Koopman Constrained Policy Optimization: A Koopman operator theoretic method for differentiable optimal control in robotics

Matthew Retchin [1] **Brandon Amos** [2] **Steven L. Brunton** [3] **Shuran Song** [1]

Abstract

We introduce Koopman Constrained Policy Optimization (KCPO), combining implicitly differentiable model predictive control with a deep Koopman autoencoder for robot learning in unknown and nonlinear dynamical systems. KCPO is a new policy optimization algorithm that trains neural policies end-to-end with hard box constraints on controls. Guaranteed satisfaction of hard constraints helps ensure the performance and safety of robots. We perform imitation learning with KCPO to recover expert policies on the Simple Pendulum, Cartpole Swing-Up, Reacher, and Differential Drive environments, outperforming baseline methods in generalizing to out-of-distribution constraints in most environments after training.

1. Introduction

The problem of real-world control presents difficulties from nonlinearity and unknown dynamics. One possible solution is via model-based optimal control. Optimal control in an unknown system requires system identification, or the data-driven estimation of a mathematical model characterizing a system's dynamics (Fasel et al., 2021). Yin et al. (2022) use differentiable Riccati solving to obtain optimal controls with a Koopman autoencoder-based model, which learns a linear model from data in order to solve the classical control problem of Linear Quadratic Regulator (LQR). Like our work, Yin et al. (2022) is interested in box-constrained control trajectories, using the hyperbolic tangent function to squash the output of their LQR solver.

Trajectory optimization for nonlinear dynamical systems typically relies on local linearization. *Koopman operator theory* (Koopman, 1931; Koopman & Neumann, 1932; Budišić et al., 2012; Mezic, 2013; Otto & Rowley, 2021; Brunton et al., 2022; Colbrook & Townsend, 2021; Colbrook et al., 2023), on which Yin et al. (2022) relies, may provide an alternative path via global linearization: every point in state space shares one linear model of the dynamics (Brunton et al., 2022, Section 1.1). It is possible to employ functions that *lift* from the original state space, where the dynamics are nonlinear and difficult to both predict and control, to a new space where a linear model is sufficient to characterize the dynamics everywhere. Once dynamics have been linearized globally, the strong guarantees of optimality from classical control theory become available.

Koopman operator theory has recently been employed in many domains in robotics and control (Surana, 2016; Korda & Mezić, 2018; Peitz et al., 2020; Folkestad et al., 2020a;b;c; Peitz & Klus, 2020; Folkestad & Burdick, 2021; Wang et al., 2023; Calderón et al., 2021; Junker et al., 2022). This paper focuses on one approach to apply Koopman operator theory, a neural network architecture called a Koopman autoencoder that learns the lifting functions and linear operator from data (Lusch et al., 2018; Takeishi et al., 2017; Yeung et al., 2017; Mardt et al., 2018; Otto & Rowley, 2019; Mardt et al., 2020). While the first Koopman autoencoders simply predict dynamics, later works besides Yin et al. (2022) have successfully used Koopman autoencoders for optimal control (Korda & Mezić, 2018; Han et al., 2022; Li et al., 2020; King et al., 2022). These other works either are unconstrained control methods, or they have probabilistic constraints that are learned incrementally through training instead of being enforced from the start as *hard* constraints.

Choosing a loss for learning a quality Koopman embedding space is an open research topic. Most existing Koopman autoencoder methods use a combination of prediction loss and task loss, pre-training the Koopman model before using it for a policy trained separately, although Yin et al. (2022) learns the autoencoder end-to-end with task loss. Yin et al. (2022)'s differentiable LQR solver enables the use of task losses for learning Koopman dynamics.

Yin et al. (2022) does not explicitly consider the control constraints in the Koopman embedding space. Constraining the actions is crucial for safety in the control of robotic systems for flying and locomotion tasks. Serious damage to the robots or to surrounding humans and property could

[1]Columbia University [2]Meta AI [3]University of Washington, Seattle. Correspondence to: Matthew Retchin <mhr2145@columbia.edu>.

Published at the Differentiable Almost Everything Workshop of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. July 2023. Copyright 2023 by the author(s).

result if robots exceed their physical limits (Gu et al., 2022).

Contributions. We propose to use Koopman embeddings for learning dynamics for *Constrained Policy Optimization* (CPO) , *e.g.* as in Achiam et al. (2017), which we refer to as KCPO. KCPO performs constrained policy optimization, optimizing a policy that respects hard box constraints during both training and inference. KCPO generates box-constrained control trajectories for robots, bounding between an upper and lower limit. We experimentally study KCPO in fully-observable and deterministic environments (where perception is not a point of concern).

2. Background & Preliminaries

2.1. Model Predictive Control (MPC)

We solve a trajectory optimization problem with convex cost function $c(\tau_t)$, nonconvex dynamics $f(\tau_t)$, and box constraints $\underline{\mathbf{u}}$ and $\overline{\mathbf{u}}$, where $\tau_t := \{x_t, u_t\}$:

$$\tau^{\star}_{1:T} = \arg\min_{\tau_{1:T}} \sum_{t=1}^{T} c(\tau_t) \quad \text{subject to } \mathbf{x}_1 = \mathbf{x}_{\text{init}}, \mathbf{x}_{t+1} = f(\tau_t), \underline{\mathbf{u}} \leq \mathbf{u} \leq \overline{\mathbf{u}}. \tag{1}$$

MPC, *cf.* (Tedrake, Chapter 10), solves eq. (1) for $\mathbf{u}^*_1$ with a finite horizon T and then advances the limited horizon forward and solves eq. (1) in the next timestep.

2.2. Implicitly Differentiable Model Predictive Control

Amos et al. (2018), whose work KCPO builds upon, use an implementation of MPC with box-constrained controls, modifying the Control-Limited Differential Dynamic Programming heuristic (also known as Box-DDP) (Tassa et al., 2014). Appendix A.5 further describes their formulation. Amos et al. (2018)'s implementation of Box-DDP uses first-order linearization instead of the second-order linearization in Tassa et al. (2014). Following Amos et al. (2018), we will refer to the first-order iLQR version as "Box-DDP". Although linearization and quadratization are not needed for KCPO, linearization is required to generate a dataset from a Box-DDP-based expert for imitation learning, in which the expert can access to the true nonlinear dynamics equations.

2.3. Koopman Autoencoders

Koopman operator theory offers hope that it may be feasible to extend the benefits of existing linear control theory techniques to nonlinear dynamical systems. In 1931, Bernard Koopman showed that it is possible to diffeomorphically lift an original state $\mathbf{x}$, which has nonlinear dynamics, into a new space with linear dynamics (Koopman, 1931). The *Koopman operator* is a linear operator that completely characterizes the original nonlinear dynamics. In this newly lifted space, classical linear control techniques like LQR could be used (Brunton et al., 2022, Section 6).

The key challenge in Koopman theory is that the operator is infinite-dimensional that lives in a Hilbert space of measurement functions $g(\mathbf{x})$, also called *observables*. This is an obstacle indeed, as LQR only applies to finite-dimensional matrices. Thus, many researchers have focused on how to approximate Koopman operators with finite-dimensional matrices and also approximate the Koopman operator's true infinite-dimensional Hilbert space in finite dimensions .

Koopman autoencoders like those used in KCPO combine Koopman operator theory with the traditional autoencoder neural network architecture to learn approximate, finite-dimensional observables and the Koopman operator simultaneously from data.

A finite Koopman operator is likely to be limited compared to an infinite-dimensional operator. We use the Koopman autoencoder architecture from Lusch et al. (2018), which overcomes the limitations of an autoencoder's finite-dimensional bottleneck by using an auxiliary neural network to model the entire continuous spectrum of Koopman operator's eigenvalues. This allows the latent space to use significantly fewer dimensions than the true infinite-dimensional Koopman operator. We augment Lusch et al. (2018)'s original design with auxiliary neural networks to generate cost matrices for the optimal control problem formulation presented in eq. (1). KCPO's autoencoder extends Lusch et al. (2018)'s auxiliary network design from dynamics prediction to control tasks.

2.4. Koopman Dynamics for Control

The paradigm of end-to-end differentiable Koopman policies comes from Yin et al. (2022), who use analytical Riccati solving to obtain optimal controls with their Koopman autoencoder-based model, and they impose box constraints on the output of the solver with a hyperbolic tangent squashing function. However, squashing functions often lead to poor performance in practice; a numerical optimization method with built-in support for box constraints may be more appropriate (Tassa et al., 2014, Section III.A). Problematically, numerical optimization is not easily or efficiently differentiable. We propose to retain end-to-end differentiability in a box-constrained Koopman policy by replacing Yin et al. (2022)'s analytical Riccati solver layer with the implicitly differentiable constrained numerical MPC layer of Amos et al. (2018).

Unlike Lusch et al. (2018) and our work, the Koopman autoencoder of Yin et al. (2022) does not parameterize the continuous spectrum of the Koopman operator's eigenvalues. As a consequence, Yin et al. (2022)'s embedding space is much higher dimensional and requires far more parameters to model the dynamics than our work.

Watter et al. (2015); Banijamali et al. (2018) and follow-

up works also explore the idea of learning latent linear dynamics for end-to-end differentiable visual control. Although Watter et al. (2015); Banijamali et al. (2018) are not inspired by Koopman operator theory and do not label their autoencoder a Koopman autoencoder, their design is similar to ours in that they each rely on local linearization prior to iLQR or iLQG (iterative-Linear Quadratic Gaussian, which is iLQR with Gaussian noise). However, their iLQR layers are not differentiable and do not impose box constraints on controls. They also are focused on the dual problem of control and perception, while we restrict ourselves to just control by assuming full observability and determinism.

3. Koopman Constrained Policy Optimization

Our method estimates Koopman dynamics for control by combining a Koopman autoencoder with differentiable MPC and differentiating end-to-end through these components.

3.1. Combining Koopman Autoencoders with Differentiable MPC

The original differentiable MPC introduced by Amos et al. (2018) had a great limitation specifically when using neural networks to approximate the dynamics and cost function. The nonlinear neural networks introduce strong nonconvexity that prevents MPC from reaching a fixed point solution to eq. (1) (Amos et al., 2018). The requirement for fixed point solutions to trajectory optimization arises inherently when using the IFT to differentiate through MPC. If neural networks cannot be used in MPC, then the scope of applicability for differentiable MPC becomes limited.

Combining implicitly differentiable MPC with a Koopman autoencoder addresses this issue by producing linear dynamics without Taylor expansion. With linear dynamics, constrained MPC optimization provably converges to a fixed point, assuming the feasible set is nonempty and the cost function is convex. In most trajectory optimization, the cost function is convex: the quadratic distance function from current state to goal. We enable end-to-end training in the KCPO algorithm by composing the Koopman autoencoder and implicitly differentiable MPC (see algorithm 1 and fig. 1). Additional details on Amos et al. (2018)'s stability issues may be found in appendix A.6.

The positive semi-definiteness of $\mathbf{C}_t$ is a critical assumption for the feasibility of the trajectory optimization; otherwise, a solution will likely not exist. Ensuring that cost matrix $\mathbf{C}_t$ for the KCPO algorithm is positive semi-definite, the auxiliary neural network outputting the parameters for the cost function (AUXILIARYCOSTNN in algorithm 1) generates the lower triangle of $\mathbf{C}$'s Cholesky decomposition: $\mathbf{C} = \mathbf{L}\mathbf{L}^\top$. Thus, the cost $\mathbf{C}$ from AUXILIARYCOSTNN is positive semi-definite (Golub & Van Loan, 1996).

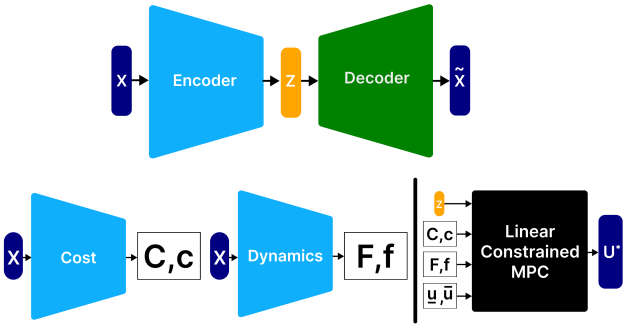

Figure 1. Koopman Constrained Policy Optimization architecture. An autoencoder transforms an observed state $\mathbf{X}$ into latent state vector $\mathbf{Z}$. Auxiliary neural networks generate dynamics and cost matrices for MPC, solving eq. (1) for optimal controls $\mathbf{U}^*$.

Algorithm 1 Koopman Constrained Policy Optimization

T is the MPC horizon length.
$\mathbf{x} \in \mathbb{R}^n$ is a state vector.
$\mathbf{u} \in \mathbb{R}^m$ is a control vector.
$\underline{\mathbf{u}}, \overline{\mathbf{u}} \in \mathbb{R}^m$ are the constraints on controls (eq. (1)).
$\mathbf{C} \in \mathbb{R}^{T \times n+m \times n+m}$ and $\mathbf{c} \in \mathbb{R}^{T \times n+m}$ are objective cost terms.
  $\mathbf{C}_t$ must be positive semi-definite.
$\mathbf{F} \in \mathbb{R}^{T \times n \times n+m}$ and $\mathbf{f} \in \mathbb{R}^{T \times n}$ are dynamics cost terms.

function KCPO($\mathbf{x}_{1:T}, \underline{\mathbf{u}}, \overline{\mathbf{u}}; \theta$)
  // Obtain the Koopman cost, dynamics, and states
  $\mathbf{C}, \mathbf{c} \leftarrow$ AUXILIARYCOSTNN($\mathbf{x}_1; \theta$)
  $\mathbf{F}, \mathbf{f} \leftarrow$ AUXILIARYDYNAMICSNN($\mathbf{x}_1; \theta$)
  $\mathbf{z}_{1:T} \leftarrow$ ENCODER($\mathbf{x}_{1:T}; \theta$)
  // Solve for the controls in the Koopman space
  $\mathbf{u}^\star_{1:T} \leftarrow$ DIFFERENTIABLEMPC$_{T,\underline{\mathbf{u}},\overline{\mathbf{u}}}(\mathbf{z}_1, \mathbf{C}, \mathbf{c}, \mathbf{F}, \mathbf{f})$
  // Decode back to the original space
  $\hat{\mathbf{x}}_{1:T} \leftarrow$ DECODER($\mathbf{z}_{1:T}; \theta$)
  return $\mathbf{u}^\star_{1:T}, \hat{\mathbf{x}}_{1:T}$
end function

3.2. Learning & Backpropagation

Training KCPO requires backpropagating through a task-specific policy loss and an autoencoding reconstruction loss.

$$\ell = \ell_{\text{policy}}(\mathbf{u}^\star_{1:T}) + \ell_{\text{reconstr}}(\hat{\mathbf{x}}_{1:T}). \tag{2}$$

Equation (2) differs from Yin et al. (2022)'s loss by lacking Koopman operator regularization and dynamics prediction error terms. In-loss regularization is unnecessary because we train with AdamW, a regularizing optimizer (Loshchilov & Hutter, 2014). We do not find the prediction error term helpful and omit it, noting that Yin et al. (2022) heavily downweight it. $\nabla_\theta \ell_{\text{reconstr}}$ is backpropagated through KCPO($\mathbf{x}_{\text{init}}; \theta$) with explicit differentiation, and $\nabla_\theta \ell_{\text{policy}}$ is backpropagated through KCPO($\mathbf{x}_{\text{init}}; \theta$) in two parts. Using the chain rule, PyTorch seamlessly connects two derivatives: first, the derivative of the MPC layer w.r.t. the loss, and

second, the derivative of the previous layers w.r.t. their inputs and parameters, even as those layers are differentiated using distinct methods (implicit and explicit, respectively) (Paszke et al., 2019).

4. Experiments & Results

We evaluate KCPO's efficacy with imitation learning to recover expert policies in the Simple Pendulum, Cartpole Swing-Up, Differential Drive (LaValle, 2006), and Reacher environments. The Reacher experiments were conducted using a custom environment inspired by the implementation of the Reacher task from Ivy Gym (Lenton et al., 2021). We compare with three baselines, each using the hyperbolic tangent activation function to squash input between $(-1, 1)$ before scaling to fit box constraints.

The first baseline is a Long Short-Term Memory-based Recurrent Neural Network (RNN) (Hochreiter & Schmidhuber, 1997). The initial state is given to the RNN and inputted recurrently until a control trajectory is fully generated. The second baseline modifies the "ReflexNet" of (Kurtz et al., 2022) to enforce hard constraints using hyperbolic tangent. The ReflexNet architecture is a multilayer perceptron and outputs the entire trajectory given the initial state:

$$\mathbf{u}_{1:T} = \pi_\theta(\mathbf{x}_0). \tag{3}$$

The last baseline, "RiccatiNet" is inspired by Yin et al. (2022). It slightly modifies our specific autoencoder architecture so as to work with Yin et al. (2022)'s approach of differentiable Riccati solving and squashing function. This enables a head-to-head comparison of implicitly differentiable MPC against Yin et al. (2022)'s Riccati solving and squashing function while holding the autoencoder constant.

These are good baselines for assessing KCPO because they can generate full box-constrained control trajectories given initial states. Hyperparameters are in appendix A.1. The source code to reproduce our experiments is available at `https://github.com/mhr/kcpo-icml`.

4.1. Out-of-Distribution Generalization

Our first experiment measures imitation loss through Mean Squared Error (MSE) between the expert and the trained policy π in the Differential Drive environment averaged over ten trials, each deterministically reproducible with a separate pseudo-random seed.

Figure 2 shows KCPO typically performing worse with baselines for a test set with constraints identical to those seen during training, but KCPO beats baseline performance for a test set with constraints different than those seen during training. This suggests the baselines may be overfitting to the training data. Similar generalization results exist for Simple Pendulum and Cartpole Swing-Up, while Reacher results in similar generalization within the margin of error, to be found in appendix A.3. Details on which constraints were used for each experiment can be found in appendix A.2.

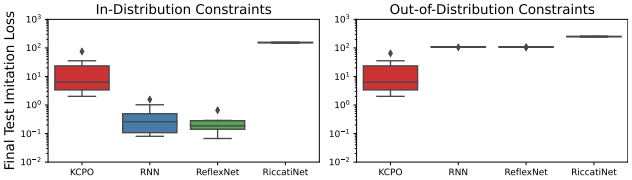

Figure 2. Test imitation loss at last epoch for Differential Drive. Left panel tests with in-distribution constraints. Right panel tests with new, out-of-distribution constraints unseen during training. (lower=better).

4.2. Speed

We timed one batch of KCPO during training and inference for the Cartpole Swing-Up environment using each model with MPC horizon $T = \{10, 20\}$ for ten batches (figures included in appendix A.4). An entire second of wall clock time is required for inference on a KCPO network with a horizon of 20 on a 13th Gen Intel(R) Core(TM) i9 with 32 GB of RAM. Thus, it is impractical to use KCPO for real-time control, but with a future improvement in speed, the control performance of KCPO would be valuable.

5. Conclusion

We have presented a new method for constrained policy optimization in unknown, highly nonlinear systems: Koopman Constrained Policy Optimization (KCPO). KCPO harnesses Koopman autoencoders' linearized dynamics with differentiable trajectory optimization for end-to-end trainable constrained policy optimization. KCPO exceeds or matches baseline performance in the Simple Pendulum, Cartpole Swing-Up, Reacher, and Differential Drive environments when tested with constraints unseen during training. KCPO is a new policy optimization algorithm that trains end-to-end with inviolable, hard box constraints on controls.

Future Extensions. KCPO is limited to applying only box constraints to controls. One approach for state constraints is Shi & Meng (2022), who concatenate the state vector with the latent embedding vector. Another approach could be to use a linear decoder like Korda & Mezić (2018) and Han et al. (2022). King et al. (2022) achieve state and control constraints, but these constraints are probabilistic, so the architecture is safe only for offline training. Another extension could borrow from Han et al. (2022) to achieve stochastic control robust to random perturbations in dynamics. However, Box-DDP (an extension of LQR) is already optimal for stochastic dynamics with a Gaussian noise distribution (Tedrake, Chapter 14).

References

Achiam, J., Held, D., Tamar, A., and Abbeel, P. Constrained policy optimization. In *Proceedings of the 34th International Conference on Machine Learning*, 2017. 2

Amos, B., Jimenez, I., Sacks, J., Boots, B., and Kolter, J. Z. Differentiable mpc for end-to-end planning and control. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. 2, 3, 9, 10

Banijamali, E., Shu, R., Bui, H., Ghodsi, A., et al. Robust locally-linear controllable embedding. In *International Conference on Artificial Intelligence and Statistics*, pp. 1751–1759. PMLR, 2018. 2, 3

Brunton, S. L., Budišić, M., Kaiser, E., and Kutz, J. N. Modern Koopman theory for dynamical systems. *SIAM Review*, 64(2):229–340, 2022. 1, 2

Budišić, M., Mohr, R., and Mezić, I. Applied Koopmanism a). *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 22(4):047510, 2012. 1

Calderón, H. M., Schulz, E., Oehlschlägel, T., and Werner, H. Koopman operator-based model predictive control with recursive online update. In *2021 European Control Conference (ECC)*, pp. 1543–1549. IEEE, 2021. 1

Colbrook, M. J. and Townsend, A. Rigorous data-driven computation of spectral properties of koopman operators for dynamical systems. *arXiv preprint arXiv:2111.14889*, 2021. 1

Colbrook, M. J., Ayton, L. J., and Szőke, M. Residual dynamic mode decomposition: robust and verified koopmanism. *Journal of Fluid Mechanics*, 955:A21, 2023. 1

Fasel, U., Kaiser, E., Kutz, J. N., Brunton, B. W., and Brunton, S. L. Sindy with control: A tutorial, 2021. 1

Folkestad, C. and Burdick, J. W. Koopman nmpc: Koopman-based learning and nonlinear model predictive control of control-affine systems. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7350–7356. IEEE, 2021. 1

Folkestad, C., Chen, Y., Ames, A. D., and Burdick, J. W. Data-driven safety-critical control: synthesizing control barrier functions with koopman operators. *IEEE Control Systems Letters*, 5(6):2012–2017, 2020a. 1

Folkestad, C., Pastor, D., and Burdick, J. W. Episodic koopman learning of nonlinear robot dynamics with application to fast multirotor landing. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9216–9222. IEEE, 2020b. 1

Folkestad, C., Pastor, D., Mezic, I., Mohr, R., Fonoberova, M., and Burdick, J. Extended dynamic mode decomposition with learned koopman eigenfunctions for prediction and control. In *2020 american control conference (acc)*, pp. 3906–3913. IEEE, 2020c. 1

Golub, G. H. and Van Loan, C. F. *Matrix computations*. Johns Hopkins University Press, 1996. 3

Gu, S., Yang, L., Du, Y., Chen, G., Walter, F., Wang, J., Yang, Y., and Knoll, A. A review of safe reinforcement learning: Methods, theory and applications. *arXiv preprint arXiv:2205.10330*, 2022. 2

Han, M., Euler-Rolle, J., and Katzschmann, R. K. DeSKO: Stability-assured robust control with a deep stochastic koopman operator. In *International Conference on Learning Representations*, 2022. 1, 4

Hendrycks, D. and Gimpel, K. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 7

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735. 4

Junker, A., Pape, K., Timmermann, J., and Trächtler, A. Holistic adaptive controller and observer design using the koopman operator. *arXiv preprint arXiv:2211.09512*, 2022. 1

King, E., Drgona, J., Tuor, A. R., Abhyankar, S. G., Bakker, C., Bhattacharya, A., and Vrabie, D. L. Koopman-based differentiable predictive control for the dynamics-aware economic dispatch problem. In *American Control Conference (ACC 2022)*. IEEE, 2022. doi: 10.23919/ACC53348.2022.9867379. 1, 4

Kolter, Z. and Duvenaud, D. Chapter 5: Differentiable optimization. URL http://implicit-layers-tutorial.org/differentiable_optimization/. 9

Koopman, B. and Neumann, J. v. Dynamical systems of continuous spectra. *Proceedings of the National Academy of Sciences of the United States of America*, 18(3):255, 1932. 1

Koopman, B. O. Hamiltonian systems and transformation in Hilbert space. *Proceedings of the National Academy of Sciences*, 17(5):315–318, 1931. 1, 2

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

Korda, M. and Mezić, I. Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. *Automatica*, 93:149–160, 2018. ISSN 0005-1098. doi: https://doi.org/10.1016/j.automatica.2018.03.046. 1, 4

Krantz, S. G. and Parks, H. R. *The implicit function theorem: History, theory, and applications*. Birkhäuser, 2002. 9

Kurtz, V., Li, H., Wensing, P. M., and Lin, H. Mini cheetah, the falling cat: A case study in machine learning and trajectory optimization for robot acrobatics. *International Conference on Robotics and Automation (ICRA)*, 2022. doi: 10.1109/icra46639.2022.9812120. 4

LaValle, S. M. *Planning Algorithms*. Cambridge University Press, 2006. doi: 10.1017/CBO9780511546877. 4

Lenton, D., Pardo, F., Falck, F., James, S., and Clark, R. Ivy: Templated deep learning for inter-framework portability. *arXiv preprint arXiv:2102.02886*, 2021. 4

Li, Y., He, H., Wu, J., Katabi, D., and Torralba, A. Learning compositional koopman operators for model-based control. In *International Conference on Learning Representations*, 2020. 1

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *ArXiv*, abs/1711.05101, 2014. 3, 7

Lusch, B., Kutz, J. N., and Brunton, S. L. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications*, 9(1), 2018. doi: 10.1038/s41467-018-07210-0. 1, 2

Mardt, A., Pasquali, L., Wu, H., and Noé, F. VAMPnets: Deep learning of molecular kinetics. *Nature Communications*, 9(5), 2018. 1

Mardt, A., Pasquali, L., Noé, F., and Wu, H. Deep learning markov and koopman models with physical constraints. In *Mathematical and Scientific Machine Learning*, pp. 451–475. PMLR, 2020. 1

Mezic, I. Analysis of fluid flows via spectral properties of the Koopman operator. *Annual Review of Fluid Mechanics*, 45:357–378, 2013. 1

Otto, S. E. and Rowley, C. W. Linearly-recurrent autoencoder networks for learning dynamics. *SIAM Journal on Applied Dynamical Systems*, 18(1):558–593, 2019. 1

Otto, S. E. and Rowley, C. W. Koopman operators for estimation and control of dynamical systems. *Annual Review of Control, Robotics, and Autonomous Systems*, 4, 2021. 1

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. 4

Peitz, S. and Klus, S. Feedback control of nonlinear pdes using data-efficient reduced order models based on the koopman operator. In *The Koopman Operator in Systems and Control*, pp. 257–282. Springer, 2020. 1

Peitz, S., Otto, S. E., and Rowley, C. W. Data-driven model predictive control using interpolated koopman generators. *SIAM Journal on Applied Dynamical Systems*, 19 (3):2162–2193, 2020. 1

Shi, H. and Meng, M. Q.-H. Deep koopman operator with control for nonlinear systems. *IEEE Robotics and Automation Letters*, 7(3):7700–7707, 2022. doi: 10.1109/lra.2022.3184036. 4

Surana, A. Koopman operator based observer synthesis for control-affine nonlinear systems. In *55th IEEE Conference on Decision and Control (CDC*, pp. 6492–6499, 2016. 1

Takeishi, N., Kawahara, Y., and Yairi, T. Learning koopman invariant subspaces for dynamic mode decomposition. In *Advances in Neural Information Processing Systems*, pp. 1130–1140, 2017. 1

Tassa, Y., Mansard, N., and Todorov, E. Control-limited differential dynamic programming. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1168–1175. IEEE, 2014. 2

Tedrake, R. *Underactuated Robotics*. 2, 4

Wang, Z., Tan, J., and Liu, C. K. Koopman operators for modeling and control of soft robotics, 2023. 1

Watter, M., Springenberg, J., Boedecker, J., and Riedmiller, M. Embed to control: A locally linear latent dynamics model for control from raw images. *Advances in neural information processing systems*, 28, 2015. 2, 3

Yeung, E., Kundu, S., and Hodas, N. Learning deep neural network representations for koopman operators of nonlinear dynamical systems. *arXiv preprint arXiv:1708.06850*, 2017. 1

Yin, H., Welle, M. C., and Kragic, D. Embedding koopman optimal control in robot policy learning. *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022. doi: 10.1109/iros47612.2022.9981540. 1, 2, 3, 4

A. Appendix

A.1. Hyperparameters

Table 1 displays the hyperparameters for the experiments described in section 4. Please note that the choice of hidden layer activation functions differ between models because we observed these functions to be most optimal empirically.

Table 1. Hyperparameters

MPC Horizon Length	10
Epochs	250
Total Training Samples	1000
Total Testing Samples	100
Total Validation Samples	100
State Dimensions	2 (Pendulum), 5 (Cartpole), 4 (Reacher), 3 (Differential Drive)
Control Dimensions	1 (Pendulum), 1 (Cartpole), 2 (Reacher), 2 (Differential Drive)
Koopman Operator Dimensionality	2×2 (Pendulum), 6×6 (Cartpole), 4×4 (Reacher), 6×6 (Differential Drive)
Control Matrix Dimensionality	2×1 (Pendulum), 6×1 (Cartpole), 4×1 (Reacher), 6×6 (Differential Drive)
RNN Hidden Layer Activation Function	ReLU
ReflexNet & KCPO Hidden Layer Activation Function	GELU (Hendrycks & Gimpel, 2016)
Optimizer	AdamW (Loshchilov & Hutter, 2014)
Learning Rate	1×10^{-3}

A.2. In-Distribution and Out-of-Distribution Constraints

Table 2 contains the box constraints used in the generalization experiments described in section 4.1 for the Simple Pendulum, Cartpole Swing-Up, Reacher, and Differential Drive environments.

Table 2. Test Constraints

Environment	In-Distribution	Out-of-Distribution
Simple Pendulum	(-2, 2)	(-1, 1)
Cartpole Swing-Up	(-10, 10)	(-5, 5)
Reacher	(-1, 1)	(-0.5, 0.5)
Differential Drive	(-100, 100)	(-80, 80)

A.3. Additional experimental results for out-of-distribution generalization using new constraints

Figures 3 to 5 depict our generalization experiments described in section 4.1 for the Simple Pendulum, Cartpole Swing-Up, and Reacher environments, respectively. In these experiments, unlike with our Differential Drive experiment, KCPO performs roughly on par or slightly worse than the baselines for in-distribution constraints, but the trend discussed in section 4.1 of KCPO performing better than baselines with out-of-distribution constraints continues in these other three environments.

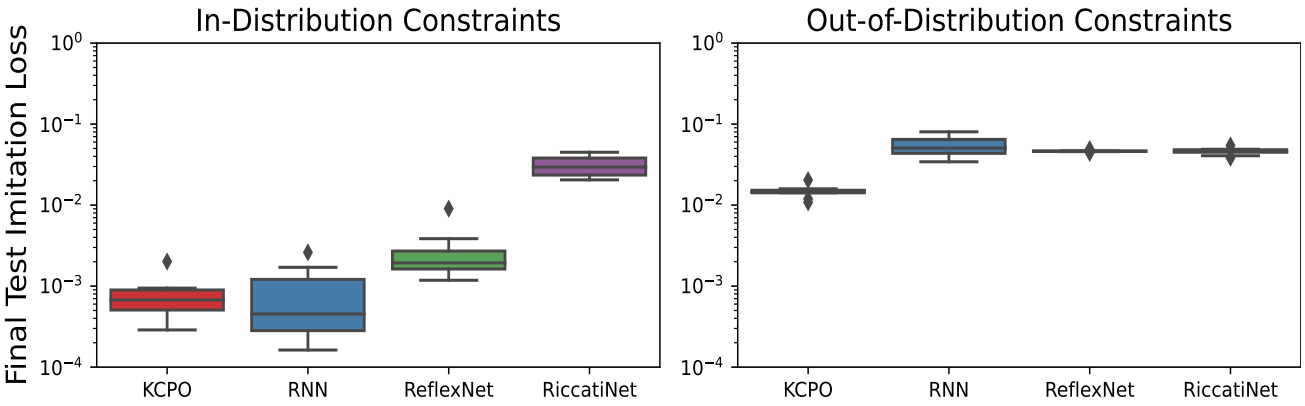

Figure 3. Test imitation loss at last epoch for **Simple Pendulum**. Left panel tests with in-distribution constraints. Right panel tests with new, out-of-distribution constraints unseen during training. (lower=better).

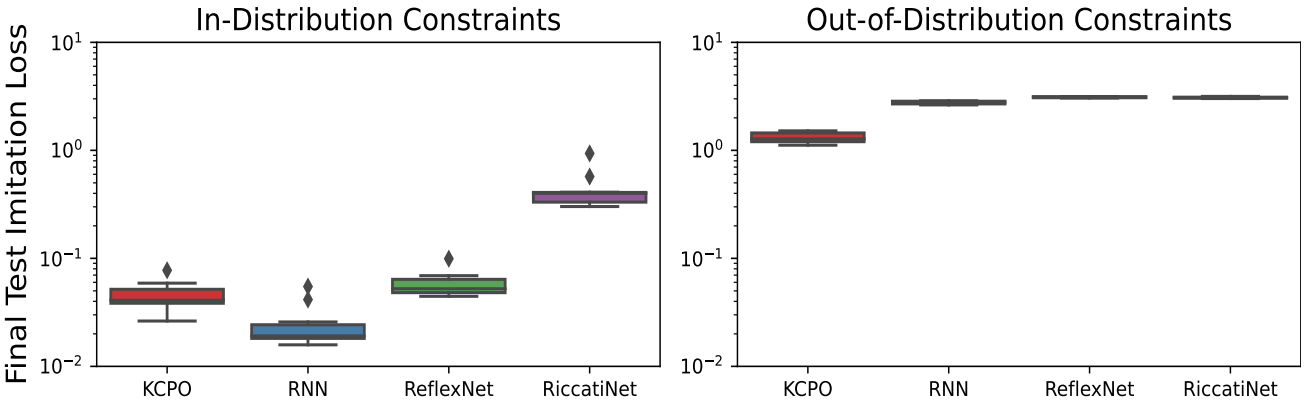

Figure 4. Test imitation loss at last epoch for **Cartpole Swing-Up**. Left panel tests with in-distribution constraints. Right panel tests with new, out-of-distribution constraints unseen during training. (lower=better).

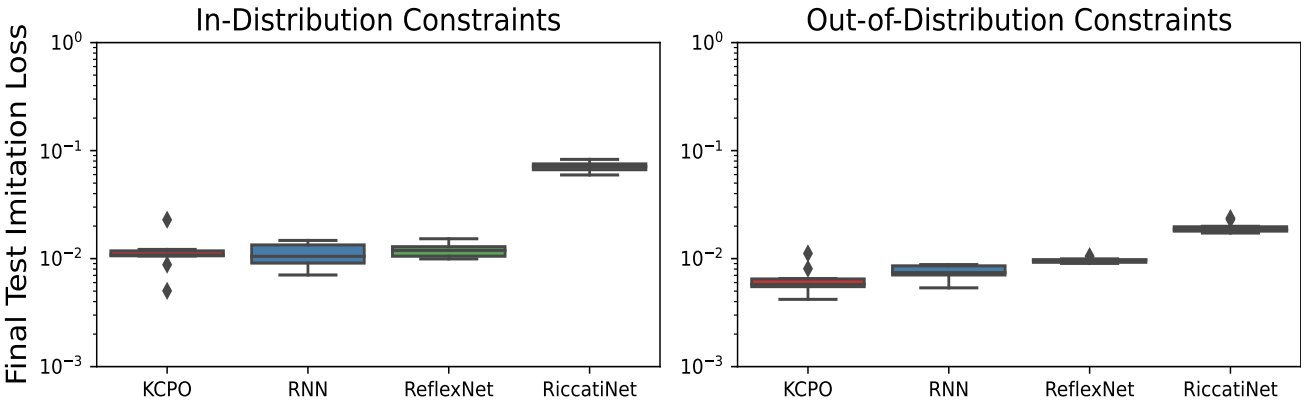

Figure 5. Test imitation loss at last epoch for **Reacher**. Left panel tests with in-distribution constraints. Right panel tests with new, out-of-distribution constraints unseen during training. (lower=better).

A.4. Timing

Figure 6 depicts our timing experiments described in section 4.2.

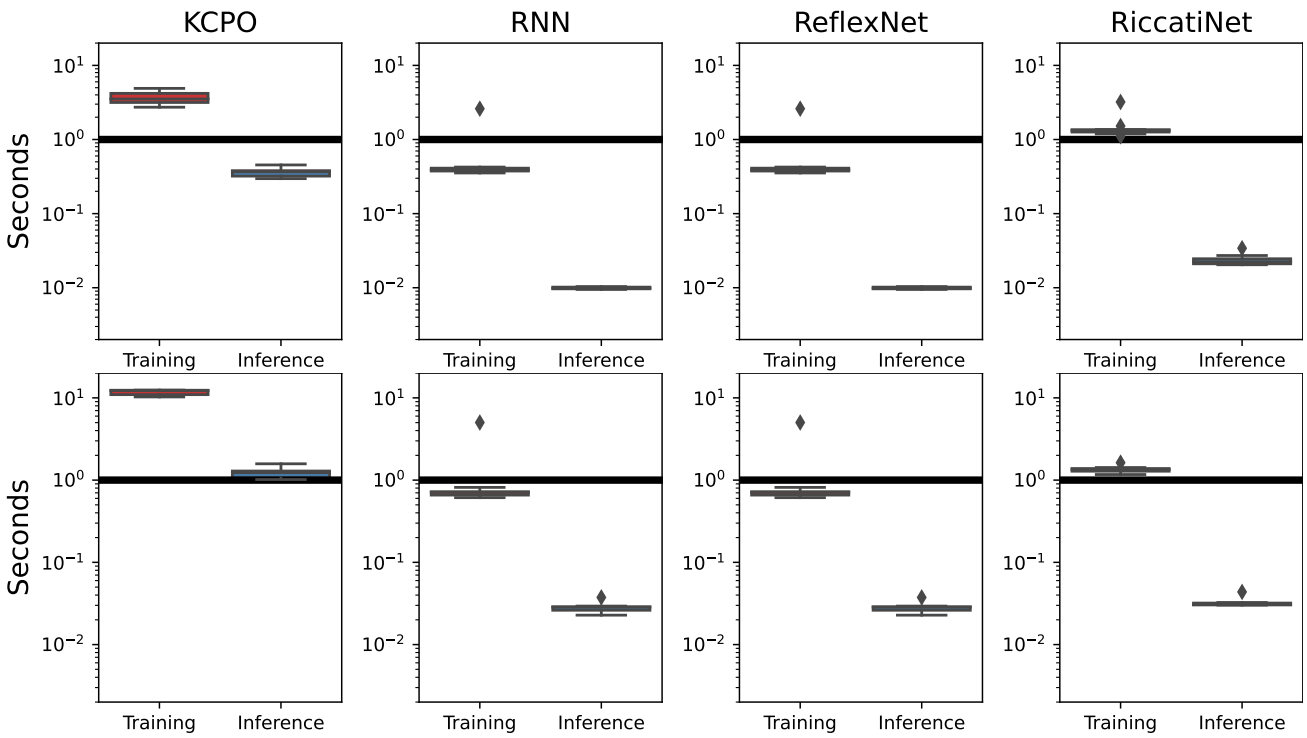

Figure 6. The X-axis is separated into categories for training and inference forward passes of each method (KCPO, RNN, ReflexNet, and RiccatiNet), and the Y-axis is time in seconds per forward pass of a single batch of data (lower=better). A black line is drawn where the time reaches one second to demonstrate when a given controller's speed becomes far too slow for real-time control. The top row is with horizon=10, while the bottom row is with horizon=20.

A.5. Differentiable MPC with Implicit Function Theorem

In practice, most constrained MPC solvers are not explicitly differentiable, making it difficult to use them to train neural policies with hard constraints. Following Amos et al. (2018) and Kolter & Duvenaud, let us treat a constrained MPC solver as a function $\mathbf{z}^\star(\mathbf{x})$, with $\mathbf{x}$ being the input state and $\mathbf{z}^\star$ being the optimal outputs (both the optimal primal and dual variables). In practice, $\mathbf{z}^\star(\mathbf{x})$ does not easily admit an explicit definition in terms of $\mathbf{x}$. A consequence of lacking such a definition is that $\frac{\partial \mathbf{z}^\star(\mathbf{x})}{\partial \mathbf{x}}$ cannot be computed using the typical automatic differentiation pipeline. One could compute $\frac{\partial z^\star(x)}{\partial x}$ by programming an MPC solver with differentiable operations and unrolling all the operations into a differentiable computational graph. However, computing that Jacobian via unrolling is prohibitively slow in practice because it often requires hundreds of iterations for an MPC solver to reach a fixed point.

Fortunately, the Implicit Function Theorem (IFT) provides an alternative differentiation method (Krantz & Parks, 2002). The constrained MPC solver may be considered to be an *implicit* function $g(\mathbf{x}, \mathbf{z}^\star)$, which parameterizes a new function in terms of both the independent and dependent variable. Intuitively, our choice of $g(\mathbf{x}, \mathbf{z}^\star)$ is an optimality function where suboptimal inputs u will result in $g(\mathbf{x}, \mathbf{u}) > 0$, but the root $g(\mathbf{x}, \mathbf{z}^\star) = 0$ exists for the fixed point solution to the MPC optimization.

With this implicit formulation of the constrained MPC solver function, $\frac{\partial g(\mathbf{x}, \mathbf{z}^\star)}{\partial \mathbf{x}}$ can be *solved for*, under the condition that $g(\mathbf{x}, \mathbf{z}^\star) = 0$. First, let us define the optimality function, which is at a root at MPC's optimal fixed point $\mathbf{z}^\star$, satisfying the aforementioned condition.

$$g(\mathbf{x}, \mathbf{z}^\star(\mathbf{x})) = 0 \tag{4}$$

In the next step of the theorem, both sides may be differentiated w.r.t. x.

$$\frac{\partial g(\mathbf{x}, \mathbf{z}^\star(\mathbf{x}))}{\partial \mathbf{x}} = 0 \tag{5}$$

The chain rule expands the partial derivative of eq. (5) into two new partial derivatives, each w.r.t. $\mathbf{x}$. Because $g(\mathbf{x}, \mathbf{z}^\star)$ is a multivariate function, both terms must be summed together.

$$\frac{\partial g(\mathbf{x}, \mathbf{z}^\star)}{\partial \mathbf{x}} + \frac{\partial g(\mathbf{x}, \mathbf{z}^\star)}{\partial \mathbf{z}^\star} \frac{\partial \mathbf{z}^\star(\mathbf{x})}{\partial \mathbf{x}} = 0 \tag{6}$$

Because this differentiation takes place at a root, a linear system of equations can be set up, and $\frac{\partial \mathbf{z}^\star(\mathbf{x})}{\partial \mathbf{x}}$ can be solved for.

$$\frac{\partial \mathbf{z}^\star(\mathbf{x})}{\partial \mathbf{x}} = -\left(\frac{\partial g(\mathbf{x}, \mathbf{z}^\star)}{\partial \mathbf{z}^\star}\right)^{-1} \frac{\partial g(\mathbf{x}, \mathbf{z}^\star)}{\partial \mathbf{x}} \tag{7}$$

A.6. Preventing Unstable Training via Koopman Operator Theory

There are two issues where training in Amos et al. (2018) became unstable.

First, there were occasions during training when MPC optimization fails to reach a fixed point for some or all samples in a batch. Amos et al. (2018) designed their training procedure to detach from the ruined, unconverged samples to prevent spoiling of the training with a runtime exception.

Second, for a task involving simultaneous system identification and cost function learning when imitating an expert policy, training did not converge. The authors found that cycling between training the cost function parameters and training their system parameters was the only path to solve the task. Even the method of detaching unconverged samples was unhelpful.

KCPO addresses both instability issues by eliminating the nonconvexity of the dynamics via Koopman operator theory: linearity implies MPC always converges. Batch gradient descent therefore can be used for constrained policy optimization with deep neural networks.

Empirically, we did find that there was an issue with reaching fixed points on occasion during experiments with Differential Drive only.