Extended Dynamic Mode Decomposition with Jacobian Residual Penalization for Learning Bilinear, Control-affine Koopman Models

Brian E. Jackson
Robotics Institute
Carnegie Mellon University
brianjackson@cmu.edu

Jeong Hun Lee
Robotics Institute
Carnegie Mellon University
jeonghunlee@cmu.edu

Kevin Tracy
Robotics Institute
Carnegie Mellon University
ktracy@cmu.edu

Zachary Manchester
Robotics Institute
Carnegie Mellon University
zacm@cmu.edu

1 Introduction

Controlling complex, underactuated, and highly nonlinear autonomous systems remains an active area of research in robotics, despite decades of previous work exploring effective algorithms and the development of substantial theoretical analysis. Classical approaches typically rely on local linear approximations of the nonlinear system, which are then used in any of a multitude of linear control techniques, such as PID, pole placement, Bode analysis, H-infinity, LQR, or linear MPC. These approaches only work well if the states of the system always remain close to the linearization point or reference trajectory. The region for which these linearizations remain valid can be extremely small for highly nonlinear systems. Alternatively, model-based methods for nonlinear control based on optimal control have shown great success, as long as the model is well known and an accurate estimate of the global state can be provided. These model-based techniques leverage centuries of insight into dynamical systems and have demonstrated incredible performance on extremely complicated autonomous systems [1, 2, 3, 4] . On the other hand, data-driven techniques such as reinforcement learning have received tremendous attention over the last decade and have begun to demonstrate impressive performance and robustness for complicated robotic systems in unstructured environments [5, 6, 7]. While these approaches are attractive since they require little to no previous knowledge about the system, they often require large amounts of data and fail to generalize outside of the domain or task on which they were "trained."

In this work we propose a novel method that combines the benefits of model-based and data-driven methods, based on recent work applying Koopman Operator Theory to controlled dynamical systems [8, 9, 10, 11, 12, 13] By leveraging data collected from an unknown dynamical system along with derivative information from an approximate analytical model, we can efficiently learn a bilinear representation of the system dynamics that performs well when used in traditional model-based control techniques such as linear MPC. The result is nonlinear dynamical system that is expressive enough to capture the full nonlinear dynamics across a broad range of the state space, but has a well-defined structure that is very amenable to specialized optimization-based controllers. By leveraging information from an analytical model, we can dramatically reduce the number of samples required to learn a good approximation of the true nonlinear dynamics.

TODO: bullet out contributions.

TODO: add summary of paper layout.

6th Conference on Robot Learning (CoRL 2022), Auckland, New Zealand.

2 Background and Related Work

2.1 The Koopman Operator

The theoretical underpinnings of the Koopman operator and its application to dynamical systems has been extensively studied, especially within the last decade [14, 15, 9, 16]. Rather than describe the theory in detail, we highlight the key concepts employed by the current work, and defer the motivated reader to the existing literature on Koopman theory.

We start by assuming we have some discrete approximation a of controlled nonlinear, time-dynamical system whose underlying continuous dynamics are Lipschitz continuous:

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

where $x \in \mathcal{X} \subseteq \mathbb{R}^{N_x}$ is the state vector and $u_k \in \mathbb{R}^{N_u}$ is the control vector. This discrete approximation can be obtained for any continuous dynamical system in many ways, including implicit and explicit Runge-Kutta methods, or by solving the Discrete Euler-Lagrange equations TODO: add citations.

The key idea behind the Koopman operator is that the nonlinear finite-dimensional discrete dynamics (1) can be represented by an infinite-dimensional *bilinear* system:

$$y_{k+1} = Ay_k + Bu_k + \sum_{i=1}^{m} u_{k,i} C_i y_k = g(y, u) \tag{2}$$

where $y = \phi(x)$ is a mapping from the finite-dimensional state space $\mathcal{X}$ to the (possibly) infinite-dimensional Hilbert space of *observables* $y \in \mathcal{Y}$. We also assume the inverse map is approximately linear: $x = Gy$. In practice, we approximate (1) by choosing ϕ to be some arbitrary finite set of nonlinear functions of the state variables, which in general include the states themselves such that the linear mapping $G \in \mathbb{R}^{N_x \times N_y}$ is exact. Intuitively, ϕ "lifts" our states into a higher dimensional space where the dynamics are approximately (bi)linear, effectively trading dimensionality for (bi)linearity. This idea should be both unsurprising and familiar to most roboticsts, since similar techniques have already been employed in other forms, such as maximal-coordinate representations of rigid body dynamics [17] TODO: add citations, the "kernel trick" for state-vector machines TODO: add citation, or the observation that solving large, sparse nonlinear optimization problems is often more effective than solving small, tightly-coupled dense problems TODO: add citations from the trajectory optimization literature.

The lifted bilinear system (2) can be easily learned from samples of the system dynamics (x_i^+, x_i, u_i) (where x^+ is the state at the next time step) using extended Dynamic Mode Decomposition (eDMD), which is just the application of linear least squares (LLS) to the lifted states. Details of this method will be covered later when we introduce our adaptation of eDMD and present an effective numerical technique for solving the resulting LLS problems.

3 EDMD with Jacobian Residual-Penalization

Existing Koopman-based approaches to learning dynamical systems only rely on samples of the unknown dynamics. Here we present a novel method for incorporating prior knowledge about the dynamics by adding derivative information of an approximate model into the data set to be learned via eDMD.

Given P samples of the dynamics (x_i^+, x_i, u_i), and an approximate discrete dynamics model

$$x^+ = \tilde{f}(x, u) \tag{3}$$

we can evaluate the Jacobians of our approximate model $\hat{f}$ at each of the sample points: $\tilde{A}_i = \frac{\partial \tilde{f}}{\partial x}, \tilde{B}_i = \frac{\partial \tilde{f}}{\partial x}$. After choosing nonlinear mapping $\phi : \mathbb{R}^{N_x} \mapsto \mathbb{R}^{N_y}$ We then want to find a bilinear dynamics model (2) that matches the Jacobians of our approximate model, while also matching our

dynamics samples. If we define $\hat{A}_j \in \mathbb{R}^{N_x \times N_x}$ and $\hat{B}_j \in \mathbb{R}^{N_x \times N_u}$ to be the Jacobians of our bilinear dynamics model, projected back into the original state space, our objective is to find the matrices $A \in \mathbb{R}^{N_y \times N_y}, B \in \mathbb{R}^{N_y \times N_u}$, and $C_{1:m} \in \mathbb{R}^{N_u} \times \mathbb{R}^{N_y \times N_y}$ that minimize the following objective:

$$(1-\alpha)\sum_{j=1}^{P}\left\|\hat{y}_j - y_j^+\right\|_2^2 + \alpha\sum_{j=1}^{P}\left\|\hat{A}_j - \tilde{A}_j\right\|_2^2 + \left\|\hat{B}_j - \tilde{B}_j\right\|_2^2 \tag{4}$$

where $\hat{y}_j^+ = g\left(\phi(x_j), u_j\right)$ is the output of our bilinear dynamics model, and $y_j^+ = \phi(y_j^+)$ is the actual lifted state (i.e. observables) at the next time step.

While not immediately apparent, we can minimize (4) using linear least-squares, using techniques similar to those used previously in the literature TODO: cite CalTech papers.

To start, we combine all the data we're trying to learn in a single matrix:

$$E = \begin{bmatrix} A & B & C_1 & \dots & C_m \end{bmatrix} \in \mathbb{R}^{N_y \times N_z} \tag{5}$$

where $N_z = N_y + N_u + N_y \cdot N_u$. We now rewrite the terms in (4) in terms of E. By defining the vector

$$z = \begin{bmatrix} y^T & u^T & u_1 y^T & \dots & u_m y^T \end{bmatrix} \in \mathbb{R}^{N_z} \tag{6}$$

we can write down the output of our bilinear dynamics (2) as

$$\hat{y}^+ = Ez. \tag{7}$$

The projected Jacobians of our bilinear model, $\hat{A}$ and $\hat{B}$, are simply the Jacobians of the bilinear dynamics in terms of the original state. We obtain these dynamics by "lifting" the state via ϕ and then projecting back onto the original states using G:

$$x^+ = G\left(A\phi(x) + Bu + \sum_{i=1}^{m} u_i C_i \phi(x)\right) = \hat{f}(x,u) \tag{8}$$

Differentiating these dynamics give us our projected Jacobians:

$$\hat{A}_j = G\frac{\partial \hat{f}}{\partial x}(x_j, u_j) = G\left(A + \sum_{i=1}^{m} u_{j,i} C_i\right)\Phi(x_j) = GE\bar{A}(x_j, u_j) = GE\bar{A}_j \tag{9a}$$

$$\hat{B}_i = G\frac{\partial \hat{f}}{\partial u}(x_j, u_j) = G\Big(B + \begin{bmatrix} C_1 x_j & \dots & C_m x_j \end{bmatrix}\Big) = GE\bar{B}(x_j, u_j) = GE\bar{B}_j \tag{9b}$$

where $\Phi(x) = \partial\phi/\partial x$ is the Jacobian of the nonlinear map ϕ, and

$$\bar{A}(x,u) = \begin{bmatrix} I \\ 0 \\ u_1 I \\ u_2 I \\ \vdots \\ u_m I \end{bmatrix} \in \mathbb{R}^{N_z \times N_x}, \quad \bar{B}(x,u) = \begin{bmatrix} 0 \\ I \\ [x\ 0\ \dots\ 0] \\ [0\ x\ \dots\ 0] \\ \vdots \\ [0\ 0\ \dots\ x] \end{bmatrix} \in \mathbb{R}^{N_z \times N_u}. \tag{10}$$

Substituting (7) and (9) into (4), we can rewrite our LLS problem as:

$$\underset{E}{\text{minimize}} \quad \sum_{j=0}^{P}(1-\alpha)\left\|Ez_j - y_j^+\right\|_2^2 + \alpha\left\|GE\bar{A}_j - \tilde{A}_j\right\|_2^2 + \alpha\left\|GE\bar{B}_j - \tilde{B}_j\right\|_2^2 \tag{11}$$

which is equivalent to

$$\underset{E}{\text{minimize}} \quad (1-\alpha)\left\|E\mathbf{Z_{1:P}} - \mathbf{Y^+_{1:P}}\right\|_2^2 + \alpha\left\|GE\bar{\mathbf{A}}_{\mathbf{1:P}} - \tilde{\mathbf{A}}_{\mathbf{1:P}}\right\|_2^2 + \alpha\left\|GE\bar{\mathbf{B}}_{\mathbf{1:P}} - \tilde{\mathbf{B}}_{\mathbf{1:P}}\right\|_2^2 \tag{12}$$

where $\mathbf{Z_{1:P}} \in \mathbb{R}^{N_z \times P} = [z_1\ z_2\ \dots\ z_P]$ horizontally concatenates all of the samples (equivalent definition for $\mathbf{Y^+_{1:P}} \in \mathbb{R}^{N_y \times P}$, $\bar{\mathbf{A}}_{\mathbf{1:P}} \in \mathbb{R}^{N_z \times N_x \cdot P}$, $\tilde{\mathbf{A}}_{\mathbf{1:P}} \in \mathbb{R}^{N_z \times N_x \cdot P}$, $\bar{\mathbf{B}}_{\mathbf{1:P}} \in \mathbb{R}^{N_z \times N_u \cdot P}$, and $\tilde{\mathbf{B}}_{\mathbf{1:P}} \in \mathbb{R}^{N_z \times N_u \cdot P}$).

We can rewrite (14) in standard form using the "vec trick"

$$\text{vec}(AXB) = (B^T \otimes A)\text{vec}(X) \tag{13}$$

where $\text{vec}(A)$ stacks the columns of A into a single vector.

Setting E in (14) equal to X in (13), we get

$$\underset{E}{\text{minimize}} \left\| \begin{bmatrix} (1-\alpha) \cdot (\mathbf{Z_{1:P}})^T \otimes I_{N_y} \\ \alpha \cdot (\bar{\mathbf{A}}_{\mathbf{1:P}})^T \otimes G \\ \alpha \cdot (\bar{\mathbf{G}}_{\mathbf{1:P}})^T \otimes G \end{bmatrix} \text{vec}(E) + \begin{bmatrix} (1-\alpha) \cdot \text{vec}(\mathbf{Y}^+_{\mathbf{1:P}}) \\ \alpha \cdot \text{vec}(\tilde{\mathbf{A}}_{\mathbf{1:P}}) \\ \alpha \cdot \text{vec}(\tilde{\mathbf{G}}_{\mathbf{1:P}}) \end{bmatrix} \right\|_2^2 \tag{14}$$

such that the matrix of cofficients has $(N_y + N_x^2 + N_x \cdot N_u) \cdot P$ rows and $N_y \cdot N_z$ columns.

4 Results

All continuous dynamics were discretized with an explicit fourth-order Runge Kutta integrator.

4.1 Cartpole

As a simple benchmark example, we use the canonical cartpole system. In lieu of an actual hardware experiment (left for future work), we specify two separate analytical models used in simulation: the *nominal* model is a simple cartpole system without friction or damping, whereas the *simulated* model, used exclusively for simulating the system in place of a real hardware platform, includes viscous damping on both degrees of freedom, a tanh Coloumb friction model for the cart, and a deadband on the control signal. Additionally, we altered the mass of the cart and pole from the nominal model by 20% and 25%, respectively. See the provided code for the actual values and models used in the experiments.

We split the analysis of this system into two separate tasks: stabilization about the upward unstable equilibrium from perturbed initial condtions, and the swing-up task where the system must successfully stabilize after starting from the downward equilibrium.

4.1.1 Training

All models were trained by simulating the "real" system with an arbitrary controller to collect data in the region of the state space relevant to the task. A set of fixed-length trajectories were collected, each at a sample rate of 20 Hz. The bilinear eDMD model was trained using the same approach in [18]. For the proposed jDMD method, the Jacobians of the nominal model were calculated at each of the sample points and the bilinear model was learned using the approach outlined in Section 3.

4.1.2 Stabilization

For stabilizing the system about the upward unstable equilibrium we used an LQR controller designed using to nominal model to collect trajectories on the simulated system. To learn the bilinear models, we used the following nonlinear mapping: $\phi(x) = [\ 1,\ x,\ \sin(x),\ \cos(x),\ \sin(2x)\ ] \in \mathbb{R}^{17}$. After learning both models, an MPC controller was designed using the nominal, eDMD, and jDMD models. For the learned bilinear models, MPC controllers were designed using the "lifted" Jacobians of the bilinear dynamics and the "projected" Jacobians (9). The MPC controllers linearized the dynamics about the equilibrium of $x = [\ 0,\ \pi,\ 0,\ , 0\ ]$ and solved the resulting equality-constrained quadratic program using Riccati recursion and a horizon of 41 time steps (2 seconds).

To analyze sample efficiency of the algorithms, we trained the bilinear models with an increasing number of samples. The Each controller was tested using 100 different initial conditions sampled from a uniform distribution of initial conditions

The distance of the state after 4 seconds for the LQR and MPC controllers is shown in Figure

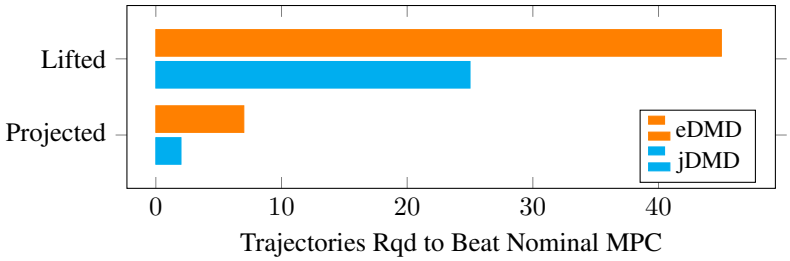
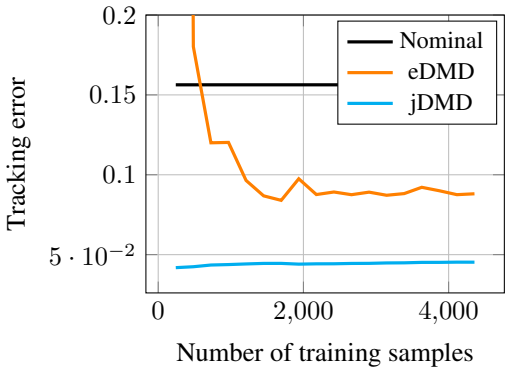
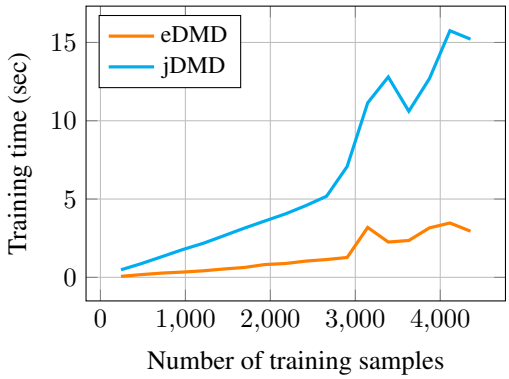


Figure 1: Number of training trajectories requires to beat the nominal MPC controller. The criteria is the L2 norm of the state from the goal state after 4 seconds. The "Lifted" MPC controllers compute the MPC solution in the lifted state space (17 states), whereas the "Projected" MPC contollers project the Jacobians of the bilinear system back into the original state space.



(a) MPC tracking error vs training samples for the cartpole. Tracking error is defined as the average L2 error over all the test trajectories between the reference and simulated trajectories. The proposed jDMD method immediately learns a model good enough for control with just a couple hundred dynamics samples (2 swing-up trajectories).



(b) Training time for cartpole models as a function of training samples. The training time complexity is approximately linear.

4.1.3 MPC Tracking Performance

To train the bilinear model for the swing-up task, we generated 50 training swing-up trajectories using ALTRO, an open-source nonlinear trajectory optimization solver [19, 20], on the nominal cartpole model. Each trajectory was 5 seconds long and sampled at 20 Hz. A linear MPC controller was then used to track the swing-up trajectories on the simulated system, resulting in significant tracking error due to the model mismatch. After learning both models, a linear MPC policy was used to track the original swing-up trajectory generated by ALTRO. The MPC policy linearized the dynamics about the reference trajectory, used a quadratic penalty on deviations from the reference, and was solved using Riccati recursion for a horizon of 41 time steps (2 seconds). To successfully stabilize the system at the upward equilibrium, the MPC policy extrapolated the terminal state and control reference indefinitely. For each of the bilinear models, the Jacobians were projected back onto space of the original dynamics, resulting in an MPC policy that was just as efficient to solve online as the nominal model (all controllers ran at several thousand Hertz without much performance tuning).

The tracking error, defined as the average L2 norm of the error between the reference trajectory and the actual trajectory of the simulated system for 10 test trajectories not included in the training data, was recorded after training both the eDMD and jDMD models with an increasing number of trajectories. The results in Figure 2a clearly show that jDMD produces a high-quality model with extremely few samples, whereas eDMD—even after many training samples—never achieves

the same level of performance. The lack of progress with increasing samples is likely a result of poor variety in the training data: after enough samples both methods effectively learn all the information that can be learned from the distribution from which the training data is sampled. This example highlights the value of adding even just a little derivative information to a data-driven approach: it dramatically increases sample efficiency while also improving the quality of the learned model, especially when that model is used in optimization-based controllers such as MPC that rely on derivative information. For reference, the training times are shown in Figure 2b. Note that the training algorithms haven't yet been optimized for max performance but reflect a decent first implementation.

References

[1] F. Farshidian, M. Neunert, A. W. Winkler, G. Rey, and J. Buchli. An efficient optimal planning and control framework for quadrupedal locomotion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 93–100. doi:10.1109/ICRA.2017.7989016.

[2] S. Kuindersma, F. Permenter, and R. Tedrake. An efficiently solvable quadratic program for stabilizing dynamic locomotion. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 2589–2594, sep 2014. ISSN 10504729. doi:10.1109/ICRA.2014.6907230.

[3] M. Bjelonic, R. Grandia, O. Harley, C. Galliard, S. Zimmermann, and M. Hutter. Whole-Body MPC and Online Gait Sequence Generation for Wheeled-Legged Robots. *IEEE International Conference on Intelligent Robots and Systems*, pages 8388–8395, 2021. ISSN 21530866. doi:10.1109/IROS51168.2021.9636371.

[4] J. K. Subosits and J. C. Gerdes. From the racetrack to the road: Real-time trajectory replanning for autonomous driving. *IEEE Transactions on Intelligent Vehicles*, 4(2):309–320, jun 2019. ISSN 23798858. doi:10.1109/TIV.2019.2904390.

[5] N. Karnchanachari, M. I. Valls, S. David Hoeller, and M. Hutter. Practical Reinforcement Learning For MPC: Learning from sparse objectives in under an hour on a real robot. *Proceedings of Machine Learning Research*, pages 1–14, 2020. doi:10.3929/ETHZ-B-000404690. URL https://doi.org/10.3929/ethz-b-000404690.

[6] D. . Hoeller, F. . Farshidian, M. Hutter, F. Farshidian, and D. Hoeller. Deep Value Model Predictive Control. *Proceedings of the Conference on Robot Learning*, 100:990–1004, 2020. ISSN 2640-3498. doi:10.3929/ETHZ-B-000368961. URL https://doi.org/10.3929/ethz-b-000368961.

[7] Z. Li, X. Cheng, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath. Reinforcement Learning for Robust Parameterized Locomotion Control of Bipedal Robots. *Proceedings - IEEE International Conference on Robotics and Automation*, 2021-May:2811–2817, 2021. ISSN 10504729. doi:10.1109/ICRA48506.2021.9560769.

[8] A. Meduri, P. Shah, J. Viereck, M. Khadiv, I. Havoutis, and L. Righetti. BiConMP: A Nonlinear Model Predictive Control Framework for Whole Body Motion Planning. jan 2022. doi:10.48550/arxiv.2201.07601. URL https://arxiv.org/abs/2201.07601v1.

[9] D. Bruder, X. Fu, and R. Vasudevan. Advantages of Bilinear Koopman Realizations for the Modeling and Control of Systems with Unknown Dynamics. *IEEE Robotics and Automation Letters*, 6(3):4369–4376, 2021. ISSN 23773766. doi:10.1109/LRA.2021.3068117.

[10] M. Korda and I. Mezić. Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. *Automatica*, 93:149–160, 2018. doi:10.1016/j.automatica.2018.03.046. URL https://doi.org/10.1016/j.automatica.2018.03.046.

[11] C. Folkestad, D. Pastor, and J. W. Burdick. Episodic Koopman Learning of Nonlinear Robot Dynamics with Application to Fast Multirotor Landing. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 9216–9222, may 2020. ISSN 10504729. doi: 10.1109/ICRA40945.2020.9197510.

[12] C. Folkestad, D. Pastor, I. Mezic, R. Mohr, M. Fonoberova, and J. Burdick. Extended Dynamic Mode Decomposition with Learned Koopman Eigenfunctions for Prediction and Control. *Proceedings of the American Control Conference*, 2020-July:3906–3913, jul 2020. ISSN 07431619. doi:10.23919/ACC45564.2020.9147729.

[13] H. J. Suh and R. Tedrake. The Surprising Effectiveness of Linear Models for Visual Foresight in Object Pile Manipulation. *Springer Proceedings in Advanced Robotics*, 17:347–363, feb 2020. ISSN 25111264. doi:10.48550/arxiv.2002.09093. URL https://arxiv.org/abs/2002.09093v3.

[14] U. Fasel, E. Kaiser, J. N. Kutz, B. W. Brunton, and S. L. Brunton. SINDy with Control: A Tutorial, 2021. ISSN 25762370. URL https://github.com/urban-fasel/SEIR.

[15] J. L. Proctor, S. L. Brunton, and J. Nathan Kutz. Generalizing koopman theory to allow for inputs and control. *SIAM Journal on Applied Dynamical Systems*, 17(1):909–930, 2018. ISSN 15360040. doi:10.1137/16M1062296. URL http://www.siam.org/journals/siads/17-1/M106229.html.

[16] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley. A Data–Driven Approximation of the Koopman Operator: Extending Dynamic Mode Decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, dec 2015. ISSN 14321467. doi:10.1007/S00332-015-9258-5/FIGURES/14. URL https://link.springer.com/article/10.1007/s00332-015-9258-5.

[17] D. Baraff. Linear-time dynamics using Lagrange multipliers. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 137–146. ACM.

[18] C. Folkestad and J. W. Burdick. Koopman NMPC: Koopman-based Learning and Nonlinear Model Predictive Control of Control-affine Systems. In *Proceedings - IEEE International Conference on Robotics and Automation*, volume 2021-May, pages 7350–7356. Institute of Electrical and Electronics Engineers Inc., 2021. ISBN 9781728190778. doi: 10.1109/ICRA48506.2021.9562002.

[19] T. A. Howell, B. E. Jackson, and Z. Manchester. ALTRO: A Fast Solver for Constrained Trajectory Optimization. *IEEE International Conference on Intelligent Robots and Systems*, pages 7674–7679, nov 2019. ISSN 21530866. doi:10.1109/IROS40897.2019.8967788.

[20] B. E. Jackson, T. Punnoose, D. Neamati, K. Tracy, R. Jitosho, and Z. Manchester. ALTRO-C: A Fast Solver for Conic Model-Predictive Control; ALTRO-C: A Fast Solver for Conic Model-Predictive Control. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021. doi:10.1109/ICRA48506.2021.9561438. URL https://github.com/.