Highlights

Experimental Validation of Deep Koopman MPC for Real-Time Pasteurization Unit Control

Patrik Valábek, Michaela Horváthová, Gabriele Pannocchia, Martin Klaučo

- First experimentally validated Koopman MPC in chemical process control
- Novel deep Koopman MPC with lifted state correction for energy-intensive processes
- 30% control performance improvement over conventional subspace identification method
- Real-time implementation demonstrating industrial deployment feasibility
- Demonstrated energy efficiency and waste reduction in pasteurization control

Experimental Validation of Deep Koopman MPC for Real-Time Pasteurization Unit Control

Patrik Valábek[a], Michaela Horváthová[a], Gabriele Pannocchia[b], Martin Klaučo[a,c]

[a] *Slovak University of Technology in Bratislava, Institute of Information Engineering, Automation, and Mathematics, Radlinského 9, Bratislava, Slovakia*
[b] *University of Pisa, Department of Civil and Industrial Engineering, Largo Lucio Lazzarino, Pisa, Italy*
[c] *Czech Technical University, Department of Control Engineering, Karlovo náměstí 13, Prague, Czechia*

Abstract

This paper presents a novel deep Koopman Model Predictive Control framework for energy-intensive processes, addressing the critical challenge of real-time nonlinear control in pasteurization systems. The proposed approach employs neural networks to learn lifting functions that transform nonlinear system dynamics into a linear representation in the lifted space, enabling computationally efficient MPC implementation. A key innovation is the introduction of a lifted state correction mechanism that compensates for the mismatch between unmeasurable lifted states and real measurements, significantly improving control accuracy and practical implementability. Experimental validation on a laboratory-scale pasteurization unit demonstrates that the proposed deep Koopman MPC framework achieves 30% improvement in control performance compared to conventional subspace identification methods, while maintaining real-time execution within 10ms on standard hardware. The enhanced control accuracy translates to reduced product waste and improved energy efficiency in pasteurization operations without additional computational overhead. This work represents one of the first experimental implementations of Koopman MPC in chemical process control and energy-intensive applications, establishing its practical feasibility for industrial deployment.

Keywords:
Deep Koopman Operator, Model Predictive Control, Energy-Intensive Processes, Pasteurization Unit, Lifted State Correction, Nonlinear Control, Real-time Implementation, Chemical Process Control

1. Introduction

Pasteurization is an important part of modern food and beverage production, which ensures microbial safety and extends shelf life without compromising product quality. Beyond the food industry, pasteurization is also relevant in sectors like pharmaceutics and biotechnology, where microbial control is essential for quality and safety, and thermal processing must be tightly regulated. On one hand, insufficient thermal treatment fails to eliminate harmful microorganisms and thus poses significant health risks. On the other hand, excessive thermal exposure can deteriorate sensory attributes, reduce nutritional content, and lead to unnecessary energy consumption. Therefore, accurate identification and effective control of pasteurization processes are important (Martin et al., 2018).

To address these challenges, we propose a Koopman-based MPC framework for a laboratory-scale pasteurization unit characterized by nonlinear heat transfer dynamics, time delays, and sensor noise. Its linear structure in the lifted space enables real-time optimization using standard linear MPC solvers, combining computational efficiency with the ability to handle complex, nonlinear dynamics. This

is essential for pasteurization, where thermal processes must be controlled accurately and must achieve the desired temperature of the product quickly, as even minor deviations in temperature can compromise safety or degrade product quality, leading to increased waste.

In the past, the heuristic fuzzy controllers were introduced in Shieh et al. (1992) to improve the control performance over the standard PID controllers. The simple design of fuzzy controllers allows for easy implementation and tuning. However, heuristic fuzzy controllers use a model that is based on expert-defined rules that lack any process modeling needed for consistent and safe thermal control. Later, a cascade control structure, as discussed in Morison (2005). Despite its improved disturbance rejection, cascade control remains limited by its reliance on heuristic tuning and lack of explicit process modeling, making it suboptimal for systems with strong nonlinearities and time delays.

With the increase in computational power of available industrial hardware, the Model predictive control (MPC), which computes the optimal control input based on a prediction of a model, took the place of state-of-the-art controller for PU (Ibarrola et al., 2002). Hadisupadmo et al. (2016) highlight the superior performance of MPC over the classical cascade control using the same linear model for the tuning of PID controllers and in MPC. These studies underscore the shift towards data-driven modeling approaches in pasteurization processes, moving beyond traditional linear models to capture the system's nonlinearities.

Further development led to the use of nonlinear models for the MPC design, as shown in Thostrup et al. (2022). This further improved the control performance due to the nonlinear nature of a PU. However, nonlinear models are often more challenging to obtain than linear or data-based models and require first principle knowledge of a specific unit. The work of Riverol et al. (2008) uses a neural network to approximate the nonlinear model of a pasteurization unit, but this model is used only in fuzzy controllers. Even with the neural network as a data-based nonlinear model, the nonlinear control is often computationally expensive. It requires more time to solve the optimization problem, which is a limitation in real-time applications and can resolve suboptimal or no solutions in the available time. Subspace identification methods have also been applied to PUs to compute linear state-space models directly from data Dzurková et al. (2024), enabling efficient MPC design. Their key limitation is the assumption of linearity, which can lead to poor performance in systems with significant nonlinear dynamics, like the heat exchange in pasteurization.

Koopman-based models offer a state-of-the-art alternative for the modeling of the PU as they transform nonlinear system dynamics into a lifted, multi-dimensional space, which can be approximated linearly. The Koopman operator theory was initially introduced by Koopman (1931), its use in control has only gained popularity in recent decades, particularly following the work of Mezić (2005). Methods like Dynamic Mode Decomposition (DMD) and its extension (eDMD) (Schmid, 2010; Williams et al., 2015) provide practical, data-driven approaches to approximating the Koopman operator. More recently, deep learning approaches have been proposed to automate the design of lifting functions, as seen in the deep Koopman operator (Lusch et al., 2018) and deep eDMD (Li et al., 2017), improving the modeling of complex, nonlinear dynamics without requiring expert knowledge. The choice of lifting functions can be difficult and time-consuming; a poor choice of lifted function can lead to poor performance Valábek et al. (2025).

The use of the Koopman operator in MPC was introduced by Korda and Mezić (2018). They introduced the necessary transformations of nonlinear MPC to linear MPC using the eDMD approximation of the Koopman operator. Since this transformation was established, several papers present a combination of MPC with the deep Koopman approach, mainly in the fields of soft robotics (Bruder et al., 2020), autonomous driving (Švec et al., 2025) and power systems (Zhao et al., 2023). As of our knowledge, a practical

chemical engineering or energy-intensive application of deep Koopman MPC has not yet been published. However, an experimental validation of the Koopman-based model for a thermal process – a fuel cell stack was presented in Huo et al. (2025). The study employed a Linear Quadratic Regulator (LQR), which is optimal and computationally extremely cheap but lacks systematic input and output constraint handling. In contrast, our work addresses a thermal control problem in the context of pasteurization, an energy-intensive and safety-critical process, using a Koopman MPC framework, not LQR.

The practical application of Koopman models in control remains limited by the challenge of reconstructing the lifted state from real-time measurements. The controlled system is Markovian (the next state depends only on a previous state and the control input) in a lifted state space. However, it is non-Markovian with respect to measurement space. Due to model inaccuracies and measurement noise, we are not able to reconstruct the lifted space only from the measurements at each time step, as is often proposed, leading to degraded control performance. This issue is more significant in deep Koopman models, where inaccuracies in the learned lifting function can introduce steady-state offsets and reduce energy efficiency. Recent work has explored embedding historical information (Han et al., 2023) and improving lifting functions to mitigate these effects, but a fully robust solution remains an open challenge. In contrast, Jin et al. (2024) proposes a method where the lifting function properties are improved to achieve a minor mismatch between lifted states and the ones that are received by transforming the measurements of a system. This leads to a better approximation but does not solve the problem completely.

In this work, we compensate for the gap between lifted measurements and real states. Rather than relying purely on direct lifting of measurements, our method considers model predictions with real-time measurements to correct the lifted state estimates dynamically. This correction mechanism helps compensate for noise, reduce the impact of lifting errors, and improve the overall control performance of the Koopman MPC. Without this correction, the Koopman MPC may not be able to achieve the desired performance in real-world control scenarios.

The contribution of this paper can be summarised as follows: (i) Integration of the Koopman MPC framework with state observers to match lifted states with measured physical states, (ii) Experimental validation of the proposed framework using a pasteurisation unit, a multivariable system with non-linear dynamics, demonstrating its effectiveness in a real-world application. (iii) Reduction of economic costs in an energy-intensive process through a model-based optimal control strategy. To the best of the author's knowledge, this is the first real-time application of the Koopman MPC framework in the field of chemical process control and one of the first experimental applications of Koopman MPC in general.

The paper begins with Section 2, introducing the Koopman theory, MPC framework, and subspace identification. Section 3 presents the proposed deep Koopman MPC framework, detailing the data-driven model identification, lifted state correction, and implementation workflow. Section 4 describes the experimental setup, with Section 5 covering both the subspace and deep Koopman models of the plant. Section 6 outlines the control implementation, and Section 7 presents and discusses the experimental results. The paper concludes in Section 8 with a summary of key contributions.

2. Preliminaries

In this section, we present the preliminary concepts used throughout this paper. We provide an overview of Koopman theory and the Koopman operator. Additionally, we discuss the subspace identification method, which serves as a benchmark in the case study. Finally, we present the Model Predictive Control (MPC) framework used to evaluate the performance of both Koopman-based and subspace models in closed-loop in multiple control setups.

2.1. Koopman Theory and Koopman Operator

The Koopman theory was first introduced by Koopman (1931) and gained significant attention only in the last two decades, starting with the work of Mezić (2005). The theory is based on the idea that the dynamics of a nonlinear system can be represented in a higher-dimensional space, where the system behaves linearly. This is achieved by lifting the original state space into a multi-dimensional space, where the dynamics can be described by a linear operator known as the Koopman operator. Considering the nonlinear discrete dynamical system:

$$x_{k+1} = f(x_k, u_k), \tag{1}$$

where $x_k \in \mathbb{R}^{n_\mathrm{x}}$ is the state of the system at time step k, $u_k \in \mathbb{R}^{n_\mathrm{u}}$ are the control inputs, and $f(\cdot,\cdot) : \mathbb{R}^{n_\mathrm{x}} \times \mathbb{R}^{n_\mathrm{u}} \to \mathbb{R}$ is a nonlinear function that maps the current state to the next state. Symbol $\mathbb{R}$ denotes the set of real numbers. The Koopman operator $\mathcal{K}$ acts on the space of observables $g(\cdot) : \mathbb{R}^{n_\mathrm{x}} \to \mathbb{R}$, and is defined as:

$$\mathcal{K}g(\cdot) = g(\cdot) \circ f(\cdot,\cdot), \tag{2}$$

where the Koopman operator $\mathcal{K}$ is a linear operator that evolves lifted states of the system forward in time. The symbol $\circ$ denotes the composition of functions. The Koopman operator sets up a discrete-time dynamical system on the observable function $g(\cdot)$:

$$\mathcal{K}g(x_k, u_k) = g(f(x_k, u_k)) = g(x_{k+1}). \tag{3}$$

We can also obtain, from a Koopman operator eigenfunctions $\varphi(x)$ corresponding to eigenvalues λ:

$$\varphi(x_{k+1}) = \mathcal{K}\varphi(x) = \lambda\varphi(x). \tag{4}$$

Obtaining Koopman eigenfunctions from data or analytic representations of a system is the main challenge. Computation of these eigenfunctions enables linear representations of strongly nonlinear systems. In this work, the control inputs u_k are not lifted but used directly in the linear system. This simplifies the model structure and makes the design of the controller more straightforward.

Because the linear operator $\mathcal{K}$ is infinite-dimensional, a data-driven approximation of this parameter is necessary. The parameter is approximated via the matrices of an LTI system $A_\mathcal{K}$, $B_\mathcal{K}$ and $C_\mathcal{K}$ that linearly approximate the behaviour of the system. In the Koopman-based identification of models, we also obtain the approximation of the Koopman operator and the lifting function $g(\cdot)$. There are currently several well-performing methods for the computation of the Koopman operator, based on the DMD (Schmid, 2010) and its extensions (Williams et al., 2015; Proctor et al., 2016; Brunton et al., 2016). In this work, we propose a variation of the state-of-the-art method called Deep Koopman operator (Lusch et al., 2018). In this work, we consider the finite-dimension truncation in the form of a linear time-invariant (LTI) predictor defined as:

$$z_{k+1} = A_\mathcal{K} z_k + B_\mathcal{K} u_k, \tag{5a}$$

$$x_k = C_\mathcal{K} z_k, \tag{5b}$$

$$z_0 = g(x_0), \tag{5c}$$

where $z_k \in \mathbb{R}^{n_\mathrm{z} \times 1}$, $A_\mathcal{K} \in \mathbb{R}^{n_\mathrm{z} \times n_\mathrm{z}}$, $B_\mathcal{K} \in \mathbb{R}^{n_\mathrm{z} \times n_\mathrm{u}}$ and $C_\mathcal{K} \in \mathbb{R}^{n_\mathrm{x} \times n_\mathrm{z}}$. Where the $g(\cdot)$ is the lifting function that maps the states x of the system to the lifted states z. The u_k is the control. The matrices in (5a) are referred to as Koopman matrices.

2.2. Subspace Identification

In this paper, the subspace identification serves as a benchmark for comparison of models identified using the Koopman Operator. The Koopman-based identification method provides a dynamic extension to linear models, and comparing its performance to subspace identification helps evaluate its effectiveness in capturing system dynamics, particularly in nonlinear systems. The Subspace Identification method—particularly the N4SID (Numerical algorithms for Subspace State Space System IDentification) method offers a robust and well-established linear alternative for system modeling. It is well-established and reliable linear system identification method (Katayama, 2005). The Subspace Identification method is also implemented in MATLAB's System Identification Toolbox.

2.3. Model Predictive Control

To analyse the performance of both the Koopman and Subspace models, we implement them within the MPC framework in various closed-loop scenarios. MPC explicitly relies on the accuracy of the prediction model to compute optimal control inputs, making it a natural choice for analysing the performance of these models. By comparing the closed-loop performance, we showcase how well each model captures the system dynamics. MPC also respects constraints and provides optimal control actions in real time.

$$\min_{u_0,\ldots,u_{N-1}} \quad \sum_{k=0}^{N-1} y_k^\intercal Q_\mathrm{y} y_k + \sum_{k=0}^{N-1} u_k^\intercal Q_\mathrm{u} u_k, \tag{6a}$$

$$\text{s.t.} \quad x_{k+1} = Ax_k + Bu_k, \quad k \in \mathbb{N}_0^{N-1}, \tag{6b}$$

$$y_k = Cx_k, \quad k \in \mathbb{N}_0^{N-1}, \tag{6c}$$

$$u_{\min} \leq u_k \leq u_{\max}, \quad k \in \mathbb{N}_0^{N-1}, \tag{6d}$$

$$x_0 = \hat{x}_{k|k}, \tag{6e}$$

where $x_k \in \mathbb{R}^{n_\mathrm{x}\times 1}$ represents the state vector at time step k. The vector of control inputs at each time step, denoted as $u_k \in \mathbb{R}^{n_\mathrm{u}\times 1}$, is the manipulated variable. The controlled output at time step k is denoted as $y_k \in \mathbb{R}^{n_\mathrm{y}\times 1}$. The parameters $n_\mathrm{x}, n_\mathrm{u}, n_\mathrm{y}$ represent the number of states, manipulated variables and controlled variables, respectively. Symbol $\mathbb{N}_0$ the set of non-negative integers

The matrices $A \in \mathbb{R}^{n_\mathrm{x}\times n_\mathrm{x}}, B \in \mathbb{R}^{n_\mathrm{x}\times n_\mathrm{u}}$ and $C \in \mathbb{R}^{n_\mathrm{y}\times n_\mathrm{x}}$ are the state, input and output matrices, respectively.

The prediction horizon is defined by N, which specifies how many time steps the MPC controller will predict based on the model. The matrices Q_y and Q_u represent the weighting factors for the output and control input in the objective function, respectively. The constraints on the control inputs are given by $u_{\min}$ and $u_{\max}$. Finally, x_0 denotes the initial state of the system, providing information about the estimation for the optimization problem at the current time step.

The control problem in (6) is a quadratic optimization problem with linear constraints, therefore it can be solved to global optimality with numerous solvers. To obtain the numerical solution we first reformulate the problem in dense-representation of the MPC. This is achieved by eliminating the state variables from the list of optimization variables and expressing them as a function of the initial condition and future control inputs. The dynamics are predicted in a compact matrix representation of the states based just on the control trajectory. This significantly reduces the number of optimization variables and constraints, making the problem more computationally efficient and better suited for real-time applications.

3. Deep Koopman Model Predictive Control Framework

This section proposes a framework for data-driven identification and control using Koopman operator theory. It outlines the proposed model identification process, proposed lifted states correction, and the formulation of the MPC controller, including a complete step-by-step workflow suitable for real-time deployment.

3.1. Data-Driven Identification for Deep Koopman Models

We leverage the deep Koopman-based identification, which is a data-driven method that uses neural networks to approximate the lifting function and the Koopman operator. Note that the proposed method is generalizable to any Koopman-based data-driven identification method that results in an LTI system. The main advantage of deep Koopman models lies in their ability to use NN to learn complex, nonlinear dynamics while still representing the dynamics as a linear model using the Koopman operator. This modeling paradigm was first introduced by Lusch et al. (2018) and has since been extended to incorporate control inputs in several follow-up studies (Shi and Meng, 2022; Han et al., 2020).

Unlike extended Dynamic Mode Decomposition (eDMD), which requires the explicit selection of lifting functions, deep Koopman models operate under the premise that such lifting functions are not known a priori. In eDMD, an inappropriate choice of lifting functions can result

in a poor approximation of the Koopman operator (Valábek et al., 2025). Moreover, eDMD suffers from the curse of dimensionality, i.e. the exponential increase in computational cost as the dimension of the observation space grows. Consequently, the Koopman matrix in eDMD is often truncated to a lower-dimensional representation, possibly introducing further approximation errors.

In contrast, deep Koopman models learn optimal lifting functions for a predefined observable space dimension via neural network training. These lifting functions are typically obtained using an autoencoder-like architecture. However, for control applications, the projection function is often approximated with a matrix to facilitate integration into model-based control schemes. An illustrative architecture of the deep Koopman model used for system identification with control inputs is shown in Figure 1.

As a result of the identification process, we obtain the following linear representation of the deep Koopman model:

$$z_{k+1} = A_{\mathcal{K}} z_k + B_{\mathcal{K}} u_k, \tag{7a}$$

$$y_k = C_{\mathcal{K}} z_k, \tag{7b}$$

where $z_k \in \mathbb{R}^{n_z \times 1}$ is the lifted state, $u_k \in \mathbb{R}^{n_u \times 1}$ is the control input, and $y_k \in \mathbb{R}^{n_y \times 1}$ is the system output. The matrices $A_{\mathcal{K}} \in \mathbb{R}^{n_z \times n_z}$, $B_{\mathcal{K}} \in \mathbb{R}^{n_z \times n_u}$, and $C_{\mathcal{K}} \in \mathbb{R}^{n_y \times n_z}$ are the Koopman system matrices identified from the data. The parameters n_z, n_u, n_y represent the number of lifted states, manipulated variables and controlled variables, respectively.

This representation is beneficial within an MPC framework, where the obtained lifted linear dynamics of the Koopman-based model preserve the convexity of the optimization problem while precisely capturing the complex and nonlinear behaviour of the original system.

3.2. Correction of the Lifted States

Another key novelty of the proposed approach is the direct estimation of the unmeasurable lifted states in the deep Koopman model. While other methods often rely on alternative approximations, see Section 1, we apply the Kalman filter to improve the accuracy of the approximation of the lifted states z_k. The lifted states z_k obtained from the linear deep Koopman model in (7) are not directly measurable. Furthermore, the values of z_k we obtain solely by lifting the measurements, e.g. $z_k = g(y_k)$, are in a noisy scenario and, without incorporating the history of a system, strongly inaccurate. For the proper functioning of the MPC controller, it is crucial to have accurate estimates of the lifted states. Therefore, we propose to use Kalman filter estimation to estimate the unmeasurable lifted states z_k. The Kalman filter is an optimal recursive estimator that uses measurements of the system output to continuously update the state estimate. The filter works in a forward mode, meaning that it predicts the next state based on the current state and then corrects it with new measurement data.

The choice of the corrector of lifted states is motivated by two practical factors. First, in our experimental validation on a pasteurization unit, the system is often operated from a wide range of initial conditions, many of which are far from the steady-state. This leads to significant nonlinear behavior during the initial phase, where static estimators may become inadequate. Second, the sampling time of the pasteurization process is sufficiently long, making it feasible to run the KF in real time without introducing computational delays. As a result, there is no need to simplify the estimator or resort to static alternatives.

At the start of the process, the initial state $\hat{z}_0$ is typically obtained from the first measurement y_0, and the initial error covariance P_0 is set to reflect the uncertainty of the initial estimate. A common approach is to assume that the initial state is somewhat uncertain, so a large initial error covariance matrix is used.

Kalman filter provides an optimal estimate of the Koopman model, even in the presence of its unmeasurable states, uncertainties, and noise in the system itself.

3.3. Deep Koopman Model Predictive Control

In this section, we present the details of the implementation of the MPC framework utilizing

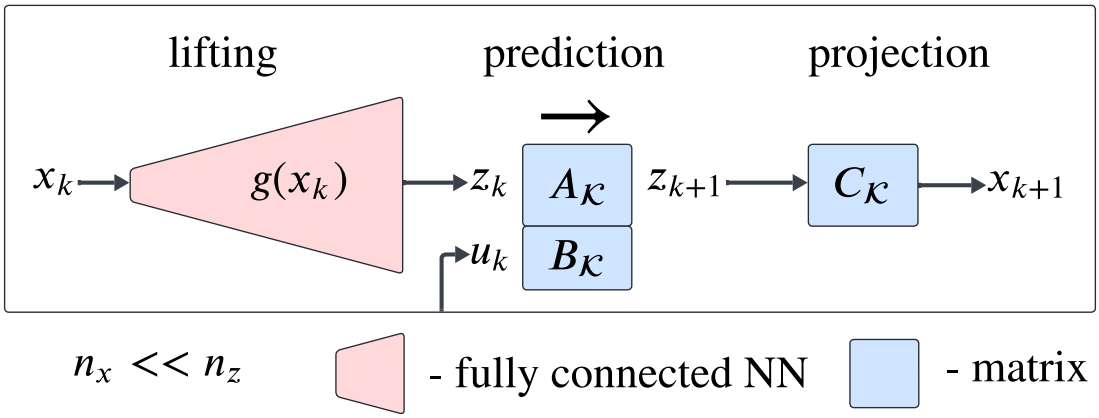


Figure 1: Architecture for the identification of a Deep Koopman model with control. Shapes with blue backgrounds represent matrices, while shapes with pink backgrounds correspond to fully connected neural networks.

the deep Koopman model. The controller is constructed as follows:

$$\min_{u_0,\dots,u_{N-1}} \quad \sum_{k=0}^{N-1} y_k^\intercal Q_\mathrm{y} y_k + \sum_{k=0}^{N-1} u_k^\intercal Q_\mathrm{u} u_k, \tag{8a}$$

$$\text{s.t.} \quad z_{k+1} = A_\mathcal{K} z_k + B_\mathcal{K} u_k, \quad k \in \mathbb{N}_0^{N-1}, \tag{8b}$$

$$y_k = C_\mathcal{K} z_k, \quad k \in \mathbb{N}_0^{N-1}, \tag{8c}$$

$$u_\mathrm{min} \leq u_k \leq u_\mathrm{max}, \quad k \in \mathbb{N}_0^{N-1}, \tag{8d}$$

$$z_0 = \hat{z}_{k|k}, \tag{8e}$$

where z_k denotes the state in the Koopman space, u_k is the control input, and y_k is the system output. The matrices $A_\mathcal{K}$, $B_\mathcal{K}$, and $C_\mathcal{K}$ represent the Koopman linear dynamics and output mappings.

An important advantage of this formulation is its compatibility with standard MPC frameworks: the same weighting matrices used in the nominal model can be directly applied here.

The model predictive control problem (8) is formulated as a traditional quadratic optimization problem with linear constraints. To remedy the dependence on the high dimensionality of the lifted state space, we can use the dense formulation of the MPC problem as outlined in the Section 2.3. This also follows the principles outlined by Korda and Mezić (2018).

The deep Koopman MPC framework can be divided into an offline phase focused on system identification and controller design and an online phase for real-time execution. The offline phase includes data collection, Koopman operator approximation, model verification, MPC controller formulation, and observer design. The online phase implements real-time state estimation and model predictive control using the previously identified Koopman-based model. The detailed workflow is outlined below.

- Offline Phase (Before Real-Time Execution)
  1. Data Collection
     - Collect open-loop input-output trajectories from the experimental system
     - Split into training and testing datasets
  2. Koopman Operator Identification
     - Obtain an approximation of the Koopman operator using the training data
     - Compute lifted linear system matrices: $A_\mathcal{K}, B_\mathcal{K}, C_\mathcal{K}$ in (7) These two steps are happening at the same time.
  3. Model Verification (if relevant)
     - Verify controllability, observability and reachability of the lifted model
     - Note that the stability of the Koopman model can be enforced in the NN architecture during the

computation of system matrices $A_\mathcal{K}, B_\mathcal{K}, C_\mathcal{K}$ in (7). If the stability is not enforced, a stability check is necessary for the Koopman-based model.
  - Evaluate prediction accuracy using the test dataset
4. MPC Formulation
  - Formulate MPC optimization problem in (8) using $A_\mathcal{K}, B_\mathcal{K}, C_\mathcal{K}$ in (7)
5. Correction of the Lifted States
  - Design a state observer (e.g., Kalman filter) for estimating the lifted states z_k from output measurements y_k
  - Initialize the lifted state $z_0 = g(y_0)$

- Online Phase (Real-Time Execution)

7. State Estimation
  - Use Kalman filter to estimate lifted states z_k at each time step from the measured output y_k
8. MPC Execution
  - Solve the MPC optimization problem in (8) at each time step, based on the estimated lifted states z_k
  - Apply the first control input u_0 to the controlled plant
9. Real-Time Control
  - Tune the weighting matrices of MPC Q_y and Q_u and weighting matrices of Kalman filter Q_KF and R_KF if necessary

4. Pasteurization Unit

The experimental plant considered in this case study is a laboratory-scaled pasteurization unit (PU) made by Armfield displayed in Figure 3. Given the complexity and nonlinear behaviour of these energy-intensive processes, this application aims to demonstrate that using a deep Koopman model will lead to improved prediction accuracy

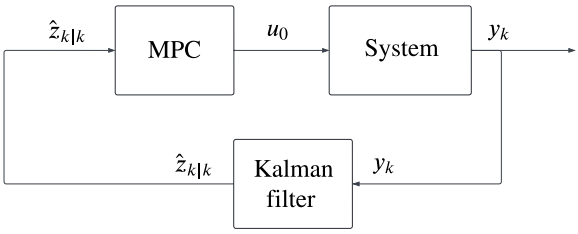


Figure 2: Control scheme of the Deep Koopman MPC Framework.

Figure 3: Experimental device - the pasteurization unit includes a plate heat exchanger (1), a tank for the cold medium (2), a tank for the hot medium with a heating system (3), a peristaltic pump for the cold medium (4), and a peristaltic pump for the hot medium (5).

and, when combined with MPC, improve closed-loop performance compared to subspace identification methods. The scheme of the plant can be seen in Figure 4. The system is a multiple-input-multiple-output (MIMO) dynamic process with three manipulated variables and three output variables. It represents a scaled-down version of an industrial pasteurization system.

Pasteurization is a continuous process where the primary control objective is to maintain the temperature at predefined setpoints based on the specific product being processed. The setup includes two feed tanks that store the cold liquid to be pasteurized. This cold liquid is pumped by a feed pump into the heat recovery chamber of a heat exchanger, and the flow rate of the cold liquid u_1 represents the first manipulated variable. In the heat recovery chamber, the cold liquid absorbs residual heat from already pasteurized products exiting the unit.

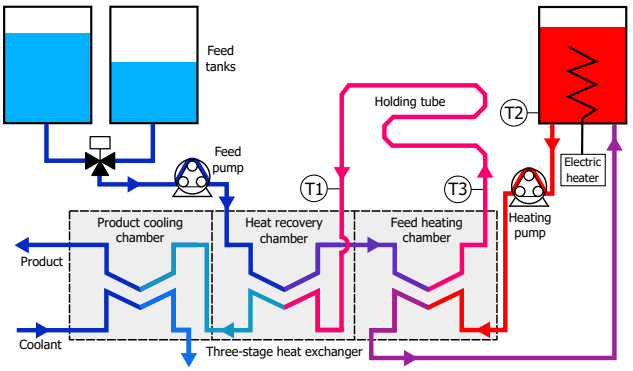


Figure 4: Scheme of experimental device - pasteurization unit. The variables in the scheme have the following meaning: u_1 - flow rate of the feed pump, u_2 - flow rate of the hot water pump heating pump, u_3 - the power to the heating spiral in the boiler electric heater, y_1 - the temperature at the end of a holding tube T1, y_2 - the temperature in the boiler T2, y_3 - the temperature at the output from a heat exchanger T3.

Next, the cold liquid enters the main heating chamber, where it is heated by a hot medium circulated through a closed loop by a heating pump. The flow rate of the hot medium u_2 represents the second manipulated variable. The hot water is stored in a tank equipped with an electric heater. Its heating power u_3 is the third manipulated variable. In the scheme (Figure 4), the temperature of this hot medium is referred to as T2. At the same time, the temperature of the hot medium inside the heating tank y_2 is the second controlled variable.

Once the pasteurized liquid reaches the required pasteurization temperature (T3 in Figure 4), which is the third controlled variable y_3, it flows through an insulated holding tube designed to maintain this temperature long enough to neutralize any pathogens. The temperature measured at the end of the holding tube y_1 is the first controlled variable. The heated and pasteurized product, now at temperature T1 in Figure 4, first passes again through the heat recovery chamber—where it transfers heat to incoming cold liquid—before continuing to the final cooling chamber. There, it releases any remaining heat to a coolant medium. The designed controller aims to maintain the reference pasteurization temperature.

5. Model of the Pasteurization Unit

To identify the process model of the pasteurization unit (PU), we conducted a series of step-change experiments. For the pump actuators, step changes were introduced every 250 s, while for the heating spiral within the boiler, changes were applied every 500 s. These perturbations covered the full operating range of all three manipulated variables, including their upper and lower bounds.

The system outputs were recorded at a sampling rate of 1 s, and data collection spanned approximately one hour. Two separate experiments were conducted: the first served as the testing dataset, while the second was designated for model training. The training dataset comprised a total of 12 step changes for the pump inputs, whereas the testing dataset included 10 step changes for the pumps and approximately half as many for the heating spiral.

For both models, data scaling was applied prior to the identification process. This step was particularly important for the Koopman model, which relies on neural network training and is sensitive to the scale of input features. The scaling was performed using the `StandardScaler` utility from the `scikit-learn` library. Inputs and outputs were scaled separately to standardize their distributions.

5.1. Subspace Identification Model

For the subspace identification of the PU model (see Section 2.2), we utilized the System Identification Toolbox in MATLAB, version 2024b (The MathWorks, Inc., 2024).

The lowest MSE was achieved with a state-space model comprising the following dimensions: $A \in \mathbb{R}^{3\times3}$, $B \in \mathbb{R}^{3\times3}$, and $C \in \mathbb{R}^{3\times3}$. Increasing the number of states beyond this point resulted in a higher MSE on the test dataset, indicating overfitting. The state-space representation employed an identity output matrix, corresponding to full-state measurement.

5.2. Model of the Deep Koopman Approach

The deep Koopman model (see Section 2.1) was trained using the designated training dataset.

Table 1: Summary of key hyperparameters used during training.

Component	Value
Optimizer	ADAM
Learning Rate	0.001
Bias Initialization	0.0
Weights Initialization	Kaiming Uniform (He)
Stopping Criterion	Early Stopping
Warm up Steps	100
Patience	200
Epochs	2000
Batch Size	80

The architecture employed was inspired by an autoencoder structure. Specifically, the encoder (lifting) network consisted of three fully connected layers with 60, 120, and 180 neurons, respectively, each followed by a ReLU activation function. This architecture resulted in a lifted latent state vector z of dimension 30. The dimension for lifted states was chosen based on the controllability criteria. Initially, we set a higher number of lifted states, but the controllability criteria weren't satisfied. Then, we run the training again, lowering the number of lifted states to a maximum number of linearly independent rows of the controllability matrix. After a few iterations, we obtained a system that is controllable with a maximum of 30 lifted states.

Accordingly, the learned Koopman matrices have the following dimensions: $A_{\mathcal{K}} \in \mathbb{R}^{30\times30}$, $B_{\mathcal{K}} \in \mathbb{R}^{30\times3}$, and $C_{\mathcal{K}} \in \mathbb{R}^{3\times30}$.

The training parameters are summarised in Table 1. The training of the Koopman model was done on a MacBook Pro with an Apple M1 Pro processor.

5.3. Open-Loop Performance of the Models

In this subsection, we discuss the analysis of model identification performance between the N4SID and deep Koopman approaches. Both models, trained solely on the training dataset, were evaluated using open-loop predictions on the test dataset. This means that they only obtained measurement of the output in the first time step and applied control inputs. The quantitative results are summarized in Table 2.

The evaluation reveals that the deep Koopman model achieves approximately twice the prediction accuracy of the N4SID model. Specifically, it exhibits a lower mean-squared error (MSE) across all outputs.

6. Control Setup

To evaluate the performance of the identified models in a realistic setting, we implemented MPC closed-loop control using the PU. The experiments were designed to include both heating and cooling scenarios by changing the temperature from an initial steady state with higher temperature to a new steady state with lower temperature ($\approx 50\,^{\circ}\mathrm{C}$) and vice versa. This approach reflects practical operating scenarios such as start-up, shutdown, or product switching. To further investigate the model accuracy, we tuned the weighting matrices Q_{y} and Q_{u} in the MPC cost function, analyzing various weights on control effort and tracking precision. These experiments provide insight into each model's ability to deliver accurate, energy-efficient closed-loop performance under different control priorities. Pasteurization at $\approx 50\,^{\circ}\mathrm{C}$ to $60\,^{\circ}\mathrm{C}$ is considered a low-temperature, long-time method and is typically used for heat-sensitive products like egg yolks or honey, requiring extended time to reduce microbes without compromising quality (Robertson and Muriana, 2004). The control setup for the pasteurization unit follows the same scheme as outlined in Figure 2.

For the closed-loop control experiments, several scenarios were considered, differing only in the choice of weighting matrices used in the objective function. To ensure a fair comparison of the control performance across models, the control configuration was kept consistent across both obtained models. The only variation was the underlying model (deep Koopman or N4SID) and the corresponding state estimator.

Matrices of Kalman filter were set as Q_{KF} an identity matrix scaled by 0.1 with the dimensions corresponding to specific model and $R_{\mathrm{KF}} \in$

Table 2: Comparison of models (normalized as 100% stands for the performance of the N4SID model).

Model	MSE(y_1)	MSE(y_2)	MSE(y_3)	Mean MSE
N4SID	100.0	100.0	100.0	100.0
Deep Koopman	55.5	32.6	52.2	48.7
Improvement	**44.5%**	**67.4%**	**47.8%**	**51.3%**

$\mathbb{R}^{n_y \times n_y}$ an identity matrix scaled by a scaling factor of 0.5. P_0 was set to an identity matrix with the corresponding dimensions. The covariance matrices Q_{KF} and R_{KF} of the Kalman filter were initialised and systematically tuned using experiments to achieve satisfactory control. The final values were kept identical across all scenarios and both models.

As noted by Simon (2006), the Kalman filter can be initialised with any reasonable prior guess of the state, with the uncertainty of this guess reflected in the initial covariance P_0. In practice, when no prior information about z_0 is available, it is common to use the first measurement y_0 through the measurement equation $y_0 = Cz_0$ to form an initial guess, for example $\hat{z}_0 \approx C^\dagger y_0$, where $C^\dagger$ denotes the pseudoinverse of C. From our computation of test prediction error, this approach is similarly ineffective as using the initial condition $\hat{z}_0 = \mathbf{0}$ directly, as shown in Figure 5. $\mathbf{0}$ denotes the zero vector of appropriate dimension. The smallest error was achieved by using the initial condition $\hat{z}_0 = g(y_0)$. The error of this approach is reflected in the initial covariance P_0.

At each time step (sampled every second), the optimal control input was computed and applied to the PU. The control input was obtained by solving the optimization problem either using the Koopman-based model (8) or the N4SID model (6). The prediction horizon was set to $N = 20$, with input constraints defined as:

$$u_{\min} = \begin{bmatrix} 30 \\ 20 \\ 10 \end{bmatrix}, \quad u_{\max} = \begin{bmatrix} 100 \\ 100 \\ 40 \end{bmatrix}. \tag{9}$$

The weighting matrix for the control input, Q_{u}, remained the same across all CS:

$$Q_{\text{u}} = \text{diag}\left(\begin{bmatrix} 1, & 1, & 1 \end{bmatrix}\right). \tag{10}$$

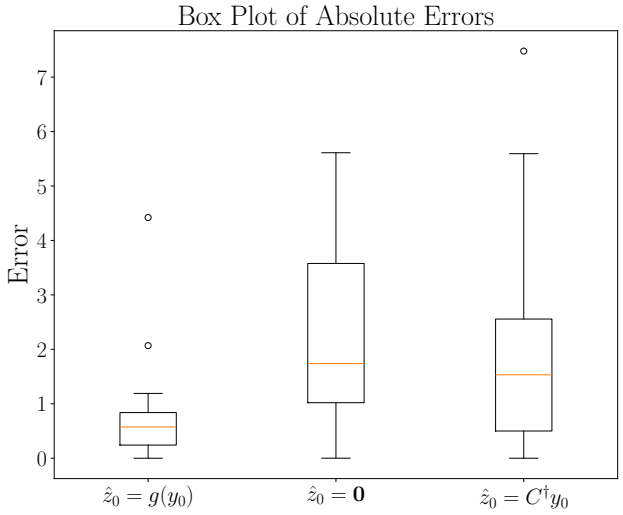


Figure 5: Box plot of absolute prediction errors for different initialization strategies.

In contrast, the output weighting matrix Q_{y} was varied across three control setups, as can be seen in Table 3

Table 3: Control setup variations and corresponding output weighting matrix Q_{y}. Thanks to the scaling of the data, there is no need to scale specific values in the weighting matrix.

CS	Q_{y}	Description
#1	diag(5, 5, 5)	Lower uniform
#2	diag(20, 5, 20)	Higher on y_1 and y_3
#3	diag(15, 15, 15)	Higher uniform

Prior to each set of trials, the steady-state output was computed as: $u_{\text{s}} = [65.8\ 54.5\ 25.0]^\intercal$. Each experiment began with an open-loop phase lasting 10 min, during which the PU reached a new steady state. Following this phase, the MPC controller was operated in closed-loop mode for an additional 10 min.

7. Results and Discussion

In this section, we present and analyse the experimental results from the closed-loop control conducted using the laboratory PU.

One of our main contributions focuses on introducing a state observer to correct lifted states in Koopman MPC. Figure 6 illustrates the impact of the lifted states correction. A naive initialisation $z_k = g(y_k)$, as we can see in many applications, for example in Han et al. (2020), Han et al. (2024, 2021); neglects historical information and amplifies the impact of process-model mismatch and noise, resulting in steady-state offsets and a significant decrease in control performance. This leads to increased product waste and overall is not suitable for deployment in real-world conditions. On the other hand, incorporating the Kalman filter correction, as we propose, removes the offset and restores accurate setpoint tracking. Importantly, this demonstrates that the correction is not merely a performance enhancement, but a necessity for reliable real-time application of Koopman MPC in practice. Note that the Kalman filter correction is not necessary for the N4SID model, as it is in a form of full-state measurement.

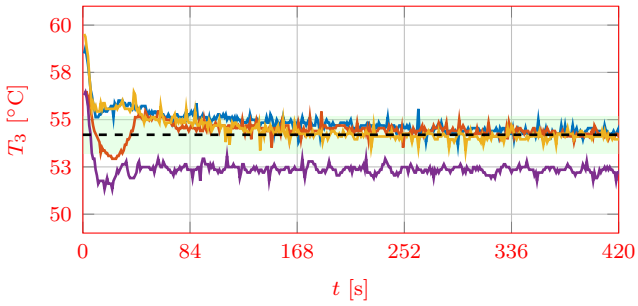


Figure 6: The T_3 steady-state tracking performance of the used algorithms with and without Kalman filter correction. The blue line represents the MPC using the N4SID model and Kalman filter, while the red line represents the MPC using the deep Koopman model and Kalman filter. The yellow line is for the MPC using the N4SID model without a Kalman filter, and the purple line is for the MPC using the deep Koopman model without a Kalman filter. The Black dashed line is the setpoint. The light green area indicates the $\pm 1\,^{\circ}\mathrm{C}$ tolerance band around the setpoint, which is considered the steady-state once the response remains within it.

The performance of the models identified using the Koopman operator and N4SID is compared in terms of tracking accuracy, control effort, and energy efficiency. We evaluate the impact of varying the MPC weighting matrices and examine how each model performs during both the heating and cooling phases. To the best of the authors' knowledge, this is one of the earliest experimental applications of Koopman MPC overall, especially in the context of energy-intensive plants. Experiments were conducted for each CS, and a summary of the results is provided in Table 4. We refer to the analysed models used in the MPC framework as Deep Koopman (DK) MPC and N4SID MPC. In Table 4, the CS correspond to those described in Section 6, with an upward (↑) or downward (↓) arrow indicating negative or positive of deviation from the initial steady state.
The variable E represents the energy used to heat the heating liquid in the boiler (using the electric heater), which is the most energy-intensive process in the system. As such, it serves as a key metric for evaluating energy efficiency. The second metric is the Mean Integral Absolute Error (MAE), which reflects the control performance. MAE is defined as the integral of the mean absolute error over time, where the mean is computed across all three control outputs. This provides a measure of how closely the temperatures in the PU track its desired setpoints throughout the process. A higher MAE in a control setup indicates underperforming temperature control. This may result in insufficient heating, which fails to eliminate pathogenic bacteria, or excessive heating, which can degrade essential bioactive compounds. As a result, a larger amount of liquid must be discarded to ensure product safety and quality. Therefore, improved control performance of the PU, i.e., lower MAE, minimizes product waste.

For CS # 3, we also provide visualizations of system outputs and the corresponding control inputs. The results of control from above the steady state are shown in Figure 7 (outputs) and Figure 8 (inputs), while results from below the steady state are shown in Figures 9 and 10, respectively.

A consistent pattern is observed across all experiments: the DK model consistently achieves

significantly lower MAE values compared to the N4SID model, while the increase in energy consumption is negligible — typically less than 1%, or even reduced in certain cases (e.g., CS # 3 ↑). In CS # 2 ↑, the experiment successfully mitigated the influence of the most inaccurately modelled variable by assigning higher weights to the other two outputs. Even in this case, the DK model achieved more than a 9% reduction in MAE compared to the N4SID model.

For the displayed experiments in Figures 7 and 9, the DK model achieved an average MAE reduction of 22.2% and a corresponding decrease in energy consumption of 0.5%. For these simulations also, a criterion of closed-loop value of objective function was calculated according to: To evaluate the closed-loop performance from experimental data, we compute the cumulative cost as:

$$J = \sum_{t=1}^{N_{\text{exp}}} y_t^{\intercal} Q_{\text{y}} y_t + u_t^{\intercal} Q_{\text{u}} u_t, \tag{11}$$

where N_{exp} denotes the total number of time steps in the experiment, and y_t, u_t are the measured outputs and control inputs at each control step t. The DK model consistently outperformed the N4SID model in shown control setups, with the value of closed-loop objective function (11) averagely reduced by 8.5%.

Table 4: Comparison of N4SID and deep Koopman models in the control of the PU (normalized as 100% stands for the performance of using the N4SID model).

CS	E_{N4SID}	E_{DK}	$\text{MAE}_{\text{N4SID}}$	MAE_{DK}
#1 ↓	100.0	100.2	100.0	69.2
#1 ↑	100.0	100.3	100.0	82.7
#2 ↑	100.0	100.3	100.0	90.9
#3 ↑	100.0	98.2	100.0	81.8
#3 ↓	100.0	100.8	100.0	73.8

For the evaluation of steady-state performance, we computed the same metrics as in Table 4. We consider a steady-state to be achieved once the response remains within the ±1 °C tolerance band around the setpoint. In this case, when the system achieves this condition at different times with each considered control setup (CS), we normalised the metrics accordingly. The results are summarised in Table 5. The DK model outperformed the N4SID model in all control setups, with an average steady-state MAE reduction of almost 30% and an average conservation of steady-state energy consumption of 1.0%. The reduction of the steady-state closed-loop objective function (11) was more significant, averaging 45.7%.

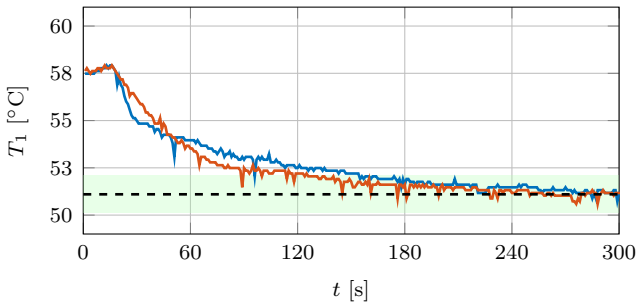
Overall, the DK model outperformed the N4SID model in all control scenarios, consistently delivering better control accuracy while maintaining, or in some cases improving energy efficiency.

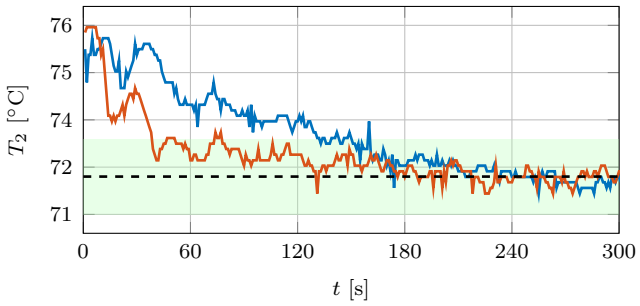
For completeness, we also benchmarked the proposed Koopman MPC against a linear–quadratic-integral (LQI) controller designed using an alternative N4SID model of the same plant, presented in Furka (2023). While this baseline is evaluated under different operating conditions, we computed performance using the same MAE and energy definitions and normalised them by the total reference change ΔT. The Koopman MPC achieved approximately 66% lower normalised MAE and 43% lower normalised energy consumption relative to LQI. The comparison confirms that Koopman MPC outperforms both predictive and conventional baselines.

Table 5: Steady state comparison of N4SID and deep Koopman models in the control of the PU (normalized as 100% stands for the performance of using the N4SID model).

CS	E_{N4SID}	E_{DK}	$\text{MAE}_{\text{N4SID}}$	MAE_{DK}
#1 ↓	100.0	98.2	100.0	68.6
#1 ↑	100.0	98.4	100.0	84.8
#2 ↑	100.0	101.2	100.0	81.1
#3 ↑	100.0	97.3	100.0	37.6
#3 ↓	100.0	100.0	100.0	82.3

As the DK model inherently increases its dimensionality, we also evaluated the computational load of the MPC controller when it was implemented with DK and N4SID models. Both the N4SID and DK models were able to compute optimal control actions under 0.01 s, indicating that real-time execution within one sampling period is





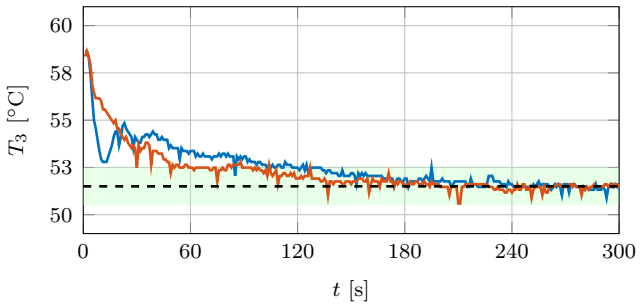
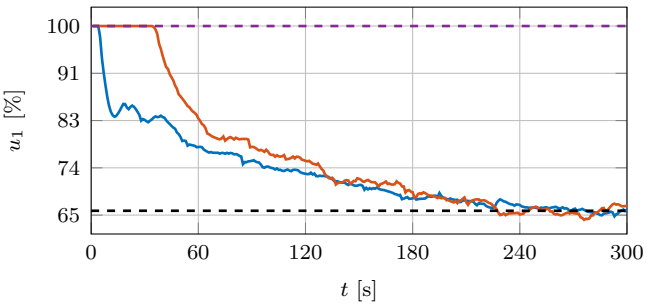
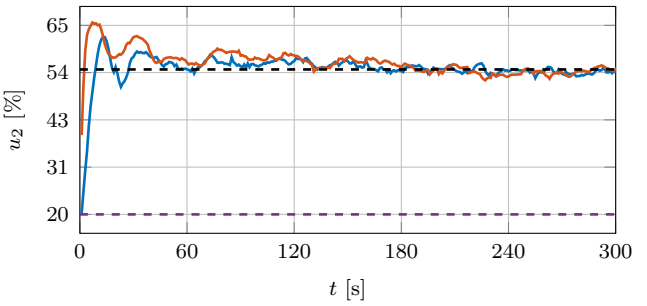


Figure 7: Control results for the controlled variables of the CS #3 ↓. The blue line is for the MPC using the N4SID model, and the red line is for the MPC using the deep Koopman model. The Black dashed line is the setpoint. The light green area indicates the ±1 °C tolerance band around the setpoint, which is considered the steady state once the response remains within it.





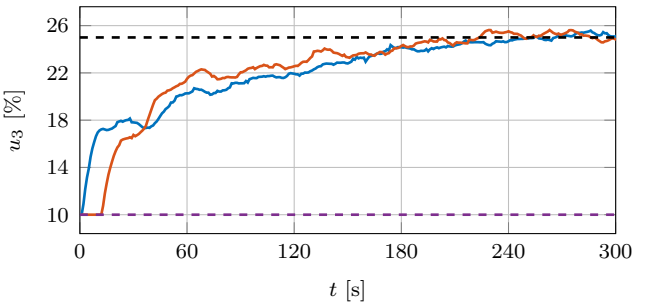


Figure 8: Control inputs for the CS #3 ↓. The blue line is for the MPC using the N4SID model, and the red line is for the MPC using the deep Koopman model. The Black dashed line is the setpoint, and the purple dashed line is the input constraint.

feasible for both approaches.

As described in Section 3.3, a dense formulation of the optimization problem was used to reduce the dependence of computational complexity on the dimensionality of the lifted space. However, this approach requires the pre-computation of matrices used in the dense formulation, which can become computationally demanding for large models or long prediction horizons.

Table 6 presents the average computation times measured across 2500 different state scenarios during simulation. The timing was performed on an Apple M1 Pro processor and includes the time required for matrix construction, optimization, and total execution.

Table 6: Comparison of offline construction time (OCT) for constructing the state space matrices and online solver time (OST) solving the MPC optimization problem for both models.

Model	n_x	OCT [ms]	OST [ms]
N4SID	3	2.370	5.642
DK	30	3.779	4.956

As expected, the offline construction of matrices for the deep Koopman model is slightly more computationally intensive than for the N4SID model due to its higher state dimensionality. On the other hand, the online optimization problem is solved slightly faster when using the deep Koopman model. This occurs because the optimization problem is warm-started using the previous solution, which, in the case of the deep Koopman model, is closer to the optimal solution of the current problem compared to the N4SID model. Therefore, the online computation time is about 13% slower when using the N4SID model. The MPC optimization problem was solved using the Gurobi solver. The computations were done on a MacBook Pro with an Apple M1 Pro processor.

From a practical control engineering perspective, the total computational time remains negligible for both approaches, providing sufficient buffer within a 1 s sampling interval.

8. Conclusions

In this paper, we presented a novel framework for real-time control using deep Koopman Model Predictive Control. This work highlights the potential of deep learning techniques in enhancing the performance of model-based control strategies, particularly in complex energy-intensive industrial processes. The experimental results indicate that the deep Koopman model can effectively capture the underlying dynamics of the system, leading to improved control performance. Furthermore, the deep Koopman model is linear in the lifted space, and neural networks are used only for the lifting. The linear model can be easily interpreted and analysed.

Considering the deep Koopman model, we demonstrated significant improvements in control accuracy compared to traditional N4SID methods using an energy-intensive laboratory pasteurization unit. This is evidenced by lower Mean Absolute Error values across all control setups. This improvement directly relates to reduced waste and improved energy efficiency, as fewer inferior quality products are produced and subsequently wasted. These improvements were achieved without a significant increase in energy consumption, and in some cases, energy usage was even reduced. The computational load of the deep Koopman model was also investigated, as the deep Koopman model inherently increases the dimensionality of the model. The total computation times of the deep Koopman model were comparable to those of the N4SID model. Future work will focus on developing an offset-free Koopman Model Predictive Control framework. This will involve addressing steady-state offset issues by integrating disturbance modeling or state estimation techniques, ensuring improved tracking performance and robustness in practical applications.

9. Acknowledgment

The authors M. Klaučo and P. Valábek gratefully acknowledge the contribution of the Scientific Grant Agency of the Slovak Republic under the grants VEGA 1/0239/24, the Slovak Research and Development Agency under

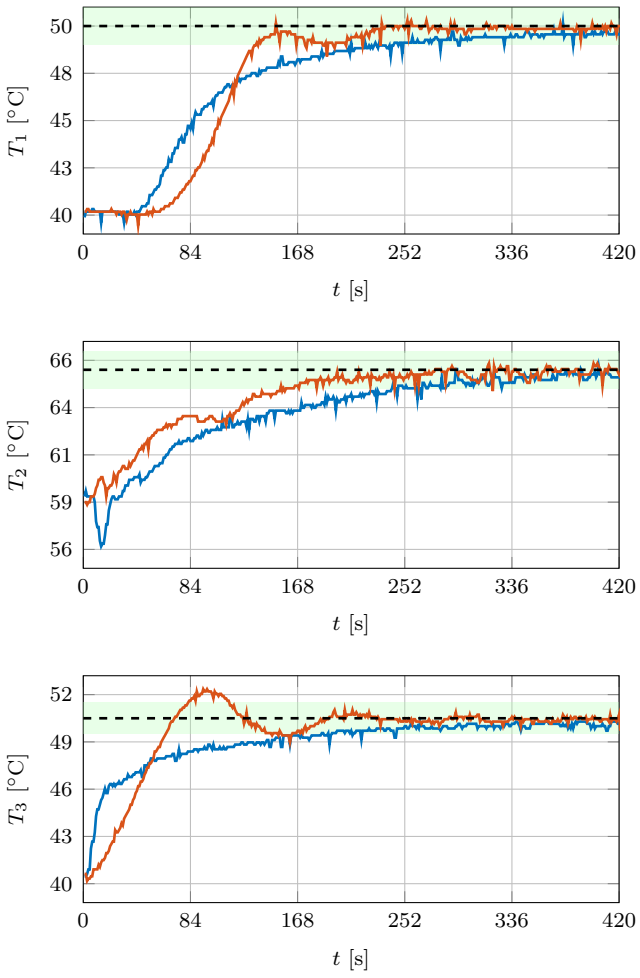


Figure 9: Control results for the controlled variables of the CS #3 ↑. The blue line is for the MPC using the N4SID model, and the red line is for the MPC using the deep Koopman model. The Black dashed line is the setpoint. The light green area indicates the ±1 °C tolerance band around the setpoint, which is considered the steady state once the response remains within it.

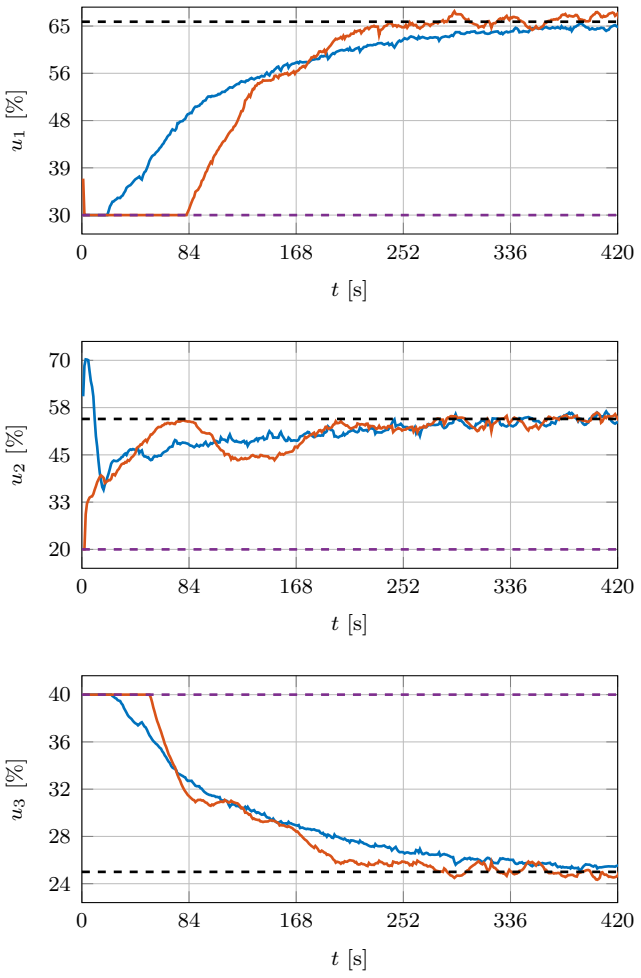


Figure 10: Control inputs for the CS #3 ↑. The blue line is for the MPC using the N4SID model, and the red line is for the MPC using the deep Koopman model. The Black dashed line is the setpoint, and the purple dashed line is the input constraint.

the project APVV-20-0261. P. Valábek is also supported by an internal STU grants for young researchers. M. Klaučo is also supported by the European Union project ROBOPROX (Reg. No. CZ.02.01.01/00/22_008/0004590). M. Horváthová acknowledges the contribution of the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00636. The manuscript was prepared under the auspices of the Slovak University of Technology in Bratislava, Slovakia, while the final submission and revisions were carried out in collaboration with the Czech Technical University in Prague, Czechia.

References

N. H. Martin, K. J. Boor, M. Wiedmann, Symposium review: Effect of post-pasteurization contamination on fluid milk quality, Journal of Dairy Science 101 (2018) 861–870. doi:https://doi.org/10.3168/jds.2017-13339.

J. Shieh, H. Chen, L. Ferng, Application of a fuzzy logic controller in temperature control of a pilot high-temperature short-time heat exchanger, Food Control 3 (1992) 91–96.

K. R. Morison, Steady-state control of plate pasteurisers, Food Control 16 (2005) 23–30.

J. Ibarrola, J. Sandoval, M. Garcıa-Sanz, M. Pinzolas, Predictive control of a high temperature–short time pasteurisation process, Control Engineering Practice 10 (2002) 713–725.

S. Hadisupadmo, E. Leksono, et al., Model predictive control design and performance analysis of a pasteurization process plant, in: 2016 International Conference on Instrumentation, Control and Automation (ICA), IEEE, 2016, pp. 81–87.

T. M. Thostrup, K. E. Pedersen, S. Tronci, M. Errico, Dynamic simulator and model predictive control of a milk pasteurizer, IFAC-PapersOnLine 55 (2022) 538–543.

C. Riverol, G. Ricart, C. Carosi, C. Di Santis, Application of advanced soft control strategies into the dairy industry, Innovative Food Science & Emerging Technologies 9 (2008) 298–305.

D. Dzurková, O. Mészáros, M. Kalúz, Privacy preserving approximated optimal control of pasteurization unit using homomorphic encryption, IFAC-PapersOnLine 58 (2024) 544–549. doi:https://doi.org/10.1016/j.ifacol.2024.07.275, 12th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes SAFEPROCESS 2024.

B. O. Koopman, Hamiltonian systems and transformation in hilbert space, Proceedings of the National Academy of Sciences 17 (1931) 315–318.

I. Mezić, Spectral properties of dynamical systems, model reduction and decompositions, Nonlinear Dynamics 41 (2005) 309–325.

P. J. Schmid, Dynamic mode decomposition of numerical and experimental data, Journal of fluid mechanics 656 (2010) 5–28.

M. O. Williams, I. G. Kevrekidis, C. W. Rowley, A data–driven approximation of the koopman operator: Extending dynamic mode decomposition, Journal of Nonlinear Science 25 (2015) 1307–1346.

B. Lusch, J. N. Kutz, S. L. Brunton, Deep learning for universal linear embeddings of nonlinear dynamics, Nature communications 9 (2018) 4950.

Q. Li, F. Dietrich, E. M. Bollt, I. G. Kevrekidis, Extended dynamic mode decomposition with dictionary learning: A data-driven adaptive spectral decomposition of the koopman operator, Chaos: An Interdisciplinary Journal of Nonlinear Science 27 (2017).

P. Valábek, M. Wadinger, M. Kvasnica, M. Klaučo, Deep dictionary-free method for identifying linear model of nonlinear system with input delay, in: 2025 25th International Conference on Process Control (PC), IEEE, 2025, pp. 1–6.

M. Korda, I. Mezić, Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control, Automatica 93 (2018) 149–160.

D. Bruder, X. Fu, R. B. Gillespie, C. D. Remy, R. Vasudevan, Data-driven control of soft robots using koopman operator theory, IEEE Transactions on Robotics 37 (2020) 948–961.

M. Švec, Š. Ileš, J. Matuško, Optimizing vehicle handling through koopman-based model predictive torque vectoring: An experimental investigation, Control Engineering Practice 158 (2025) 106272.

T. Zhao, M. Yue, J. Wang, Deep-learning-based koopman modeling for online control synthesis of nonlinear power system transient dynamics, IEEE Transactions on Industrial Informatics 19 (2023) 10444–10453.

D. Huo, A. Adunyah, C. M. Hall, Real-time application of koopman-based optimal control strategies for fuel cell stack thermal management, Control Engineering Practice 156 (2025) 106225.

M. Han, Z. Li, X. Yin, X. Yin, Robust learning and control of time-delay nonlinear systems with deep recurrent koopman operators, IEEE Transactions on Industrial Informatics (2023).

Y. Jin, L. Hou, S. Zhong, Extended dynamic mode decomposition with invertible dictionary learning, Neural networks 173 (2024) 106177.

J. L. Proctor, S. L. Brunton, J. N. Kutz, Dynamic mode decomposition with control, SIAM Journal on Applied Dynamical Systems 15 (2016) 142–161.

S. L. Brunton, J. L. Proctor, J. N. Kutz, Discovering governing equations from data by sparse identification of

nonlinear dynamical systems, Proceedings of the national academy of sciences 113 (2016) 3932–3937.

T. Katayama, Subspace Methods for System Identification, Communications and Control Engineering, Springer London, London, 2005. doi:10.1007/1-84628-158-X.

H. Shi, M. Q.-H. Meng, Deep koopman operator with control for nonlinear systems, IEEE Robotics and Automation Letters 7 (2022) 7700–7707.

Y. Han, W. Hao, U. Vaidya, Deep learning of koopman representation for control, in: 2020 59th IEEE Conference on Decision and Control (CDC), IEEE, 2020, pp. 1890–1895.

The MathWorks, Inc., System Identification Toolbox, The MathWorks, Inc., Natick, Massachusetts, United States, 2024. URL: https://www.mathworks.com/products/sysid.html, version R2024a.

W. Robertson, P. Muriana, Reduction of salmonella by two commercial egg white pasteurization methods, Journal of Food Protection 67 (2004) 1177–1183. doi:https://doi.org/10.4315/0362-028X-67.6.1177.

D. Simon, Optimal state estimation: Kalman, H infinity, and nonlinear approaches, John Wiley & Sons, 2006.

M. Han, J. Yao, A. W.-K. Law, X. Yin, Efficient economic model predictive control of water treatment process with learning-based koopman operator, Control Engineering Practice 149 (2024) 105975.

M. Han, J. Euler-Rolle, R. K. Katzschmann, Desko: Stability-assured robust control with a deep stochastic koopman operator, in: International Conference on Learning Representations, 2021.

M. Furka, Process Control Security Supervised by Homomorphic Encryption, Ph.D. thesis, 2023.