Data-Driven Model Reduction and Nonlinear Model Predictive Control of an Air Separation Unit by Applied Koopman Theory

Jan C. Schulze [a], Danimir T. Doncevic [b], Nils Erwes [a] and Alexander Mitsos [c,a,b,1]

[a] Process Systems Engineering (AVT.SVT), RWTH Aachen University, 52074 Aachen, Germany

[b] Institute of Energy and Climate Research: Energy Systems Engineering (IEK-10), Forschungszentrum Jülich GmbH, 52425 Jülich, Germany

[c] JARA-CSD, 52056 Aachen, Germany

Abstract

Achieving real-time capability is an essential prerequisite for the industrial implementation of nonlinear model predictive control (NMPC). Data-driven model reduction offers a way to obtain low-order control models from complex digital twins. In particular, data-driven approaches require little expert knowledge of the particular process and its model, and provide reduced models of a well-defined generic structure. Herein, we apply our recently proposed data-driven reduction strategy based on Koopman theory [Schulze et al. (2022), Comput. Chem. Eng.] to generate a low-order control model of an air separation unit (ASU). The reduced Koopman model combines autoencoders and linear latent dynamics and is constructed using machine learning. Further, we present an NMPC implementation that uses derivative computation tailored to the fixed block structure of reduced Koopman models. Our reduction approach with tailored NMPC implementation enables real-time NMPC of an ASU at an average CPU time decrease by 98 %.

Keywords

Data-driven model reduction, Autoencoder, Koopman theory, Nonlinear model predictive control, Air separation unit

Introduction

Computationally tractable models are a main requirement for real-time NMPC (Marquardt, 2002). Data-driven non-intrusive model reduction comprises a class of model-free methods for producing low-order representations of high-order dynamical systems from data, e.g., Antoulas et al. (2017). Similar to classical model reduction approaches (Marquardt, 2002), these data-driven methods project a high-order system from the full state space to a lower dimensional subspace, reducing the system to its dominant latent patterns. However, applying data-driven approaches shifts the reduction efforts from system-specific expert knowledge for reduced modeling towards data-based system identification. Notably, data-driven approaches allow for automated reduction frameworks as well as exploitation of the generic reduced model structure in optimization.

In recent years, a variety of data-driven frameworks for model reduction of dynamical systems has been presented in the literature, e.g., the Loewner framework (Antoulas et al., 2017), lift-and-learn methods (Qian et al., 2020), and autoencoder network structures (Watter et al., 2015). In particular, Koopman theory (Mezić, 2005) has attracted considerable attention. Model reduction using Koopman theory as well as the related dynamic mode decomposition (Schmid, 2010), build on a lift-and-project concept and aim to construct linear representations of nonlinear dynamics through (nonlinear) coordinate transformation. Applied Koopman theory has a system-theoretic foundation and naturally combines simple dynamic forms with data-driven identification of coordinate transformations, e.g., through Kernel methods (Williams et al., 2015), deep learning (Lusch et al., 2018), or sparse regression techniques (Brunton et al., 2016).

While Koopman theory was originally developed for autonomous systems, several extensions to systems with exogenous inputs have been introduced. These works derive linear (Proctor et al., 2016; Korda and Mezić, 2018) as well as bilinear (Surana, 2016) Koopman control models. Recently, we proposed a Wiener-type Koopman form (Schulze et al., 2022). More specifically, we developed a Koopman-based deep learning framework that combines autoencoders networks with linear latent dynamic blocks. Further, we demonstrated that such Wiener-type Koopman models can offer a favorable compromise between linear dynamics and nonlinear modeling. Related works using autoencoders and latent dynamics have also been presented independent of Koopman theory, e.g., Watter et al. (2015); Masti and Bemporad (2021).

[1] Corresponding author. Email: `amitsos@alum.mit.edu`.

Despite the potential of Koopman-based NMPC for process control, applications have been limited to single unit operations like a reactor (Narasingam and Kwon, 2019) and a distillation column (Schulze and Mitsos, 2022). Herein, we aim to demonstrate that applied Koopman theory enables real-time NMPC of complex processes, while requiring little expert knowledge for the model reduction procedure. We use our deep learning framework to train Wiener-type Koopman models on simulation data of a detailed digital process twin, whose solution computation requirements are prohibitive for online applications. As a further extension of our previous work (Schulze and Mitsos, 2022), we present an NMPC implementation that is tailored to the reduced Koopman structure, thereby realizing an additional speed-up in online optimization. Finally, we apply the resulting Koopman NMPC framework to an ASU case study. Our numerical results demonstrate the real-time capability and tracking performance of Koopman NMPC in load change scenarios.

Koopman-based model reduction framework

Koopman theory postulates the global linearization of nonlinear autonomous dynamics by means of nonlinear coordinate lifting to a high (generally infinite) dimensional coordinate space (Mezić, 2005). For practical application of Koopman theory, finite truncation of this lifting is sought, often using data-driven methods, e.g., Williams et al. (2015); Lusch et al. (2018). We consider the class of asymptotically stable input-affine systems:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t)) + \sum_{i=1}^{n_u} \boldsymbol{g}_i(\boldsymbol{x}(t)) u_i(t) \,, \tag{1a}$$

$$\boldsymbol{y}(t) = \boldsymbol{h}(\boldsymbol{x}(t)) \,, \tag{1b}$$

where $\boldsymbol{x}(t) \in \mathbb{R}^{n_x}$ are the differential states, $\boldsymbol{y}(t) \in \mathbb{R}^{n_y}$ are the outputs, $\boldsymbol{u}(t) \in \mathbb{R}^{n_u}$ are external inputs, and $\boldsymbol{f} : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$, $\boldsymbol{g}_i : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$, $\boldsymbol{h} : \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$ are continuously differentiable. For this system class, we have derived a Koopman representation of Wiener form (Schulze et al., 2022; Schulze and Mitsos, 2022), which uses linear time-invariant dynamics (LTI) in time-discrete form (zeroth-order hold), sandwiched by nonlinear coordinate transformations:

$$\boldsymbol{z}_{k+1} = A\boldsymbol{z}_k + B\boldsymbol{u}_k \,, \tag{2a}$$

$$\begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{y}_k \end{bmatrix} = \boldsymbol{T}^{\dagger}(\boldsymbol{z}_k) \,, \tag{2b}$$

$$\boldsymbol{z}_0 = \boldsymbol{T}(\boldsymbol{x}_0) \,. \tag{2c}$$

Herein, $\boldsymbol{z}_k \in \mathbb{R}^{n_z}$ are the Koopman coordinates, k is the time index, $\boldsymbol{T} : \mathbb{R}^{n_x} \to \mathbb{R}^{n_z}$ (encoding) represents the nonlinear transformation to the Koopman coordinates and provides initial conditions, and $\boldsymbol{T}^{\dagger} : \mathbb{R}^{n_z} \to \mathbb{R}^{n_x+n_y}$ (decoding) maps the Koopman coordinates to the original states and outputs. A, B are constant matrices. Since we target model reduction, we focus on $n_z \ll n_x$. Based on sampled process data, we aim to directly identify the discrete-time reduced form, which relieves the need for numerical integration in dynamic optimization. Further, note that the dynamics in z are linear instead of nonlinear, constituting a crucial simplification step for the computation of the Jacobian.

Deep learning implementation

We employ the Wiener-type Koopman form, Eq. (2), for data-driven model reduction. To this end, we adopt our deep learning strategy from (Schulze et al., 2022; Schulze and Mitsos, 2022), where we train reduced models on sampled data from numerical simulations of the full-order model (e.g., a digital twin). Specifically, we use artificial neural networks to learn suitable mappings $\boldsymbol{T}$ and $\boldsymbol{T}^{\dagger}$. To simplify the training, we modify the mapping $\boldsymbol{z} = \boldsymbol{T}(\boldsymbol{x})$ to $\boldsymbol{z} = \boldsymbol{T}(\boldsymbol{x}, \boldsymbol{y})$, allowing us to train autoencoder networks.

We compute the training loss $\mathcal{C}$ as the sum of mean squared error (MSE) terms for single and multi-time-step predictions:

$$\begin{aligned} \mathcal{C} = \; & \frac{1}{s-1} \sum_{k=0}^{s-1} \left\| \begin{bmatrix} \boldsymbol{x}_{k+1} \\ \boldsymbol{y}_{k+1} \end{bmatrix} - \boldsymbol{T}^{\dagger}(\boldsymbol{z}_{k+1}(k)) \right\|_{\text{MSE}} \\ & + \frac{1}{s-1} \sum_{k=0}^{s-1} \left\| \begin{bmatrix} \boldsymbol{x}_{k+1} \\ \boldsymbol{y}_{k+1} \end{bmatrix} - \boldsymbol{T}^{\dagger}(\boldsymbol{z}_{k+1}(0)) \right\|_{\text{MSE}} \,, \end{aligned} \tag{3}$$

where:

$$\boldsymbol{z}_{k+1}(j) := \begin{cases} \boldsymbol{z}_{k+1} = A\boldsymbol{z}_k + B\boldsymbol{u}_k \,, \; k = j, j+1, \ldots \\ \boldsymbol{z}_j = \boldsymbol{T}(\boldsymbol{x}_j, \boldsymbol{y}_j) \,, \end{cases}$$

and s is the number of snapshots per trajectory. The dimension n_z and number of network layers and respective neurons are hyperparameters. If system knowledge is available, a (block) diagonality of A can be prespecified which enforces learning Koopman eigenfunctions and reduces the number of trainable parameters. In contrast to our previous work, we do not formulate an additional autoencoder loss term. Thereby, we promote direct feedthrough of inputs to states and outputs, which improves the reduction procedure with respect to fast modes. Case-specific details on the data sampling and model training procedures are provided within the control study.

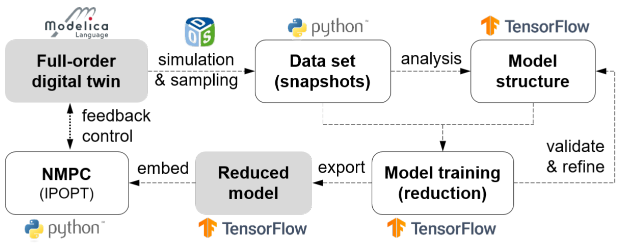

Figure 1: Model reduction and control workflow.

Fig. 1 depicts the model reduction workflow implemented in Python 3.9 and TensorFlow 2.5. We conduct the simulation experiments using our open-source dynamic optimization software DyOS with integrator NIXE (Caspari et al., 2019). However, other simulation environments may be used as well. The preliminary model structure is selected based on analysis of the simulation data set and refined iteratively by the user if necessary. After training, the reduced model is used to control the real process (or as herein, its digital twin).

Koopman NMPC

We employ the reduced Koopman models for NMPC of chemical processes. While Wiener-type Koopman models promise a higher accuracy than linear models (Schulze et al., 2022), their nonlinear type requires nonlinear optimization. However, the sequential Wiener architecture can be exploited by a tailored implementation of the NMPC. Consider the following optimal control problem solved by the NMPC:

$$\min_{\boldsymbol{u},\boldsymbol{z}} \sum_{k=1}^{N_c} \ell_k(\boldsymbol{T}^{\dagger}(\boldsymbol{z}_{k+1})) \tag{4a}$$

$$\text{s.t.} \quad \boldsymbol{z}_{k+1} = A\boldsymbol{z}_k + \boldsymbol{B}\boldsymbol{u}_k\,, \tag{4b}$$

$$\boldsymbol{c}(\boldsymbol{T}^{\dagger}(\boldsymbol{z}_{k+1})) \leq \boldsymbol{0}\,, \tag{4c}$$

$$\boldsymbol{u}_k \in \mathcal{U}\,, \tag{4d}$$

$$\boldsymbol{z}_0 = \boldsymbol{T}(\boldsymbol{x}_0, \boldsymbol{y}_0)\,, \tag{4e}$$

$$k = 0, 1, \ldots, N_c - 1\,, \tag{4f}$$

where N_c is the control horizon and Eq. (4a) describes the cost function with stage cost ℓ_k. Eq. (4b) are the linear dynamics, Eq. (4c) represents nonlinear path constraints on states and outputs, and $\mathcal{U}$ is the admissible set of controls. Eq. (4e) provides process feedback and is evaluated prior to solving the problem. Due to the sequential model structure, the decoding can be directly inserted into the cost function and constraints. Consequently, the control problem is condensed to the variables $\boldsymbol{u}$ and $\boldsymbol{z}$. Further, the sparsity of the Jacobian is known beforehand due to the well-defined model structure. These attributes enable a computationally efficient and reusable Koopman NMPC framework.

We implement Koopman NMPC in Python 3.9 by combining the NLP solver IPOPT (Wächter and Biegler, 2006) and automatic differentiation using TensorFlow. Therein, we tailor the derivative computation to the block-diagonal sparse structure of the Jacobian and compute the non-zero entries using automatic differentiation. We cache the gradient tape to avoid expensive online re-computation. We use piecewise constant control moves (zeroth-order hold) and warm-start consecutive optimizations. In contrast to, e.g., Masti and Bemporad (2021), we exploit the model structure in the optimization rather than embedding the full model.

Fig. 2 visualizes how the underlying graph structure of the model and control problem predetermines the Jacobian. Neighboring $\boldsymbol{z}$ nodes are coupled through LTI dynamics, Eq. (2a), resulting in non-zero constant Jacobian blocks. In addition, decoding branches for cost and constraints are locally attached, Eq. (2b), with non-constant Jacobian blocks.

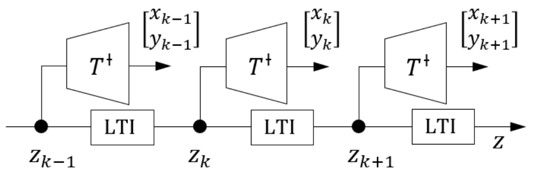
$T^{\dagger}$
$\begin{bmatrix} x_{k-1} \\ y_{k-1} \end{bmatrix}$
$\begin{bmatrix} x_{k} \\ y_{k} \end{bmatrix}$
$\begin{bmatrix} x_{k+1} \\ y_{k+1} \end{bmatrix}$
LTI
z_{k-1}
z_k
z_{k+1}
z

Figure 2: Graph structure of Koopman model prediction.

Case study: NMPC of an air separation unit

Cryogenic air separation units (ASUs) produce industrial gases such as nitrogen and oxygen. Due to their high electric energy demand, ASUs are considered as a promising candidate for load flexible operation and demand side management (Pattison et al., 2016). NMPC is a promising operating strategy for realizing major load changes on a frequent basis (Chen et al., 2010). However, especially the detailed modeling of distillation columns and heat exchangers, e.g., by means of tray-to-tray balancing and finite volume discretization, respectively, results in process models that are too complex for NMPC (Caspari et al., 2020). Thus, different model reduction approaches for individual process units have been investigated, e.g., Chen et al. (2010); Schäfer et al. (2019); Caspari et al. (2020); Schulze et al. (2021). While these physics-motivated approaches preserve the flowsheet structure of the dynamic model, their application requires additional modeling efforts and expert knowledge. In contrast, data-driven reduction is particularly suitable if the flowsheet structure does not need to be preserved and the size and complexity of the input-output data sets is thus limited.

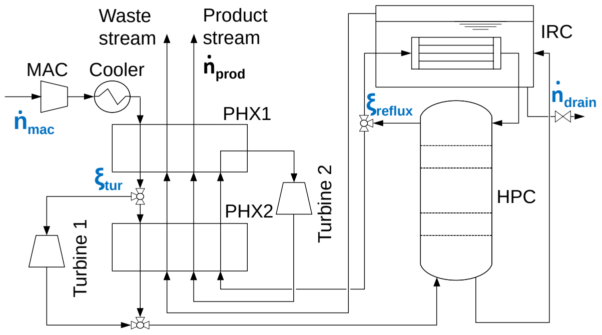

Figure 3: Air separation unit. Manipulated variables in blue.

We consider the ASU depicted in Fig. 3, which serves as a classic literature problem (Pattison et al., 2016). The process consists of the main air compressor (MAC), multi-stream heat exchangers (PHX), turbines, high-pressure distillation column (HPC) and integrated reboiler condenser unit (IRC). The nitrogen product has molar impurity fraction $1 - x_{\text{N}_2}$, and molar flow rate $\dot{n}_{\text{prod}}$. The control degrees of freedom are the feed air flow rate $\dot{n}_{\text{mac}}$, the stream split fraction to the primary turbine ξ_{tur}, the reflux fraction at the column top ξ_{reflux}, and the liquid drain from the reboiler $\dot{n}_{\text{drain}}$. The process is equipped with a stabilizing P-controller manipulating $\dot{n}_{\text{drain}}$ to counteract the integrating response of the liquid reboiler inventory M_{r} to the other inputs. We use a detailed dynamic process model, referred to as digital twin, taken from our previous work (Caspari et al., 2020). In particular, we formulate mass and energy balances for all process units, including tray-by-tray modeling of the column and finite volume discretization of PHX1 and PHX2. We complement these equations by thermodynamic correlations, including Margules activity model for equilibrium computations.

The digital twin (Caspari et al., 2020) is implemented in the modeling language Modelica and has 118 differential and 2675 algebraic equations. We assume that the model captures the process response sufficiently accurately, and we use it as a starting point for model reduction as well as a plant representative in closed-loop operation. The model is an index-one nonlinear semi-explicit differential-algebraic equation system (DAE), having a unique and smooth solution to resemble the behavior of Eq. (1b). All computations run on a terminal server with Intel XEON E5-2630 v2 CPU at 2.6 GHz and 128 GB RAM.

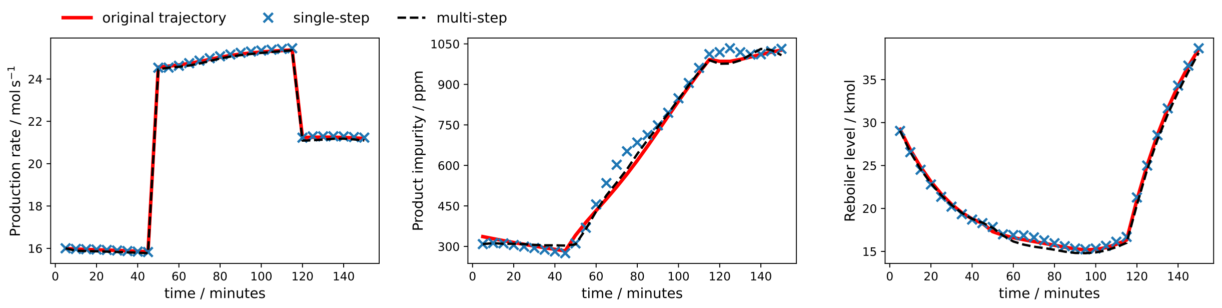

Figure 4: Independent open-loop step test of the reduced Koopman model.

Table 1: Parameters of data sampling.

Variable	Symbol	Value	Unit
Sampling time	Δt_s	5	min
No. diff. states	n_x	118	–
No. outputs	n_y	3	–
Dynamic traj. length	$t_{f,d}$	1000	h
Stationary traj. length	$t_{f,s}$	1000	h
Air feed rate	$\dot{n}_{\text{mac}}$	$[28, 52]$	mol s^{-1}
Turbine split	ξ_{tur}	$[0.88, 1.0]$	–
Reflux ratio	ξ_{reflux}	$[0.5, 0.55]$	–
Reboiler setpoint	$M_{\text{r,sp}}$	$[19, 41]$	kmol
P-controller gain	K_p	5	mol s^{-1} kmol^{-1}

Data sampling

The data sampling and model training procedure is similar to our previous work (Schulze et al., 2022). We obtain the training data set by simulating the full-order digital twin subject to a series of input steps comprising 800 random combinations of all control degrees of freedoms. For the purpose of data sampling, we retain the P-controller to ensure asymptotic stability and vary the inventory setpoint $M_{\text{r,sp}}$. However, we record the (biased) manipulation of $\dot{n}_{\text{drain}}$ to enable a Koopman model without P-controller, deciding against training a model of the process with base-layer control here. The duration of all input steps varies between 0.5 and 2 h (1000 h in total) to mix excitation of varying frequency. Moreover, we add 500 steady-state trajectories of 2 h length (1000 h in total), to ensure stability properties and small steady-state offset of the trained model.

Table 1 collects the information about the sampling. The input ranges are chosen slightly larger than the intended use in order to reduce boundary phenomena. For the simulation of the digital twin, we specify relative and absolute tolerances of 10^{-5} and 10^{-8}, respectively. We collect the snapshots as tuples of states, outputs, and controls. The three algebraic outputs $\boldsymbol{y}$ are the product impurity, production rate, and driving temperature difference in the IRC.

Model architecture and training

To identify suitable hyperparameters, we perform a systematic parameter study varying the autoencoder morphology (number of neurons and hidden layers) and the latent Koopman state dimension. Therein, we find that $n_z = 30$ and symmetric encoding and decoding with two hidden layers providing a constant relative decrease in the number of neurons, i.e., (76, 48) and (48, 76), respectively, are a suitable choice. Both encoder and decoder networks use tanh activation and linear output layers. Since the training data do not indicate oscillatory behavior, we preset a diagonal structure of A, confirmed by a training using full A. We use projection constraints to bound all elements of A to $a_{ii} \geq 0$.

We log-transform all molar fractions and scale all variables between zero and one. The trajectory data set is created by sliding along the recorded data in a moving horizon fashion, stopping every 5 sampling instants and copying $s = 24$ consecutive snapshots. We group random sets of 32 trajectories as mini-batches and divide the batched data into 80 % training and 20 % validation data. Model training is performed for 20 000 epochs using the optimizer Adam. After the training, we retrieve the weights with smallest validation loss.

Model testing

We perform an open-loop test of the reduced Koopman model in an independent test scenario in which the model is subject to two random steps of all inputs. Fig. 4 compares the open-loop prediction of the target quantities (production rate, product impurity, and reboiler hold-up) by the Koopman model to trajectories generated using the digital twin. The graphs show both a single multi-step simulation sweep over the entire horizon and a series of single-step predictions initialized along the reference trajectory. The single-step predictions are most important for short-term tracking and stabilization, whereas the multi-step prediction reflects the accuracy of the model to open-loop predict long-term trends. Despite the high degree of reduction, the Koopman model captures both short and long-term trends accurately. At high product impurities, we observe an opposite gain in a number of single-step predictions, which will be the subject of future improvements. However, this artifact did not affect the NMPC performance in the control study below.

Control case study

Since we target demand side management based on frequent load changes (Pattison et al., 2016), the control task is tracking of a series of instantaneous changes in the production rate, while maintaining product quality and stabilizing the liquid reboiler level M_{r}. We use the digital twin as plant representation in closed-loop operation and leave the effect of mismatch between digital twin and plant for future work. We formulate the NMPC tracking stage cost as:

$$\ell := w_1 \cdot (\dot{n}_{\text{prod}} - \dot{n}_{\text{prod,sp}})^2 + w_2 \cdot (M_{\text{r}} - M_{\text{r,sp}})^2 . \qquad (5)$$

Table 2 summarizes the NMPC tuning.

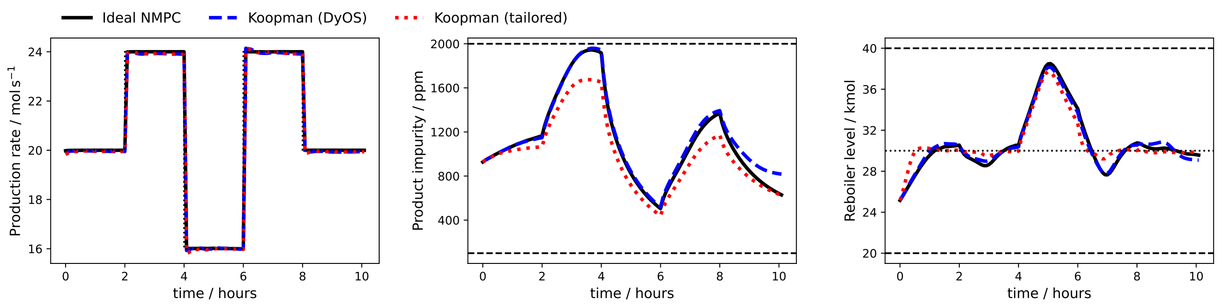

Figure 5: Closed-loop results (process response) of the controlled variables in the NMPC case study.

Table 2: NMPC tuning.

Variable	Symbol	Value	Unit
Sampling time	Δt_s	5	min
Control horizon	T_c	120	min
Tuning weight 1	w_1	1.0	$s^2\,mol^{-2}$
Tuning weight 2	w_2	0.0005	$kmol^{-2}$
Level setpoint (fixed)	$M_{r,sp}$	3.0	kmol
Level constraint	M_r	[20, 40]	kmol
Impurity constraint	$1 - x_{N_2}$	[100, 2000]	ppm
Air feed rate	$\dot{n}_{mac}$	[30, 50]	$mol\,s^{-1}$
Turbine split	ξ_{tur}	[0.9, 1.0]	–
Reflux ratio	ξ_{reflux}	[0.51, 0.54]	–
Reboiler drain	$\dot{n}_{drain}$	[0, 1.0]	$mol\,s^{-1}$
Integrator tolerances (*only DyOS*)		10^{-6}	–
Optimality tolerance		10^{-5}	–
Feasibility tolerance		10^{-3}	–

We scale all terms in Eq. (4) and warm-start all optimizations. The NMPC does not anticipate the setpoint changes, i.e., we expect a considerable CPU effort in re-optimization at setpoint updates. We apply full state feedback to exclude state estimation errors and focus on the model reduction.

We compare the tailored Koopman NMPC implementation to an ideal NMPC optimizing the full-order control model using a general-purpose DAE optimizer (DyOS). In addition, we compare to Koopman NMPC implemented with DyOS, i.e., an NMPC implementation similar to our previous work (Schulze and Mitsos, 2022). These two benchmarks enable us to evaluate two effects separately: 1) The model reduction, i.e., benefit from full-order vs. reduced Koopman using the same optimization platform, and 2) the benefit from a tailored optimization exploiting the specific problem structure. To enable 1), we transform the Koopman model to continuous-time form using MATLAB 2019a and translate it to the Modelica language. In all cases, we neglect the closed-loop effect of CPU delays on the controlled process.

Fig. 5 shows the closed-loop results of the case study, i.e., the process response to the control action. All controllers accomplish fast and precise setpoint tracking of the primary target (Fig. 5a) and feasible operation of the ASU (Figs. 5b+c), consistent with the excellent open-loop predictions by the reduced models (Fig. 4). While benchmark and Koopman NMPC using the same optimization platform (DyOS) yield almost indistinguishable closed-loop trajectories, the tailored Koopman implementation exhibits slight deviations from the other trajectories, most pronounced for the impurity. However, these deviations are limited and not associated with a noticeable tracking or feasibility loss. We attribute the deviations to numerical effects in the sensitivity computations and the respective local optimization algorithms.

Table 3: CPU effort reduction achieved by the proposed Koopman NMPC implementation.

NMPC	∅ CPU time	∅ Red.	Max. CPU time
Ideal	481 s	–	3358 s
Koopman (DyOS)	47 s	**90 %**	330 s
Koopman (tailored)	9 s	**98 %**	48 s

We compare the CPU effort of solving the NMPC implementations and assess the real-time capability in Table 3. The computational effort of ideal NMPC vastly exceeds the sampling time of 300 s and thereby introduces severe control delay. In contrast, employing the reduced Koopman model in the same NMPC framework generates a speed-up of factor 10, resulting in average CPU cost below the sampling time. However, the maximum CPU effort still lies beyond the sampling rate and the average computational delay is noticeable. Finally, the tailored NMPC facilitates most efficient optimization at an additional speed-up of over factor of 5, yielding an average CPU time decrease by 98 %. The CPU effort lies well below the sampling time and is of acceptable magnitude. Further, this speed-up is slightly improved compared to previous works applying classical model-based reduction methods to the same ASU (Schäfer et al., 2019; Caspari et al., 2020), where an average CPU reduction by 95 % was reported. Compared to these methods, our approach involves considerably less modeling effort, as the full-order process model is not modified in the reduction process and our deep learning framework is automated.

Conclusions

We apply Koopman theory for data-driven model reduction and real-time NMPC of a chemical process, specifically an ASU. The reduced model is trained using deep learning and consists of an autoencoder and linear latent dynamics, overall referred to as Wiener-type Koopman model. The data-driven nature of our approach greatly reduces the required process knowledge and enables an automated reduction procedure. In addition, we present an NMPC implementation tailored to the block structure of the reduced models.

Despite the high degree of reduction, the low-order models are accurate and enable precise control of the ASU at a reduction of CPU costs by 98 %. If required, a further speed-up could be achieved by applying a more efficient NLP solver, stronger reduction, or by increasing the tolerances at the cost of tracking performance. Additionally, employing decomposition strategies for further exploitation of the graph structure (Jalving et al., 2019) may enable faster optimization.

In this study, we did not consider state estimation to exclude the effect of estimation errors on closed-loop performance, thereby allowing an isolated study of the reduction approach. However, we have recently proposed a strategy to train state estimation into the data-driven Koopman model structure (Schulze and Mitsos, 2022). Future work will apply this integrated approach in extended studies. Moreover, we will compare against conventional controller types. Finally, we have assumed that the provided digital twin captures the real plant exactly. To account for plant-model mismatch, future work will investigate methods for online model improvement in closed-loop operation.

Acknowledgment

We gratefully acknowledge the financial support of the Kopernikus project SynErgie 2 by the Federal Ministry of Education and Research (BMBF) and the project supervision by the project management organization Projektträger Jülich, as well as from the Helmholtz Association of German Research Centers as part of the Helmholtz School for Data Science in Life, Earth and Energy (HDS-LEE).

References

Antoulas, A. C., S. Lefteriu, A. C. Ionita, P. Benner, and A. Cohen (2017). A tutorial introduction to the Loewner framework for model reduction. *Model Reduction and Approximation: Theory and Algorithms 15*, 335.

Brunton, S., B. W. Brunton, J. L. Proctor, and J. N. Kutz (2016). Koopman Invariant Subspaces and Finite Linear Representations of Nonlinear Dynamical Systems for Control. *PLOS ONE 11*(2), e0150171.

Caspari, A., A. Bremen, J. M. Faust, F. Jung, C. D. Kappatou, S. Sass, Y. Vaupel, R. Hannemann-Tamás, A. Mhamdi, and A. Mitsos (2019). DyOS – a framework for optimization of large-scale differential algebraic equation systems. In *Computer Aided Chemical Engineering*, Volume 46, pp. 619–624. Elsevier.

Caspari, A., C. Offermanns, A. M. Ecker, M. Pottmann, G. Zapp, A. Mhamdi, and A. Mitsos (2020). A Wave Propagation Approach for Reduced Dynamic Modeling of Distillation Columns: Optimization and Control. *Journal of Process Control 91*, 12–24.

Chen, Z., M. A. Henson, P. Belanger, and L. Megan (2010). Nonlinear model predictive control of high purity distillation columns for cryogenic air separation. *IEEE Transactions on Control Systems Technology 18*(4), 811–821.

Jalving, J., Y. Cao, and V. M. Zavala (2019). Graph-based modeling and simulation of complex systems. *Computers & Chemical Engineering 125*(1), 134–154.

Korda, M. and I. Mezić (2018). Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. *Automatica 93*(1), 149–160.

Lusch, B., J. N. Kutz, and S. Brunton (2018). Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications 9*(1), 1–10.

Marquardt, W. (2002). Nonlinear model reduction for optimization based control of transient chemical processes. *AIChE Symposium Series*.

Masti, D. and A. Bemporad (2021). Learning nonlinear state–space models using autoencoders. *Automatica 129*, 109666.

Mezić, I. (2005). Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dynamics 41*(1), 309–325.

Narasingam, A. and J. S.-I. Kwon (2019). Koopman Lyapunov–based model predictive control of nonlinear chemical process systems. *AIChE Journal 65*(11), e16743.

Pattison, R. C., C. R. Touretzky, T. Johansson, I. Harjunkoski, and M. Baldea (2016). Optimal process operations in fast-changing electricity markets: framework for scheduling with low-order dynamic models and an air separation application. *Industrial & Engineering Chemistry Research 55*(16), 4562–4584.

Proctor, J. L., S. L. Brunton, and J. N. Kutz (2016). Dynamic mode decomposition with control. *SIAM Journal on Applied Dynamical Systems 15*(1), 142–161.

Qian, E., B. Kramer, B. Peherstorfer, and K. Willcox (2020). Lift & Learn: Physics-informed machine learning for large-scale nonlinear dynamical systems. *Physica D: Nonlinear Phenomena 406*, 132401.

Schäfer, P., A. Caspari, A. Mhamdi, and A. Mitsos (2019). Economic nonlinear model predictive control using hybrid mechanistic data-driven models for optimal operation in real-time electricity markets: In-silico application to air separation processes. *Journal of Process Control 84*, 171–181.

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

Schulze, J. C., A. Caspari, C. Offermanns, A. Mhamdi, and A. Mitsos (2021). Nonlinear model predictive control of ultra-high-purity air separation units using transient wave propagation model. *Computers & Chemical Engineering 145*, 107163.

Schulze, J. C., D. T. Doncevic, and A. Mitsos (2022). Identification of MIMO Wiener-type Koopman models for data-driven model reduction using deep learning. *Computers & Chemical Engineering 161*(15), 107781.

Schulze, J. C. and A. Mitsos (2022). Data-driven Nonlinear Model Reduction using Koopman Theory: Integrated Control Form and NMPC Case Study. *IEEE Control Systems Letters (L-CSS)*.

Surana, A. (2016). Koopman operator based observer synthesis for control-affine nonlinear systems. *2016 IEEE 55th Conference on Decision and Control (CDC)*, 6492–6499.

Wächter, A. and L. T. Biegler (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming 106*(1), 25–57.

Watter, M., J. Springenberg, J. Boedecker, and M. Riedmiller (2015). Embed to control: A locally linear latent dynamics model for control from raw images. *Advances in Neural Information Processing Systems 28*.

Williams, M. O., C. W. Rowley, and I. G. Kevrekidis (2015). A Kernel-Based Approach to Data-Driven Koopman Spectral Analysis. *Journal of Computational Dynamics 2*(2), 247.