

# End-to-End Reinforcement Learning of Koopman Models for eNMPC of an Air Separation Unit

Daniel Mayfrank<sup>a,d</sup>, Kayra Dernek<sup>a,b</sup>, Laura Lang<sup>b</sup>, Alexander Mitsos<sup>c,a,b</sup>, Manuel Dahmen<sup>a,\*</sup>

<sup>a</sup> Forschungszentrum Jülich GmbH, Institute of Climate and Energy Systems, Energy Systems Engineering (ICE-1), Jülich 52425, Germany

<sup>b</sup> RWTH Aachen University, Process Systems Engineering (AVT.SVT), Aachen 52074, Germany

<sup>c</sup> JARA-ENERGY, Jülich 52425, Germany

<sup>d</sup> RWTH Aachen University, Aachen 52062, Germany

**Abstract** – With our recently proposed method based on reinforcement learning (Mayfrank et al. (2024), *Comput. Chem. Eng.* 190), Koopman surrogate models can be trained for optimal performance in specific (economic) nonlinear model predictive control ((e)NMPC) applications. So far, our method has exclusively been demonstrated on a small-scale case study. Herein, we show that our method scales well to a more challenging demand response case study built on a large-scale model of a single-product (nitrogen) air separation unit. Across all numerical experiments, we assume observability of only a few realistically measurable plant variables. Compared to a purely system identification-based Koopman eNMPC, which generates small economic savings but frequently violates constraints, our method delivers similar economic performance while avoiding constraint violations.

**Keywords:** Economic model predictive control; Koopman; Demand response; Air separation unit; Reinforcement learning

## 1 Introduction

Data-driven dynamic models can be trained in an end-to-end fashion for optimal performance as part of (economic) (nonlinear) model predictive control ((e)(N)MPC) (e.g., Gros and Zanon (2019); Amos et al. (2018)). We recently introduced a method (Mayfrank et al. (2024)) based on reinforcement learning (RL) for end-to-end learning of Koopman *surrogate* models (Korda and Mezić (2018)) for (e)NMPC applications. Such data-driven surrogate models can make eNMPC computationally tractable in case an accurate mechanistic process model is available but using it as part of a model predictive controller is too com-

putationally expensive. Moreover, in scenarios where no reliable model is available, the same framework can learn directly from plant data. Alternative methods for end-to-end learning of data-driven models for control applications focus on linear models (Chen et al. (2019)), optimize highly-structured models with few parameters requiring expert system knowledge (Brandner et al. (2023)), cannot handle hard constraints on system states (Amos et al. (2018)), or are only applicable to setpoint tracking problems (Iwata and Kawahara (2022)). Our method can optimize highly parameterized and thus flexible Koopman models for control problems with state constraints and arbitrary convex objective functions.

\*Manuel Dahmen, Forschungszentrum Jülich GmbH, Institute of Climate and Energy Systems, Energy Systems Engineering (ICE-1), Jülich 52425, Germany  
E-mail: m.dahmen@fz-juelich.de

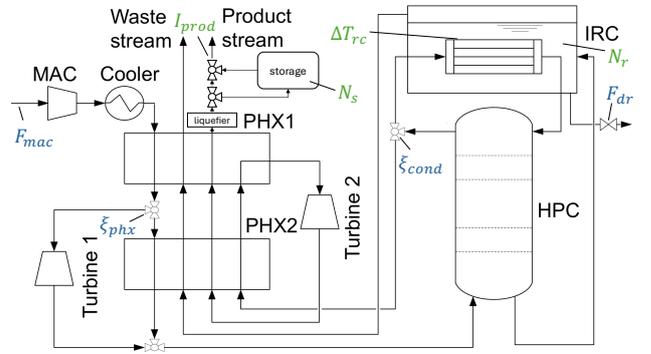
In [Mayfrank et al. \(2024\)](#), we demonstrated our method in two simulated case studies (NMPC and eNMPC) based on a small model of a continuous stirred-tank reactor (CSTR) ([Flores-Tlacuahuac and Grossmann \(2006\)](#)) comprised of just two ordinary differential equations. The resulting policies outperformed Koopman controllers employing models that were trained using the prevailing system identification (SI) approach by (i) achieving more accurate state tracking in the NMPC case study and (ii) substantially reducing the frequency of constraint violations in the eNMPC case study. In the present contribution, we demonstrate the scalability of our method ([Mayfrank et al. \(2024\)](#)) using a large-scale differential-algebraic equations (DAE) model of an air separation unit (ASU) ([Caspari et al. \(2020\)](#)).

The remainder of this short paper is organized as follows: First, the ASU demand response case study is introduced in [Sec. 2](#). Then, [Sec. 3](#) provides a brief explanation of our method, followed by a description of the adjustments to the Koopman model architecture we employed in this work. [Sec. 4](#) presents the results of the numerical experiments. Finally, [Sec. 5](#) discusses the conclusions and directions for future work.

## 2 Demand response of an air separation unit

We consider demand response of a single-product ASU for the production of purified nitrogen based on the benchmark process presented in [Caspari et al. \(2020\)](#), resulting in a mid-level complexity control problem. The process flowsheet is shown in [Figure 1](#). Because RL approaches often need many policy-environment interactions to produce good results, wall-clock simulation speed is criti-

cal for simulation models that are to be used as part of the training environment. Because the full mechanistic model [Caspari et al. \(2020\)](#) is computationally expensive, we construct the demand-response RL environment using the modified model of [Schulze et al. \(2023\)](#), which enables substantially faster simulation. The modified model is a nonlinear DAE system with 2327 algebraic and 119 differential states, implemented in Modelica and still computationally expensive, thus motivating the proposed end-to-end learning.



**Fig. 1.** Air separation process flowsheet. The following manipulated variables are shown in blue font: inlet air flow rate  $F_{mac}$ , air fraction passing through turbine 1  $\xi_{phx}$ , distillation column reflux ratio  $\xi_{cond}$ , drain stream  $F_{dr}$ . The controlled variables are depicted in green font: product impurity  $I_{prod}$ , molar holdup in storage  $N_s$  and reboiler  $N_r$ , temperature difference between reboiler and condenser  $\Delta T_{rc}$ .

Ambient air is compressed in the main air compressor (MAC), pre-cooled, and then passes through a two-part multi-stream heat exchanger (MSHE), where it is cooled against returning process streams. After the first part of the MSHE (PHX1), a fraction of the air is used in turbine 1 for power generation, while the remainder is liquefied in the second part of the MSHE (PHX2). Both streams are recombined before entering the high-pressure distillation column (HPC). The oxygen-rich bottom product of the HPC is expanded and used in an integrated reboiler condenser (IRC) to cool the reflux stream. Liquid is with-

drawn via a drain stream, and the exiting vapor leaves the process as waste after heat recovery in the MSHE. The nitrogen-rich top product passes through turbine 2 and a liquefier to yield liquid nitrogen, which can be stored in a product tank for flexible delivery.

The task of the eNMPC is to minimize operational cost by exploiting variations in the electricity price, while fulfilling a constant demand for liquid nitrogen and avoiding constraint violations. The operational cost is given by the overall power consumption  $E$  of the ASU multiplied by the electricity price. For the overall power consumption, the energy demand of the MAC and the liquefier, as well as the electricity generation from the turbines, are taken into account. Operational constraints and manipulated control inputs are shown in the ASU flowsheet in Fig. 1, and their respective lower and upper bounds are given in Table 1.

Variable	lb	ub	Constraint type
$I_{\text{prod}}$ [ppm]	0	1800	path
$\Delta T_{\text{rc}}$ [K]	2	5	path
$N_r$ [kmol]	2	10	path
$N_s$ [-]	0	6	path
$F_{\text{mac}}$ [mol/s]	30	50	input
$F_{\text{dr}}$ [mol/s]	0	2	input
$\xi_{\text{phx}}$ [kmol]	0	0.1	input
$\xi_{\text{cond}}$ [-]	0.51	0.54	input

**Tab. 1.** Summary of lower (lb) and upper (ub) bounds of the operational and input variables.

To use the Modelica model as part of an RL environment, we export it as a functional mock-up unit that can be simulated within Python code. At each control step in the environment, the policy receives the current state of the ASU and an electricity price prediction for the upcoming 9 hours in hourly resolution. After receiving a control input from the policy, the state of the ASU is updated by simulating the model for a time step of 15 min. Furthermore, analogous to the reward calculation in Mayfrank

et al. (2024), we calculate a reward based on constraint violations and electricity cost savings compared to steady-state production.

### 3 End-to-end RL of Koopman Model for eNMPC

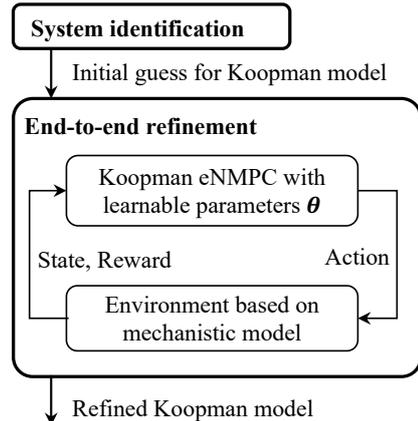
In Mayfrank et al. (2024), we utilize Koopman models of the form proposed by Korda and Mezić (2018): (i) A nonlinear state observation function  $\psi_{\theta}: \mathbb{R}^{n_x} \mapsto \mathbb{R}^{n_z}$  that transforms the initial system state  $\mathbf{x}_0 \in \mathbb{R}^{n_x}$  into the initial Koopman state  $\mathbf{z}_0 \in \mathbb{R}^{n_z}$ , where typically  $n_z \gg n_x$ :  $\mathbf{z}_0 = \psi_{\theta}(\mathbf{x}_0)$ . (ii) The  $\mathbf{A}_{\theta} \in \mathbb{R}^{n_z \times n_z}$  and  $\mathbf{B}_{\theta} \in \mathbb{R}^{n_z \times n_u}$  matrices, which linearly approximate the evolution of the Koopman state, driven by external control inputs  $\mathbf{u}_t \in \mathbb{R}^{n_u}$ :  $\mathbf{z}_{t+1} = \mathbf{A}_{\theta}\mathbf{z}_t + \mathbf{B}_{\theta}\mathbf{u}_t$ . (iii) The  $\mathbf{C}_{\theta} \in \mathbb{R}^{n_x \times n_z}$  matrix, which linearly transforms the Koopman state  $\mathbf{z}_t$  into a predicted system state  $\hat{\mathbf{x}}_t$ :  $\hat{\mathbf{x}}_t = \mathbf{C}_{\theta}\mathbf{z}_t$ . Such models can be trained by adjusting the parameters  $\theta$ .

RL algorithms (e.g., Schulman et al. (2017)) can be used to optimize parameterized policies  $\pi_{\theta}(\mathbf{u}_t|\mathbf{x}_t): \mathbb{R}^{n_x} \mapsto \mathbb{R}^{n_u}$ , which map current system states  $\mathbf{x}_t$  to control actions  $\mathbf{u}_t$ , by maximizing the expected cumulative reward of the policy. Therefore, to train a Koopman model in an end-to-end manner for a particular eNMPC application via RL, we construct an (automatically differentiable) eNMPC policy based on the Koopman model. When used as part of an eNMPC policy, Koopman models of the above stated form (Korda and Mezić (2018)) result in convex optimal control problems (OCPs) if the stage cost and all additional user-defined constraints are convex. By using the automatically differentiable solver *cvxpylayers* (Agrawal et al. (2019)) for convex optimization problems, we can

obtain  $\partial \mathbf{u}_t^* / \partial \boldsymbol{\theta}$  via implicit differentiation of the KKT conditions. The solution map  $\boldsymbol{\theta} \mapsto \mathbf{u}_t^*$  is only piecewise differentiable; nevertheless our numerical experience shows that the values returned from *cvxpylayers* as derivatives work well for the first-order training.

In our previous work (Mayfrank et al., 2024), the path constraints for the controlled variables were formulated as inequality constraints. However, when applied to the ASU model, the end-to-end learning approach showed no improvement over the initial guess. We attribute this to poor gradient estimation caused by switching between active and inactive inequality constraints. To address this issue, we reformulate the OCP in the end-to-end RL by replacing the inequality constraints with equality constraints with slack variables  $\mathbf{s}$ , and penalize the slack variables through quadratically smoothed hinge loss (Zhang (2004)), defined as  $L(s_i, \delta) = M [\max(0, |s_i| - \frac{1}{2} \Delta g_i + \delta)]^2$ , where  $\Delta g_i$  represents the admissible range of the corresponding constraint (see supplementary material). The variable  $M > 0$  is a penalty coefficient to balance the penalty and the objective performance. To more strongly penalize constraint violations, we introduce penalty scaling values  $\delta$  that slightly tighten the bounds of the inequality constraints, leading to a more conservative control behavior.

Our workflow is shown in Fig. 2. Initial values for  $\boldsymbol{\theta}$  are obtained through standard system identification (SI), i.e., we generate simulation data using the mechanistic model and fit  $\boldsymbol{\theta}$  to that data. A more detailed description of the SI is provided in the supplementary materials. The model parameters  $\boldsymbol{\theta}$  are then fine-tuned for optimal performance in a specific control task using Proximal Policy Optimization (PPO) (Schulman et al., 2017), an actor-critic RL algorithm.



**Fig. 2.** Workflow for end-to-end learning of a Koopman model for eNMPC.

The ASU model is a DAE system that features system outputs  $\mathbf{y}_t$  that exhibit discontinuities when control inputs change non-continuously. The Koopman model architecture proposed by Korda and Mezić (2018), which we used in Mayfrank et al. (2024), cannot reflect such behavior since the control inputs enter directly into the predictor-part of the model, which produces the latent space vector one time step into the future. To enable the Koopman models to represent instantaneous output responses to input changes, we extend the framework of Korda and Mezić (2018) by adding a second decoder, i.e.,  $\mathbf{y}_t = \mathbf{D}_\theta \mathbf{z}_t + \mathbf{E}_\theta \mathbf{u}_t$ , with learnable  $\mathbf{D}_\theta \in \mathbb{R}^{n_y \times n_z}$  and  $\mathbf{E}_\theta \in \mathbb{R}^{n_y \times n_u}$  matrices. Note that this second decoder was not needed in our previous work (Mayfrank et al., 2024) because the CSTR investigated there was described by an ordinary differential equation (ODE) system for which discontinuities in control inputs do not cause discontinuities in states/outputs. The overall architecture of the Koopman model is visualized in Figure 3.

We do not assume full observability of the ASU. Instead, we use only the following states as inputs to the Koopman surrogate model: the product purity  $I_{\text{prod}}$ , the temperature difference between reboiler and condenser

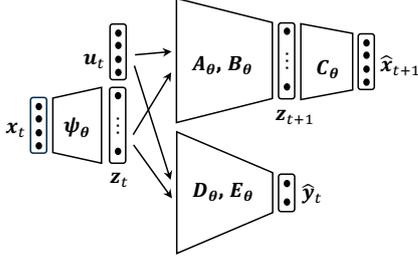


Fig. 3. Koopman model architecture.

$\Delta T_{rc}$ , the reboiler holdup  $N_r$ , and the temperature of tray 20 in the distillation column  $T_{tray,20}$ . Based on this information, the model constructs a latent representation  $z_t$  of the state of the ASU. The vector of control inputs  $u_t$  consists of the four inputs listed in Table 1. The model predicts the evolution of the control-relevant entries of  $x_t$  through the  $A_\theta$ ,  $B_\theta$ , and  $C_\theta$  matrices. The resulting vector  $\hat{x}_t \in \mathbb{R}^3$  consists of  $I_{prod}$ ,  $\Delta T_{rc}$ , and  $N_r$ . The control-relevant, discontinuous outputs  $y_t \in \mathbb{R}^2$  are the energy demand  $E$  of the process and the production rate  $\dot{n}_{product}$ . The latter is used to calculate the molar holdup  $N_s$  of the storage tank through a mass balance, where the change in tank storage is given by the difference between the inflow and outflow rates, i.e.,  $\frac{dN_s}{dt} = \dot{n}_{product} - \dot{n}_{demand}$ . The molar holdup  $N_s$  is upper and lower bounded to reflect the physical storage capacity limits of the tank.

We choose a latent space dimensionality of 10 for the Koopman model, i.e.,  $z_t \in \mathbb{R}^{10}$ . Altogether, the Koopman model thus consists of the matrices  $A_\theta \in \mathbb{R}^{10 \times 10}$ ,  $B_\theta \in \mathbb{R}^{10 \times 4}$ ,  $C_\theta \in \mathbb{R}^{3 \times 10}$ ,  $D_\theta \in \mathbb{R}^{2 \times 10}$ ,  $E_\theta \in \mathbb{R}^{2 \times 4}$ , and an encoder  $\psi_\theta: \mathbb{R}^4 \rightarrow \mathbb{R}^{10}$ . The encoder is a feedforward neural network with two hidden layers with 50 neurons each, and tanh activation functions.

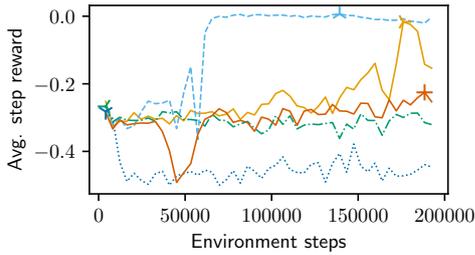
## 4 Numerical experiments

All training code used in this work, including the implementation of the ASU demand response RL environment, is available online<sup>1</sup>.

We obtain an initial guess for the Koopman model via the iterative data sampling and system identification approach outlined in the supporting information. Then, we repeat the end-to-end refinement five times using different fixed seeds in every training run. We use the historic German day-ahead electricity prices from 2023-01-01 to 2023-12-31 (EPEX (2023)) for training.

Fig. 4 illustrates the evolution of the reward over 200,000 environment steps for each training run. Each training run takes approx. two days on a Windows 11 workstation with an Intel Core i7-14700 CPU. Three elements contribute to overall training time: (i) simulation of the mechanistic DAE environment, (ii) repeated solution of the convex eNMPC problem, and (iii) gradient-based policy optimization. The dominant contributor to the training time is the policy optimization step, which requires backpropagating through the Koopman eNMPC. In contrast, pure policy inference typically takes well under one second per control step, making the controller easily real-time capable. After training has completed, our final performance evaluation is based on the policy that attained the highest average reward during a policy rollout. Consequently, our primary metric of interest is the maximum average reward achieved by each training run, rather than the reward at the final training step. Three out of five training runs demonstrate a substantial improvement in the policy's average reward compared to the initial guess. After training, we test the performance

<sup>1</sup><https://jugit.fz-juelich.de/iek-10/public/optimization/koopmanenmpc4asu>



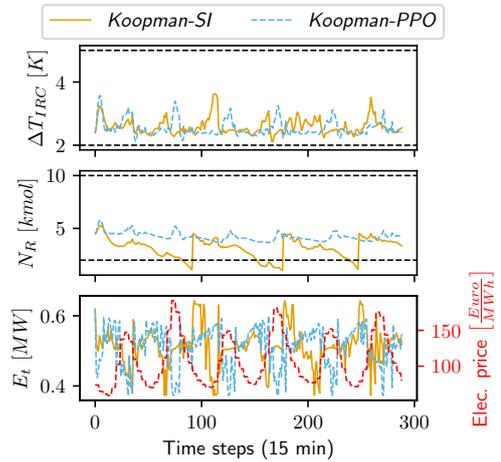
**Fig. 4.** Learning progress during end-to-end model refinement. Each line represents one training run. For each training run, a marker depicts the highest average reward achieved in one policy rollout.

of the policy that obtained the highest average reward in a three-day test episode. The test electricity price trajectory is generated using the method by Papadimitriou et al. (2024), which constructs representative price profiles from 2023 historical data while preserving key statistics (mean and variance). The resulting profiles are distinct from those used during training, ensuring a clear separation between training and testing data.

In the test episode, the eNMPC employing the Koopman model obtained solely via system identification (from hereon called *Koopman-SI* policy) saves 1% of electricity cost compared to steady-state production while producing small constraint violations in 16.3% of the time steps, resulting in an average reward of  $-0.26$ . The eNMPC employing the end-to-end refined Koopman model (*Koopman-PPO*) improves performance, and produces 2% cost savings. Furthermore, it does so without violating any constraints in the test episode, thus yielding an average reward of 0.01.

Figure 5 illustrates the behavior of the policies in the three-day test episode. We show the evolution of  $\Delta T_{IRC}$  and  $N_R$ , since these are the only states that operate close to their bounds, as well as the evolution of the energy demand  $E$ . Both policies exhibit an intuitive inverse relationship between the electricity price and the energy

demand. It can be seen that the behavior of the policies differs for states  $\Delta T_{IRC}$  and  $N_R$ . While *Koopman-SI* exhibits violations in  $N_R$ , *Koopman-PPO* maintains operation safely within the operation bounds. Both policies exhibit high-frequency oscillations with respect to  $E_t$ .



**Fig. 5.** Comparison of the control behavior of *Koopman-SI* and *Koopman-PPO*.

As stated above, the architecture of the Koopman models that we use (see Figure 3) results in convex OCPs, which are relatively easy to solve. In the 72 hour test episode, i.e., 288 control steps of 15 min length each, the inference time of the *Koopman-PPO* policy was between 0.1 s and 3 s with an average time of 0.5 s.

## 5 Conclusion

We apply our previously published method (Mayfrank et al. (2024)) for end-to-end learning of Koopman surrogate models for (e)NMPC applications to a high-dimensional nonlinear demand response problem based on a mechanistic model of an ASU, demonstrating its applicability to complex, real-world systems.

Among the five independent end-to-end training runs, three significantly improve eNMPC performance over the

SI baseline, reducing and even eliminating constraint violations while maintaining high economic efficiency. Compared to our earlier two-state CSTR case study (Mayfrank et al. (2024)), where all ten training runs consistently improved performance and rewards saturated after roughly 180,000 steps, the present large-scale ASU problem poses a more challenging learning task, reflected in the lower fraction of successful runs. Note that while our method strongly incentivizes constraint satisfaction, it does *not* provide formal theoretical guarantees.

Our method produces data-driven predictive control policies that strike an exceptional balance between control performance (high economic performance and consistent constraint satisfaction) and computational efficiency (policy inference time  $\ll$  sampling time). By demonstrating its scalability to an industrially-relevant process, we show that our method could be applicable to complex real-world control problems where mechanistic predictive control policies are not real-time capable and system identification-based data-driven controllers produce unsatisfactory performance.

Future work should test our method on a real-world process. Because the Koopman surrogate model is trained in an environment based on a mechanistic simulation model, it may overfit to systematic simulation errors, e.g., parameter mismatch or unmodeled dynamics. The resulting controller could perform suboptimally on the real process. This risk does not exist in the present work, since training and final evaluation use the same simulated environment. In the described real-world scenario, our method may be used to further refine the surrogate model via interactions with the real process. In such an approach, the sample efficiency of the learning procedure would become critical, which could motivate employing

a model-based RL algorithm as done in Mayfrank et al. (2025).

## Declaration of Competing Interest

We have no conflict of interest.

## Acknowledgements

This work was performed as part of the Helmholtz School for Data Science in Life, Earth and Energy (HDS-LEE) and received funding from the Helmholtz Association of German Research Centres. Additionally, the authors gratefully acknowledge the financial support of the Kopernikus project SynErgie by the Federal Ministry of Education and Research (BMBF), and the project supervision by the project management organization Projektträger Jülich (PtJ).

## Author contributions

Daniel Mayfrank: Conceptualization, Methodology, Software, Investigation, Writing - original draft, Writing - review & editing, Visualization

Kayra Dernek: Methodology, Software, Investigation, Writing - review & editing, Visualization

Laura Lang: Methodology, Writing - review & editing, Writing - original draft

Alexander Mitsos: Conceptualization, Writing - review & editing, Supervision, Funding acquisition

Manuel Dahmen: Conceptualization, Methodology, Writing - review & editing, Supervision, Funding acquisition

## Bibliography

- Agrawal, A., Amos, B., Barratt, S., Boyd, S., Diamond, S., and Kolter, J. Z. (2019). Differentiable convex optimization layers. *Advances in Neural Information Processing Systems*, 32:9558–9570.
- Amos, B., Jimenez, I., Sacks, J., Boots, B., and Kolter, J. Z. (2018). Differentiable MPC for end-to-end planning and control. *Advances in Neural Information Processing Systems*, 31:8299–8310.
- Brandner, D., Talis, T., Esche, E., Repke, J.-U., and Lucia, S. (2023). Reinforcement learning combined with model predictive control to optimally operate a flash separation unit. In *Computer Aided Chemical Engineering*, volume 52, pages 595–600. Elsevier.
- Caspari, A., Tsay, C., Mhamdi, A., Baldea, M., and Mitsos, A. (2020). The integration of scheduling and control: Top-down vs. bottom-up. *Journal of Process Control*, 91:50–62.
- Chen, B., Cai, Z., and Bergés, M. (2019). Gnu-RL: A precocial reinforcement learning solution for building HVAC control using a differentiable MPC policy. In *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, pages 316–325.
- EPEX (2023). Epex spot market data. <https://www.epexspot.com>. Accessed: 2025-01-02.
- Flores-Tlacuahuac, A. and Grossmann, I. E. (2006). Simultaneous cyclic scheduling and control of a multi-product CSTR. *Industrial & Engineering Chemistry Research*, 45(20):6698–6712.
- Gros, S. and Zanon, M. (2019). Data-driven economic NMPC using reinforcement learning. *IEEE Transactions on Automatic Control*, 65(2):636–648.
- Iwata, T. and Kawahara, Y. (2022). Data-driven end-to-end learning of pole placement control for nonlinear dynamics via Koopman invariant subspaces. *arXiv preprint arXiv:2208.08883*.
- Korda, M. and Mezić, I. (2018). Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. *Automatica*, 93:149–160.
- Mayfrank, D., Mitsos, A., and Dahmen, M. (2024). End-to-end reinforcement learning of Koopman models for economic nonlinear model predictive control. *Computers & Chemical Engineering*, 190:108824.
- Mayfrank, D., Velioglu, M., Mitsos, A., and Dahmen, M. (2025). Sample-efficient reinforcement learning of Koopman eNMPC. *arXiv preprint arXiv:2503.18787*.
- Papadimitriou, C., Schulze, J. C., and Mitsos, A. (2024). Representative electricity price profiles for european day-ahead and intraday spot markets. *arXiv preprint arXiv:2405.14403*.
- Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.
- Schulze, J. C., Doncevic, D. T., Erwes, N., and Mitsos, A. (2023). Data-driven model reduction and nonlinear model predictive control of an air separation unit by applied Koopman theory. *arXiv preprint arXiv:2309.05386*.
- Zhang, T. (2004). Solving large scale linear prediction problems using stochastic gradient descent algorithms. *Twenty-first international conference on Machine learning - ICML '04*, page 116.

# Supplementary Material: End-to-End Reinforcement Learning of Koopman Models for eNMPC of an Air Separation Unit

Daniel Mayfrank<sup>a,d</sup>, Kayra Dernek<sup>a,b</sup>, Laura Lang<sup>b</sup>, Alexander Mitsos<sup>c,a,b</sup>, Manuel Dahmen<sup>a,\*</sup>

<sup>a</sup> Forschungszentrum Jülich GmbH, Institute of Climate and Energy Systems, Energy Systems Engineering (ICE-1), Jülich 52425, Germany

<sup>b</sup> RWTH Aachen University, Process Systems Engineering (AVT.SVT), Aachen 52074, Germany

<sup>c</sup> JARA-ENERGY, Jülich 52425, Germany

<sup>d</sup> RWTH Aachen University, Aachen 52062, Germany

**Keywords:** Economic model predictive control; Koopman; Demand response; Air separation unit

## 1 System identification procedure

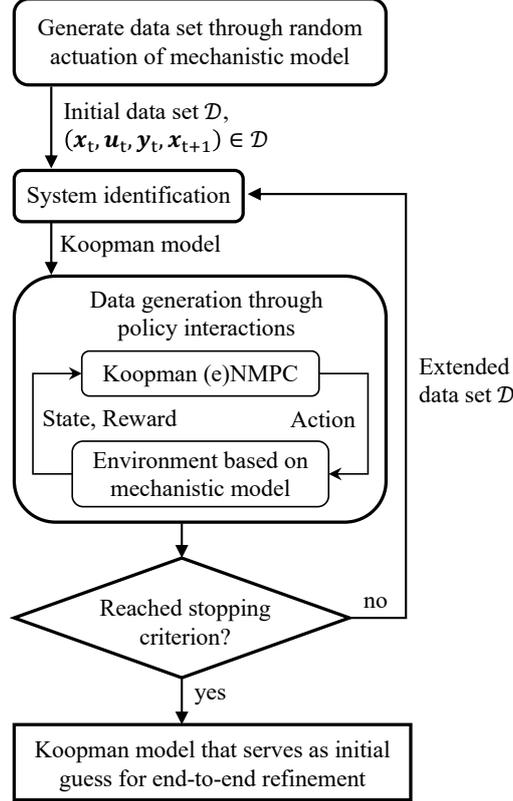
In this work, we employ the iterative data sampling and system identification approach depicted in Figure 1: We start by generating data through random actuation of the mechanistic model, followed by fitting the parameters  $\theta$  of the Koopman model to the obtained data. Then, we construct an eNMPC policy based on the Koopman model and let the policy interact with the environment for 2880 time steps, i.e., 30 simulated days, to extend the training data set, and we retrain the Koopman model on this larger data set. We keep extending the data set and retraining our Koopman model with the extended data set until the maximum average reward obtained during data sampling in one iteration does not improve for five consecutive iterations. The model of the policy that produced the maximum average reward is then used as the initial guess for the RL-based end-to-end learning procedure. At this stage, we do not employ the inequality constraints reformulation presented later; instead, we follow the same optimal control problem (OCP) formulation as in our previous work (Mayfrank et al., 2024).

## 2 Chaining of model predictions

RL-based end-to-end refinement of the Koopman model requires solving and backpropagating through the OCPs numerous times. To decrease the associated computational burden, it makes sense to minimize the number of optimization variables in the OCP, i.e., to maximize the time step duration  $\Delta t$  of the Koopman model and the resulting eNMPC controller. We observe that a discretization of the eNMPC controller using 15 min time steps is short enough to enable the controller to make use of the full feasible space of control inputs without causing an excessive number of constraint violations due to too fast process dynamics. However, during preliminary testing of system identification, we observe that we obtain more accurate model predictions if we chain the predictions of models that predict shorter time steps. Therefore, we take the following approach: During system identification,

---

\*Manuel Dahmen, Forschungszentrum Jülich GmbH, Institute of Climate and Energy Systems, Energy Systems Engineering (ICE-1), Jülich 52425, Germany  
E-mail: m.dahmen@fz-juelich.de



**Fig. 1.** Iterative data sampling and system identification procedure.

1 we generate data with a 5 min discretization and we fit a Koopman model to that data. However, before using the  
 2 model as part of eNMPC (in data sampling or end-to-end refinement), we transform the prediction components of  
 3 the Koopman model ( $\mathbf{A}, \mathbf{B}, \mathbf{D}, \mathbf{E}$ ) to predict 15 min time steps instead of 5 min steps, i.e., a one-step prediction of  
 4 the transformed model is equivalent to chaining three predictions of the original model with constant control input.  
 5 The following equations are used for this exact transformation:

$$\mathbf{A}_{15\text{min}} = \mathbf{A}^3, \quad (1a)$$

$$\mathbf{B}_{15\text{min}} = \mathbf{A}^2\mathbf{B} + \mathbf{A}\mathbf{B} + \mathbf{B}, \quad (1b)$$

$$\mathbf{D}_{15\text{min}} = \mathbf{D}\mathbf{A}^3, \quad (1c)$$

$$\mathbf{E}_{15\text{min}} = \mathbf{D}\mathbf{A}^2\mathbf{B} + \mathbf{D}\mathbf{A}\mathbf{B} + \mathbf{D}\mathbf{B} + \mathbf{E}. \quad (1d)$$

6 Eqs. 1 are derived in Eqs. 2 and Eqs. 3. Please note that the eNMPC has control steps of 15 min length, and  
 7 therefore,  $\mathbf{u}_t = \mathbf{u}_{t+1} = \mathbf{u}_{t+2}$  in the 5 min discretization Koopman model. To perform the upscaling of the Koopman  
 8 model from a 5 min time step to a 15 min time step, we derive the transformed system dynamics by chaining the  
 9 predictions of the original model over three consecutive time steps with constant input. The following equations  
 10 describe the stepwise progression for the latent state  $\mathbf{z}_t$  and output  $\mathbf{y}_t$  variables over three time steps, with constant

1 control input  $\mathbf{u}_t$  over the interval. This allows us to predict the system's behavior at the 15-minute time scale while  
 2 maintaining consistency with the original model dynamics.

$$\mathbf{z}_{t+1} = \mathbf{A}\mathbf{z}_t + \mathbf{B}\mathbf{u}_t \quad (2a)$$

$$\mathbf{z}_{t+2} = \mathbf{A}(\mathbf{A}\mathbf{z}_t + \mathbf{B}\mathbf{u}_t) + \mathbf{B}\mathbf{u}_t = \mathbf{A}^2\mathbf{z}_t + \mathbf{A}\mathbf{B}\mathbf{u}_t + \mathbf{B}\mathbf{u}_t \quad (2b)$$

$$\mathbf{z}_{t+3} = \mathbf{A}(\mathbf{A}^2\mathbf{z}_t + \mathbf{A}\mathbf{B}\mathbf{u}_t + \mathbf{B}\mathbf{u}_t) + \mathbf{B}\mathbf{u}_t = \mathbf{A}^3\mathbf{z}_t + \mathbf{A}^2\mathbf{B}\mathbf{u}_t + \mathbf{A}\mathbf{B}\mathbf{u}_t + \mathbf{B}\mathbf{u}_t \quad (2c)$$

$$\begin{aligned} &= \mathbf{A}^3\mathbf{z}_t + (\mathbf{A}^2\mathbf{B} + \mathbf{A}\mathbf{B} + \mathbf{B})\mathbf{u}_t = \mathbf{A}_{15\min}\mathbf{z}_t + \mathbf{B}_{15\min}\mathbf{u}_t \\ &\Rightarrow \mathbf{A}_{15\min} = \mathbf{A}^3, \mathbf{B}_{15\min} = \mathbf{A}^2\mathbf{B} + \mathbf{A}\mathbf{B} + \mathbf{B} \end{aligned} \quad (2d)$$

$$\mathbf{y}_{t+1} = \mathbf{D}\mathbf{z}_{t+1} + \mathbf{E}\mathbf{u}_t \quad (3a)$$

$$\mathbf{y}_{t+2} = \mathbf{D}\mathbf{z}_{t+2} + \mathbf{E}\mathbf{u}_t \quad (3b)$$

$$\mathbf{y}_{t+3} = \mathbf{D}\mathbf{z}_{t+3} + \mathbf{E}\mathbf{u}_t = \mathbf{D}(\mathbf{A}^3\mathbf{z}_t + \mathbf{A}^2\mathbf{B}\mathbf{u}_t + \mathbf{A}\mathbf{B}\mathbf{u}_t + \mathbf{B}\mathbf{u}_t) + \mathbf{E}\mathbf{u}_t \quad (3c)$$

$$\begin{aligned} &= \mathbf{D}\mathbf{A}^3\mathbf{z}_t + (\mathbf{D}\mathbf{A}^2\mathbf{B} + \mathbf{D}\mathbf{A}\mathbf{B} + \mathbf{D}\mathbf{B} + \mathbf{E})\mathbf{u}_t = \mathbf{D}_{15\min}\mathbf{z}_t + \mathbf{E}_{15\min}\mathbf{u}_t \\ &\Rightarrow \mathbf{D}_{15\min} = \mathbf{D}\mathbf{A}^3, \mathbf{E}_{15\min} = \mathbf{D}\mathbf{A}^2\mathbf{B} + \mathbf{D}\mathbf{A}\mathbf{B} + \mathbf{D}\mathbf{B} + \mathbf{E} \end{aligned} \quad (3d)$$

### 3 Reformulation of Inequality Constraints as Equality Constraints

4 In this work, inequality constraints for the controlled variables, i.e., product impurity  $I_{\text{prod}}$ , molar holdup in storage  
 5  $N_s$  and reboiler  $N_r$ , and temperature difference between reboiler and condenser  $\Delta T_{\text{rc}}$ , are converted into equality  
 6 constraints through the introduction of slack variables (see main text). The reformulation is done as follows:  
 7 Each inequality constraint of the form

$$g_i^{\min} \leq g_i(y) \leq g_i^{\max}, \quad (4)$$

8 is reformulated as an equality constraint by introducing a slack variable  $s_i$ :

$$g_i(y) + s_i = \frac{1}{2}(g_i^{\min} + g_i^{\max}). \quad (5)$$

9 We penalize the slack variable with the quadratically smoothed hinge loss penalty function (Zhang (2004)):

$$L(s_i, \delta) = M \left[ \max \left( 0, |s_i| - \frac{1}{2}(g_i^{\max} - g_i^{\min}) + \delta \right) \right]^2 \quad (6)$$

10 The penalty coefficient  $M > 0$  is set to 10,000 to balance the penalty term with the magnitude of the objective  
 11 function, which represents the cost savings. The penalty scaling factor  $\delta$  is chosen as 0.2 to impose stronger penalties

on constraint violations. Consequently, this results in effectively tighter bounds and a more conservative control behavior. In our reinforcement learning framework, all states except for the storage state, and control actions are scaled to the range  $[-1, 1]$ . The calculations of the slack variables and penalties are imposed on the scaled values of the controlled variables. The storage state is expressed as the rate of hourly demand, and its contribution to the penalty term is calculated in its unscaled value.

## 4 Hyperparameters

**Tab. 1.** Hyperparameters adapted from our previous work (Mayfrank et al., 2024). Where possible, the notation is consistent with the PPO paper (Schulman et al., 2017).

Hyperparameter	Value	Description
<b>General</b>		
$\beta$	$5 \cdot 10^{-5}$	reward calculation hyperparameter
$\sigma$	$(0.15, 0.15, 0.15, 0.15)^T$	standard deviation for action selection
$\gamma$	0.98	reward discount factor
$\lambda$	0.95	generalized advantage estimation hyperparameter
$\epsilon$	0.2	clipping hyperparameter
VF coeff.	5.0	value function coefficient
Entropy coeff.	$10^{-3}$	Entropy coefficient
$N_{\text{PPO}}$	8	number of parallel actors
$T_{\text{PPO}}$	512	control steps between updates to actor and critic
$M_{\text{PPO}}$	256	minibatch size
optimizer	Adam	optimizer used for updates to actor and critic
$K_{\text{PPO}}$	10	number of epochs per update
$\alpha$	$10^{-4}$	learning rate
max. gradient norm	0.5	gradient clipping value for actor update
Koopman MPC solver	ECOS, SCS	solver for Koopman OCPs (Domahidi et al., 2013; O’Donoghue et al., 2016)
$M$	10,000	penalty factor for slack variable usage
$\delta$	0.2	penalty scaling factor

## 1 Bibliography

- 2 Domahidi, A., Chu, E., and Boyd, S. (2013). ECOS: An SOCP solver for embedded systems. In *2013 European*  
3 *control conference (ECC)*, pages 3071–3076. IEEE.
- 4 Mayfrank, D., Mitsos, A., and Dahmen, M. (2024). End-to-end reinforcement learning of Koopman models for  
5 economic nonlinear model predictive control. *Computers & Chemical Engineering*, 190:108824.
- 6 O’Donoghue, B., Chu, E., Parikh, N., and Boyd, S. (2016). Conic optimization via operator splitting and homogeneous  
7 self-dual embedding. *Journal of Optimization Theory and Applications*, 169(3):1042–1068.
- 8 Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms.  
9 *arXiv preprint arXiv:1707.06347*.
- 10 Zhang, T. (2004). Solving large scale linear prediction problems using stochastic gradient descent algorithms. *Twenty-*  
11 *first international conference on Machine learning - ICML ’04*, page 116.