a focus on integrating analytics for advancing R&D, asset performance, manufacturing and process improvements.

Calvin Tsay is a Lecturer in Computing at Imperial College London. Calvin was the Principal Investigator (PI) for this project.

Haiting Wang is a PhD student from Imperial College London. She is working on hybrid modelling and data-driven modelling for bioprocesses.

Skyler Xie is a doctoral researcher at the Alan Turing Institute. His research interest lies in quantitative economics, which combines economic theory with econometrics and machine learning to model panel data with high-dimension and high-frequency dynamics.

Runyi Yang is a Master's student at Imperial College London. His research focuses on 3D computer vision, specialising in 3D reconstruction and neural radiance fields. He was the social facilitator for this challenge.

Kexin Yin is a first year PhD student at the University of Huddersfield. His research area includes in-line product quality inspection and parameter optimisation for medical 3D printing. He was in charge of the test of Transformer-related models in this challenge.

Challenge owners (alphabetical by last name):

Joe Emerson is a Senior Digital Chemical Engineer in Johnson Matthey's research division. His research interests are in Machine Learning, Data Analytics and Hybrid Modelling for Chemical/Biochemical Systems with applications to Process Monitoring, Control and Optimisation.

Liam Fleming is a process and optimisation engineer at Johnson Matthey. He focuses on developing opportunities to augment existing processes within the business by utilising data-centric approaches such as probabilistic modelling and evolutionary algorithms.

Robert Gallen is a principal engineer at Johnson Matthey. His focus within the JM technology centre extends to developing new technologies and capabilities across the range of JM's offerings.

Darren Gobby is a Chartered Chemical/Process Engineer with 20 years of process design and development experience within industry. His focus is leading and developing new technologies in the biochemical, manufacturing and petrochemical areas.

References

[1] Rina Foygel Barber et al. *Conformal prediction beyond exchangeability*. 2023. arXiv: 2202 . 13415 [stat.ME]. URL: https://arxiv.org/abs/2202.13415.

[2] B Benson et al. “Forecasting solar cycle 25 using deep neural networks”. In: *Solar Physics* 295.5 (2020), p. 65.

[3] Jake Grigsby, Zhe Wang, and Yanjun Qi. *Long-Range Transformers for Dynamic Spatiotemporal Forecasting*. 2021. arXiv: 2109.12218 [cs.LG].

[4] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[5] V Roshan Joseph, Evren Gul, and Shan Ba. “Maximum projection designs for computer experiments”. In: *Biometrika* 102.2 (2015), pp. 371–380.

[6] J. Nathan Kutz et al. *Dynamic Mode Decomposition*. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics, Nov. 2016. ISBN: 978-1-61197-449-2. DOI: 10.1137/1.9781611974508. (Visited on 12/07/2023).

[7] Jundong Li et al. “Feature selection: A data perspective”. In: *ACM Computing Surveys (CSUR)* 50.6 (2017), pp. 1–45.

[8] Scott M Lundberg and Su-In Lee. “A Unified Approach to Interpreting Model Predictions”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf.

[9] Christopher Olah. *Understanding LSTM Networks*. https://colah.github.io/posts/2015-08-Understanding-LSTMs/. Accessed: [Insert today’s date]. 2015.

[10] Joshua L. Proctor, Steven L. Brunton, and J. Nathan Kutz. “Dynamic Mode Decomposition with Control”. In: *SIAM Journal on Applied Dynamical Systems* 15.1 (Jan. 2016). Publisher: Society for Industrial and Applied Mathematics, pp. 142–161. DOI: 10.1137/15M1013857. URL:

https://epubs.siam.org/doi/10.1137/15M1013857 (visited on 12/07/2023).

[11] Tomas Eloy Salais-Fierro et al. “Demand prediction using a soft-computing approach: a case study of automotive industry”. In: *Applied Sciences* 10.3 (2020), p. 829.

[12] Peter J. Schmid. “Application of the dynamic mode decomposition to experimental data”. en. In: *Experiments in Fluids* 50.4 (Apr. 2011), pp. 1123–1130. ISSN: 1432-1114. DOI: 10.1007/s00348-010-0911-3. (Visited on 12/07/2023).

[13] Peter J. Schmid. “Dynamic Mode Decomposition and Its Variants”. In: *Annual Review of Fluid Mechanics* 54.1 (2022), pp. 225–254. DOI: 10.1146/annurev-fluid-030121-015835. URL: https://doi.org/10.1146/annurev-fluid-030121-015835 (visited on 12/07/2023).

[14] Peter J. Schmid. “Dynamic mode decomposition of numerical and experimental data”. en. In: *Journal of Fluid Mechanics* 656 (Aug. 2010). Publisher: Cambridge University Press, pp. 5–28. ISSN: 1469-7645, 0022-1120. DOI: 10.1017/S0022112010001217. (Visited on 12/07/2023).

[15] Jodie M Simkoff et al. “Process control and energy efficiency”. In: *Annual Review of Chemical and Biomolecular Engineering* 11 (2020), pp. 423–445.

[16] Yi Tay et al. “Efficient transformers: A survey”. In: *ACM Computing Surveys* 55.6 (2022), pp. 1–28.

[17] Calvin Tsay and Michael Baldea. “Integrating production scheduling and process control using latent variable dynamic models”. In: *Control Engineering Practice* 94 (2020), p. 104201.

[18] Laurens Van Der Maaten, Eric O Postma, H Jaap van den Herik, et al. “Dimensionality reduction: A comparative review”. In: *Journal of Machine Learning Research* 10.66-71 (2009), p. 13.

[19] Ashish Vaswani et al. “Attention Is All You Need”. In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[20] Xueqi Zhang, Meng Zhao, and Rencai Dong. “Time-series prediction of environmental noise for urban IoT based on long short-term memory recurrent neural network”. In: *Applied Sciences* 10.3 (2020), p. 1144.

Appendix

Summary of LSTM and NARX results

Table 2: RMSE Results for the LSTM Model

Output Variable	Step 1	Step 2	Step 3	Step 4	Step 5
1	9.87E-03	9.96E-03	1.18E-02	1.36E-02	1.53E-02
2	7.67E-05	7.99E-05	9.34E-05	1.12E-04	1.33E-04
3	3.67E-03	3.07E-03	3.55E-03	4.27E-03	5.15E-03
4	5.17E-03	1.77E-03	2.75E-03	4.89E-03	7.27E-03
5	3.38E-02	1.55E-02	1.76E-02	2.12E-02	2.59E-02
6	1.98E-02	2.22E-02	2.33E-02	2.50E-02	2.74E-02
7	1.83E-02	1.42E-02	1.56E-02	1.82E-02	2.11E-02
8	5.31E-03	1.41E-03	2.44E-03	4.38E-03	6.49E-03

Table 3: RMSE Results for the NARX Model

Output Variable	Step 1	Step 2	Step 3	Step 4	Step 5
1	1.41E-02	1.49E-02	1.88E-02	2.14E-02	2.45E-02
2	1.61E-04	1.91E-04	2.52E-04	2.19E-04	2.35E-04
3	6.25E-03	8.63E-03	5.55E-03	9.06E-03	1.07E-02
4	5.32E-03	3.93E-03	3.56E-03	4.43E-03	4.78E-03
5	6.57E-02	3.63E-02	4.16E-02	6.55E-02	9.11E-02
6	4.28E-02	3.25E-02	3.53E-02	3.80E-02	4.51E-02
7	2.83E-02	2.66E-02	3.03E-02	4.70E-02	5.55E-02
8	4.56E-03	3.18E-03	2.70E-03	3.36E-03	3.45E-03

Correlation analysis

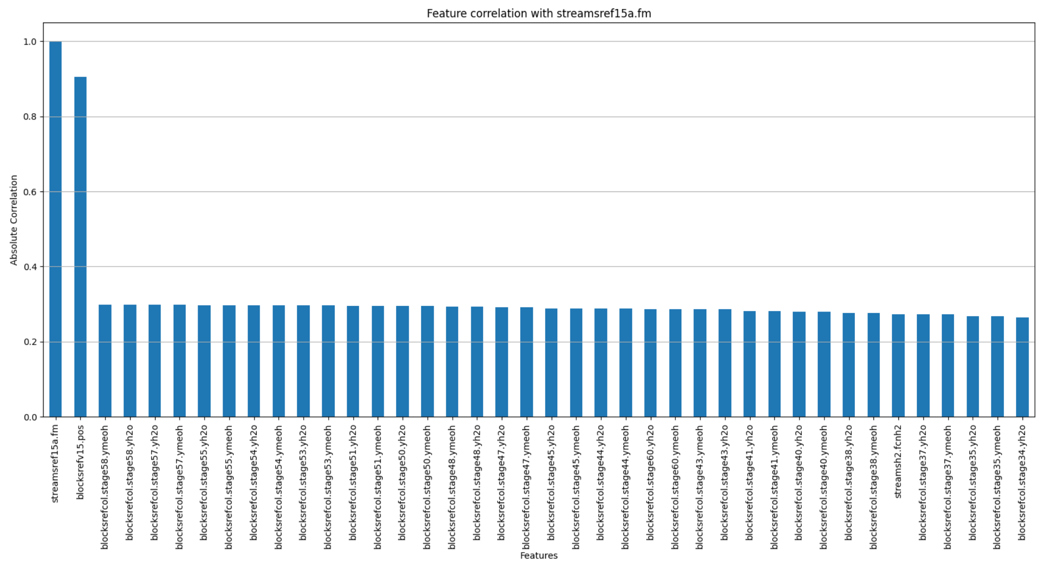


Figure 20: Absolute correlation of top 40 variables with streams.ref15a.fm.

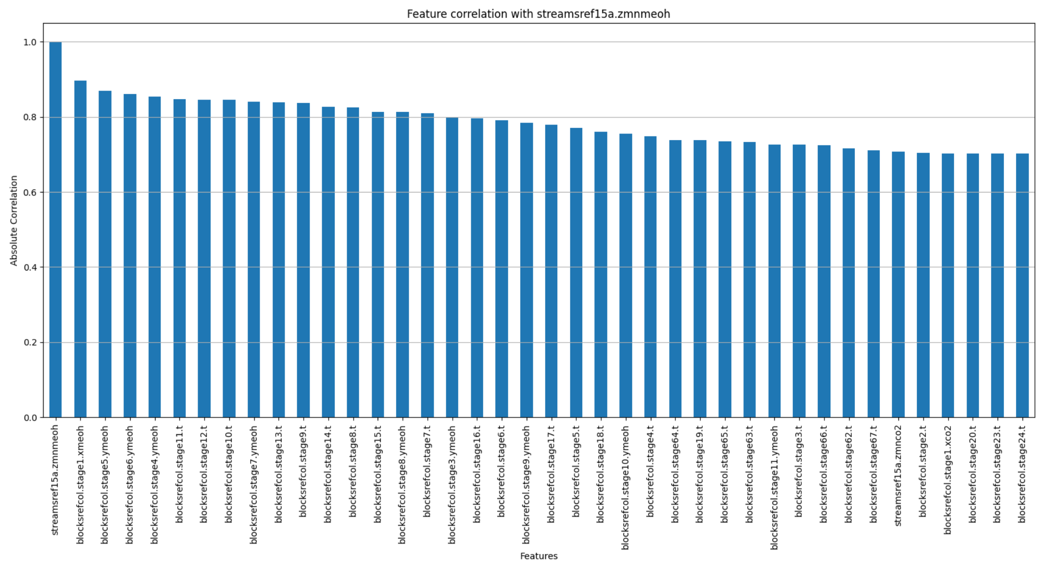


Figure 21: Absolute correlation of top 40 variables with streamsref15a.zmnmeoh.

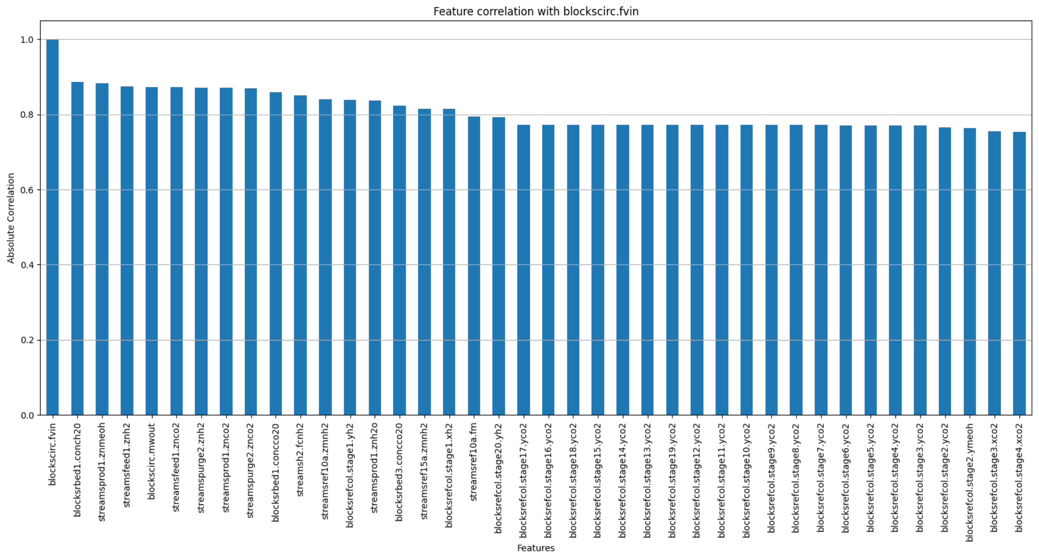


Figure 22: Absolute correlation of top 40 variables with blockscirc.fvin.

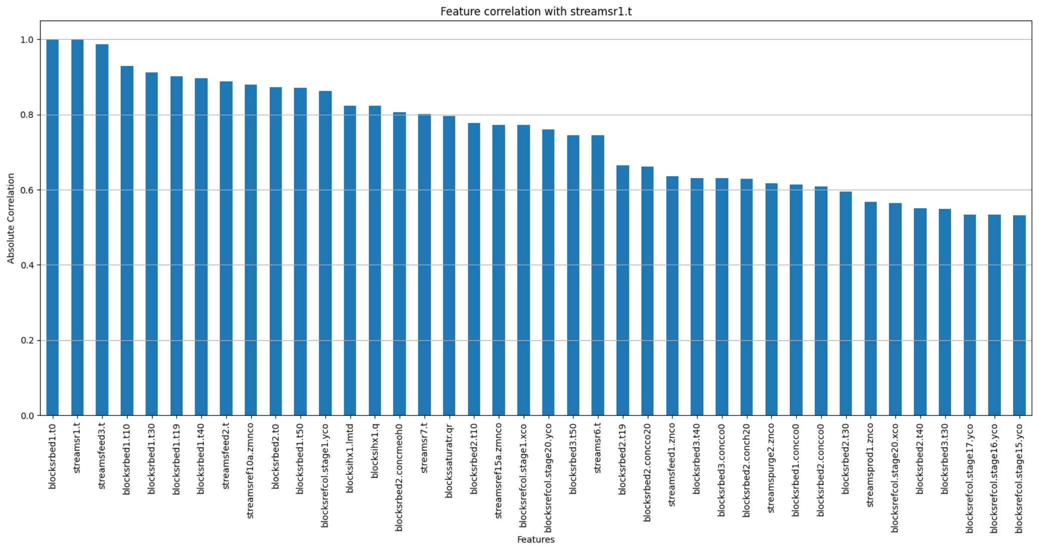


Figure 23: Absolute correlation of top 40 variables with streamsr1.t.

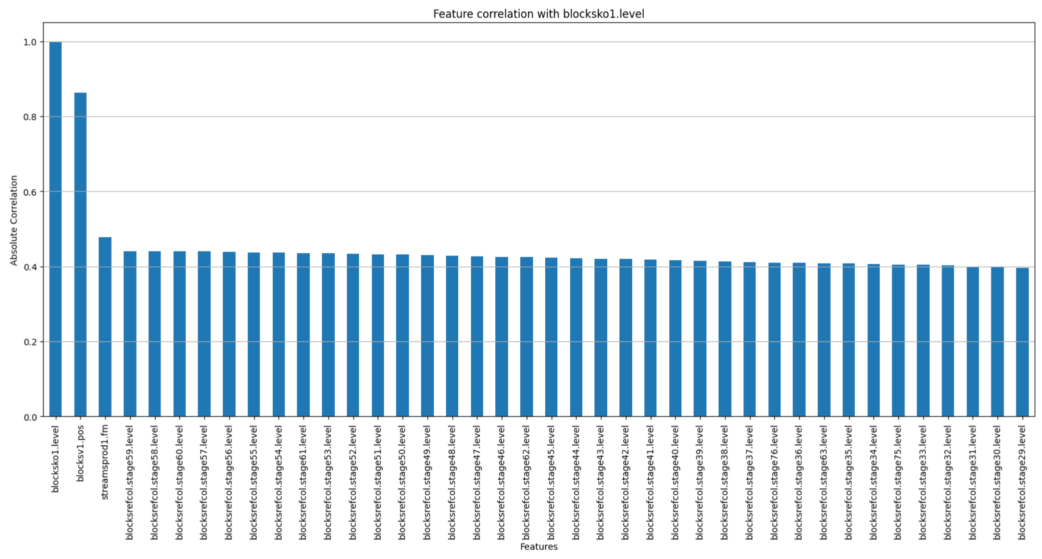


Figure 24: Absolute correlation of top 40 variables with blocksko1.level.

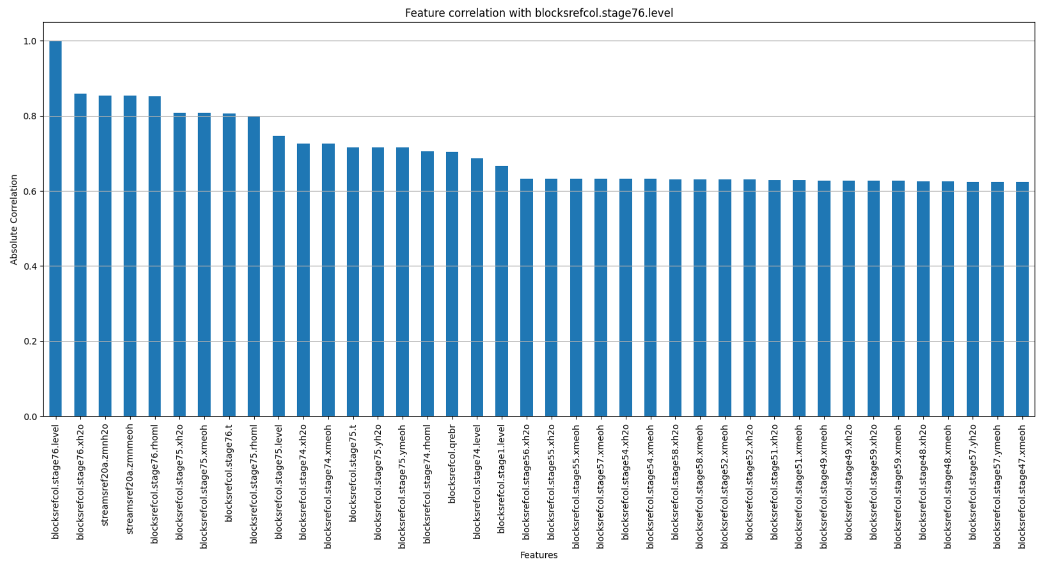


Figure 25: Absolute correlation of top 40 variables with blocksrefcol.stage76.level.

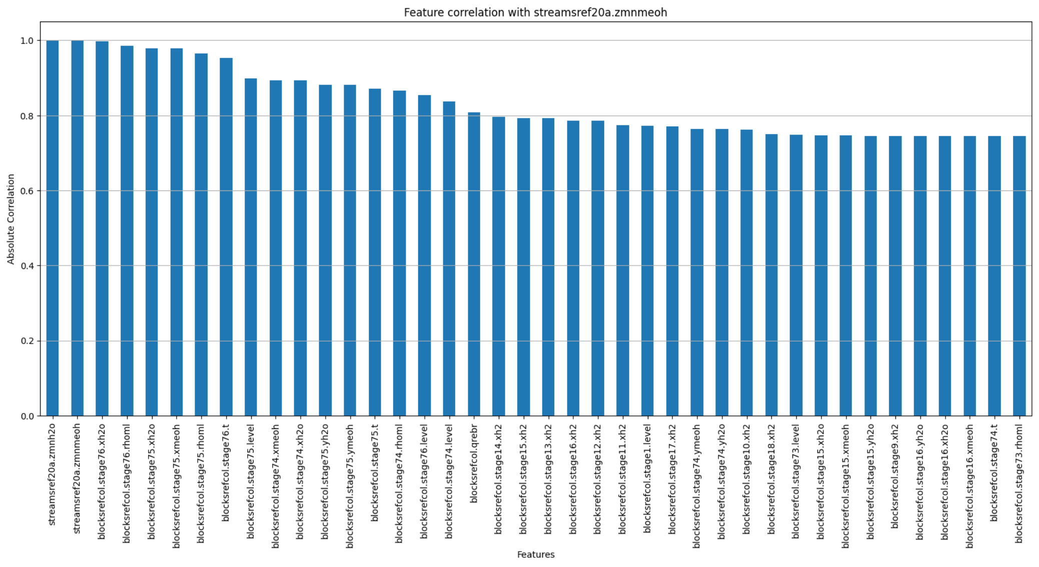


Figure 26: Absolute correlation of top 40 variables with streamsref20a.zmnmeoh.

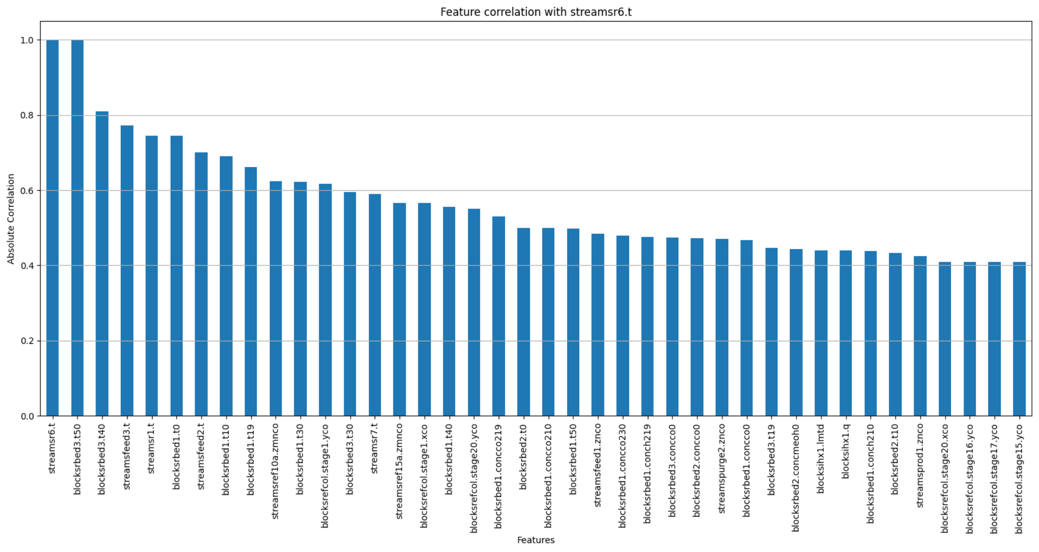


Figure 27: Absolute correlation of top 40 variables with streamsr6.t.

The Alan Turing Institute

turing.ac.uk
@turinginst