Data-Efficient Model Learning for Control with Jacobian-Regularized Dynamic Mode Decomposition

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

Abstract: We present a novel algorithm for learning Koopman models of controlled nonlinear dynamical systems from data based on Dynamic-Mode Decomposition (DMD). Our approach, Jacobian-Regularized DMD (JDMD), offers dramatically improved sample efficiency over existing DMD-based algorithms by leveraging Jacobian information from an approximate prior model of the system. We demonstrate JDMD's ability to quickly learn bilinear Koopman dynamics representations across several realistic examples in simulation, including a quadrotor and a perching fixed-wing aircraft. In all cases, we show that the models learned by JDMD provide superior tracking and generalization performance in a model-predictive control framework when compared to both the approximate prior models used in training and models learned by standard extended DMD.

1 Introduction

In recent years, both model-based optimal-control [1, 2, 3, 4] and data-driven reinforcement-learning methods [5, 6, 7] have demonstrated impressive successes on complex, nonlinear robotic systems. However, both of approaches suffer from inherent drawbacks: Data-driven methods often require extremely large amounts of data and fail to generalize outside of the domain or task on which they were trained. On the other hand, model-based methods require an accurate model of the system to achieve good performance. In many cases, high-fidelity models can be too difficult to construct from first principles or too computationally expensive to be of practical use. However, low-order approximate models that can be evaluated cheaply at the expense of controller performance are often available. With this in mind, we seek a middle ground between model-based and data-driven approaches in this work.

We propose a method for learning bilinear Koopman models TODO: cite stuff of nonlinear dynamical systems for use in model-predictive control that leverages information from an approximate prior dynamics model of the system in the training process. Our new algorithm builds on extended Dynamic Mode Decomposition (eDMD), which learns Koopman models from trajectory data [8, 9, 10, 11, 12], by adding a derivative regularization term based on derivatives computed from a prior model. We show that this new algorithm, Jacobian-regularized Dynamic Mode Decomposition (JDMD), can learn models with dramatically fewer samples than eDMD, even when the prior model differs significantly from the true dynamics of the system. We also demonstrate the effectiveness of these learned models in a model-predictive control (MPC) framework. The result is a fast, robust, and sample-efficient pipeline for quickly training a model that can outperform previous Koopman-based MPC approaches as well as purely model-based controllers that do not leverage data collected from the actual system.

Our work is most closely related to the recent work of Folkestad et. al. [11, 13, 14], which learn bilinear models and apply nonlinear model-predictive control directly on the learned bilinear dy-

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

namics. Other recent works have combined linear Koopman models with model-predictive control [10] and Lyapunov control techniques with bilinear Koopman [15]. Our contributions are:

- A novel extension to extended dynamic mode decomposition, called JDMD, that incorporates gradient information from an approximate analytic model
- A recursive, batch QR algorithm for solving the least-squares problems that arise when learning bilinear dynamical systems using DMD-based algorithms, including JDMD and eDMD
- A simple linear MPC control technique for learned bilinear control systems that is computationally efficient and, when combined with JDMD, requires very little training data to achieve good performance

The remainder of the paper is organized as follows: In Section 2 we provide some background on the application of Koopman operator theory to controlled dynamical systems and review some related works. Section 3 then describes the proposed JDMD algorithm. In Section 4 we outline a memory-efficient technique for solving the large, sparse linear least-squares problems that arise when applying JDMD and other DMD-based algorithms. Next, in Section 6.3, we propose an efficient model-predictive control technique that utilizes the learned bilinear models produced by JDMD. Section 6 then provides simulation results and analysis of the proposed algorithm applied to control tasks on a cartpole, a quadrotor, and a small foam airplane, all subject to significant model mismatch. In Section 7 we discuss the limitations of our approach, followed by some concluding remarks in Section 8.

2 Background and Related Work

2.1 Koopman Operator Theory

The theoretical underpinnings of the Koopman operator and its application to dynamical systems has been extensively studied [16, 17, 9, 18]. Rather than describe the theory in detail, we highlight the key concepts employed by the current work and refer the reader to the existing literature on Koopman theory for further details.

We start by assuming a controlled, nonlinear, discrete-time dynamical system,

$$x^+ = f(x, u), \tag{1}$$

where $x \in \mathcal{X} \subseteq \mathbb{R}^{N_x}$ is the state vector, $u_k \in \mathbb{R}^{N_u}$ is the control vector, and x^+ is the state at the next time step. The key idea behind the Koopman operator is that the nonlinear finite-dimensional dynamics (1) can be represented *exactly* by an infinite-dimensional bilinear system of the form,

$$y^+ = Ay + Bu + \sum_{i=1}^{m} u_i C_i y = g(y, u), \tag{2}$$

where $y = \phi(x)$ is a nonlinear mapping from the finite-dimensional state space $\mathcal{X}$ to the infinite-dimensional Hilbert space of *observables* $\mathcal{Y}$. In practice, we approximate (2) by restricting $\mathcal{Y}$ to be a finite-dimensional vector space, in which case ϕ becomes a finite-dimensional nonlinear function of the state variables that must be chosen by the user.

Intuitively, ϕ "lifts" our state x into a higher dimensional space $\mathcal{Y}$ where the dynamics are approximately (bi)linear, effectively trading dimensionality for (bi)linearity. Similarly, we can perform the inverse operation by projecting a lifted state y back into the original state space $\mathcal{X}$. In this work, we will assume that ϕ is constructed in such a way that this inverse mapping is linear:

$$x = Gy \tag{3}$$

2.2 Extended Dynamic Mode Decomposition

TODO: This section is too terse. A little more detail in the text about what's going on would help, e.g. "data matrices consisting of all control inputs and lifted states..." Explain a little more in words and lean a little less on people parsing all of the math.

A lifted bilinear system of the form (2) can be learned from P samples of the system dynamics (x_j^+, x_j, u_j) using extended Dynamic Mode Decomposition (eDMD) [18, 13]. We concatenate all of the model coefficient matrices as follows:

$$E = [A \quad B \quad C_1 \quad \dots \quad C_m] \in \mathbb{R}^{N_y \times N_z}, \tag{4}$$

by solving the following linear least-squares problem: TODO: typo on $Y_{1:P}^2$ below?

$$\underset{E}{\text{minimize}} \left\| E Z_{1:P} - Y_{1:P}^2 \right\|_2^2 \tag{5}$$

where $Z_{1:P} \in \mathbb{R}^{N_z \times P}$ and $Y_{1:P}^+ \in \mathbb{R}^{N_y \times P}$ are the data matrices

$$Z_{1:P} = \begin{bmatrix} y_1 & y_2 & \dots & y_P \\ u_1 & u_2 & \dots & u_P \\ u_{1,1}y_1 & u_{2,1}y_2 & \dots & u_{P,1}y_P \\ \vdots & \vdots & \ddots & \vdots \\ u_{1,m}y_1 & u_{2,m}y_2 & \dots & u_{P,m}y_P \end{bmatrix}, \quad Y_{1:P}^+ = \begin{bmatrix} y_1^+ & y_2^+ & \dots & y_P^+ \end{bmatrix}, \tag{6}$$

and $N_z = N_y + N_u + N_y \cdot N_u$.

3 Jacobian-Regularizated Dynamic Mode Decomposition

TODO: Let's standardize on "Jacobian-Regularized" instead of "Jacobian-penalized" and "JDMD" instead of jDMD

We now present JDMD as a straightforward adaptation of the original eDMD algorithm described in Section 2.2. Given P samples of the dynamics (x_i^+, x_i, u_i), and an approximate discrete-time dynamics model,

$$x^+ = \tilde{f}(x, u), \tag{7}$$

we can evaluate the Jacobians of our approximate model $\tilde{f}$ at each of the sample points: $\tilde{A}_i = \frac{\partial \tilde{f}}{\partial x}, \tilde{B}_i = \frac{\partial \tilde{f}}{\partial u}$. After choosing a nonlinear mapping $\phi : \mathbb{R}^{N_x} \mapsto \mathbb{R}^{N_y}$ our goal is to find a bilinear dynamics model (2) that matches the Jacobians of our approximate model, while also matching our dynamics samples. We accomplish this by penalizing differences between the Jacobians of our learned bilinear model with respect to the original states x and controls u, and the Jacobians we expect from our analytical model. These *projected Jacobians* are calculated by differentiating through the *projected dynamics*:

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

Differentiating (8) with respect to x and u gives us TODO: This notation isn't super clear. Can we try using some different letters instead of hats on A and B below? Also is there a typo on the $\bar{A}$ below?

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

$$\bar{B}_j = \frac{\partial \hat{f}}{\partial u}(x_j, u_j) = G\Big(B + [C_1 x_j \quad \dots \quad C_m x_j]\Big) = GE\hat{B}(x_j, u_j) = GE\hat{B}_j \tag{9b}$$

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

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

Figure 1: Airplane perching trajectory, a high angle-of-attack maneuver that minimizes velocity at the goal position

We then solve the following linear least-squares problem:

$$\underset{E}{\text{minimize}} \;\; (1-\alpha)\left\|EZ_{1:P} - Y_{1:P}^{+}\right\|_2^2 + \alpha \sum_{j=1}^{P} \left(\left\| GE\hat{A}_j - \tilde{A}_j \right\|_2^2 + \left\| GE\hat{B}_j - \tilde{B}_j \right\| \right) \tag{11}$$

The resulting linear least-squares problem has $(N_y + N_x^2 + N_x \cdot N_u) \cdot P$ rows and $N_y \cdot N_z$ columns. Given that the number of rows in this problem grows quadratically with the state dimension, solving this problem can be challenging from a computational perspective. In the following section, we propose an algorithm for solving these problems without needing to move to a distributed-memory setup in order to solve these large linear systems. The proposed method also provides a straightforward way to approach incremental updates to the bilinear system, where the coefficients could be efficiently learned "live" while the robot gathers data by moving through it's environment.

4 Efficient Recursive Least Squares

In its canonical formulation, a linear least squares problem can be represented as the following unconstrained optimization problem:

$$\min_x \|Fx - d\|_2^2. \tag{12}$$

We assume F is a large, sparse matrix and that solving it directly using a QR or Cholesky decomposition requires too much memory for a single computer. While solving (12) using an iterative method such as LSMR [19] or LSQR [20] is possible, we find that these methods do not work well in practice for solving (11) due to ill-conditioning. Standard recursive methods for solving these problems are able to process the rows of the matrices sequentially to build a QR decomposition of the full matrix, but also tend to suffer from ill-conditioning [21, 22, 23].

To overcome these issues, we propose an alternative recursive method based. We solve (12) by dividing up rows of F into batches:

$$F^T F = F_1^T F_1 + F_2^T F_2 + \ldots + F_N^T F_N. \tag{13}$$

The main idea is to maintain and update an upper-triangular Cholesky factor U_i of the first i terms of the sum (13). Given U_i, we can calculate U_{i+1} using the QR decomposition, as shown in [24]:

$$U_{i+1} = \sqrt{U_i^T U_i + F_{i+1}^T F_{i+1}} = \mathrm{QR}_{\mathrm{R}}\left(\begin{bmatrix} U_i \\ F_{i+1} \end{bmatrix} \right), \tag{14}$$

where QR_{R} returns the upper triangular matrix R from the QR decomposition. For an efficient implementation, this function should be an "economy" or "Q-less" QR decomposition [**?**], since the Q matrix is never needed.

We also handle regularization of the normal equations, equivalent to adding Tikhonov regularization to the original least squares problem TODO: cite something, during the base case of our recursion. If we want to add an L2 regularization with weight λ, we calculate U_1 as:

$$U_1 = \mathrm{QR}_{\mathrm{R}}\left(\begin{bmatrix} F_1 \\ \sqrt{\lambda} I \end{bmatrix} . \right). \tag{15}$$

5 Projected Bilinear MPC

We propose a simple approach to model-predictive control for the bilinear systems learned using either classic eDMD or the proposed JDMD approach. The key idea is to use the projected Jacobians $\bar{A}$ and $\bar{B}$ in (9), effectively reducing the problem to a standard linear MPC problem in the original state space instead of the larger, lifted one. In the all of the examples in the following section, our MPC controller solves the following convex Quadratic Program (QP):

$$\begin{aligned} \underset{x_{1:N},\, u_{1:N-1}}{\text{minimize}} \quad & \frac{1}{2} x_N^T Q_N x_N + \frac{1}{2} \sum_{k=1}^{N-1} x_k^T Q_k x_k + u_k^T R_k u_k \\ \text{subject to} \quad & x_{k+1} = \bar{A}_k x_k + \bar{B}_k u_k + d_k, \\ & x_1 = x_{\text{init}} \end{aligned} \tag{16}$$

where here we define x and u to be the "delta" from the reference trajectory TODO: how about just writing Δx and Δu explicitly above? $\bar{x}_{1:N}$, $\bar{u}_{1:N-1}$. The affine dynamics term $d_k = f(\bar{x}_k, \bar{u}_k) - \bar{x}_{k+1}$ allows for dynamically infeasible reference trajectories. The projected Jacobians can be efficiently calculated from the bilinear dynamics either offline or online, and since the problem dimension is the same size as the linear MPC problem for the original dynamics, it is no more expensive to compute. This formulation also makes it trivial to enforce additional control or path constraints, and avoids the need to regularize or otherwise constrain the lifted states.

6 Results

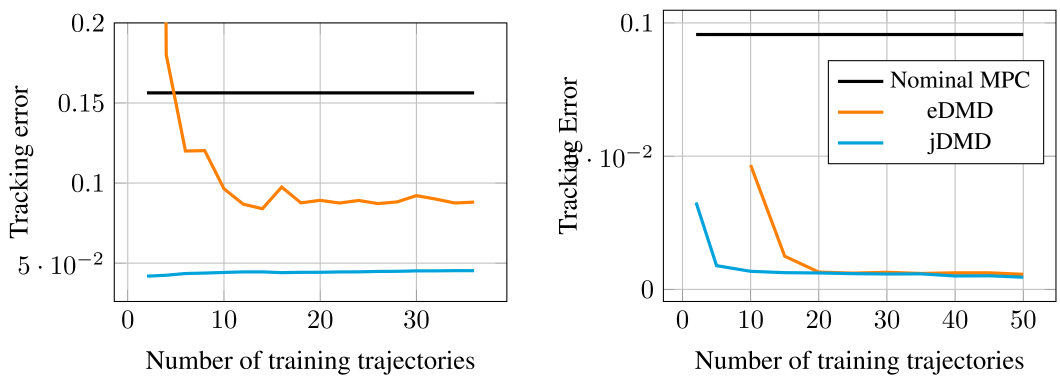
The following sections provide various numerical analyses of the proposed algorithm. In lieu of an actual hardware experiment (left for future work), for each simulated system we specify two models: a *nominal* model which is a simplified model approximating the *true* model, which contains both parametric and non-parametric model error from the nominal model. This true model is used exclusively for simulating the system.

All models were trained by simulating the "true" 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 introduced by Folkestad and Burdick [13]. All continuous dynamics were discretized with an explicit fourth-order Runge Kutta integrator. We organize the following results section by topic, we briefly describe the three simulated models used to produce the results in the paragraphs below.

The *true* cartpole model included a $\tanh$ model of Coulomb friction between the cart and the floor, viscous damping at both joints, and a control dead band that was not included in the *nominal* model. Additionally, the mass of the cart and pole model were altered by 20% and 25% with respect to the nominal model, respectively. The following nonlinear mapping was used when learning the bilinear models: $\phi(x) = [1,\, x,\, \sin(x),\, \cos(x),\, \sin(2x),\, \sin(4x),\, T_2(x),\, T_3(x),\, T_4(x)] \in \mathbb{R}^{33}$, where $T_i(x)$ is a Chebyshev polynomial of the first kind of order i. All reference trajectories for the swing up task were generated using ALTRO [24, 25].

We also include 2 quadrotor systems: a 3-DOF, planar model and a 6-DOF, full quadrotor model. For both systems, the *true* model includes aerodynamic drag terms not included in the *nominal* model, as well as parametric error of about 5% on the system properties (e.g. mass, rotor arm length, etc.). The planar model was trained using a nonlinear mapping of $\phi(x) = [1,\, x,\, \sin(x),\, \cos(x),\, \sin(2x),\, T_2(x)] \in \mathbb{R}^{25}$ while the full quadrotor model was train using a nonlinear mapping of $\phi(x) = [1,\, x,\, T_2(x),\, \sin(p),\, \cos(p),\, R^T v,\, v^T R R^T v,\, p \times v,\, p \times \omega,\, \omega \times \omega] \in \mathbb{R}^{44}$, where p is the quadrotor's position, v and ω are the translational and angular velocities respectively, and R is the Rotation matrix. Reference trajectories were produced by nominal MPC tracking controllers attempting to track infeasible, point-to-point, linear trajectories with various initial conditions.

Our most high-fidelity model is an airplane model with lift and drag coefficients fit from rigorous wind-tunnel testing [26]. The coefficients are valid for high angle-of-attach maneuvers. We produce a high-angle-of-attack (up to $40°$) manuever by solving for "perching" trajectories with ALTRO that encourage minimal velocity subject to a constraint on the final position (see Figure 1). These trajectories are tracked using a QP-based linear MPC solver, with bounds on the state and control



(a) Cartpole

(b) Airplane

Figure 2: MPC tracking error vs training trajectories for both the cartpole (left) and airplane (right). Tracking error is defined as the average L2 error over all the test trajectories between the reference and simulated trajectories.

deviations at each time step. The nominal model used a simple flat-plate wing model with linear lift and quadratic drag coefficient models. To train the bilinear models, we used a 68-dimensional nonlinear mapping composed of dynamics terms such as the rotation matrix (expressed in terms of a Modified Rodriguez Parameter), powers of the angle of attack and side slip angle, the body frame velocity, various cross products with the angular velocity, and some 3rd and 4th order Chebyshev polynomials.

The code for the experiments is located at TODO: include after review.

6.1 Sample Efficiency

We highlight the sample efficiency of the proposed algorithm in Figure 2. For both the cartpole swing up and the airplane perch trajectory tracking tasks, the proposed method achieves better tracking than the nominal MPC controller with just two sample trajectories, and performs uniformly better than eDMD on both trajectory tracking tasks. To achieve comparable performance on the perching task, eDMD requires about 4x the number of samples (20 vs 5) compared to the proposed approach. TODO: add lines for "lifted" MPC approach?

6.2 Generalization

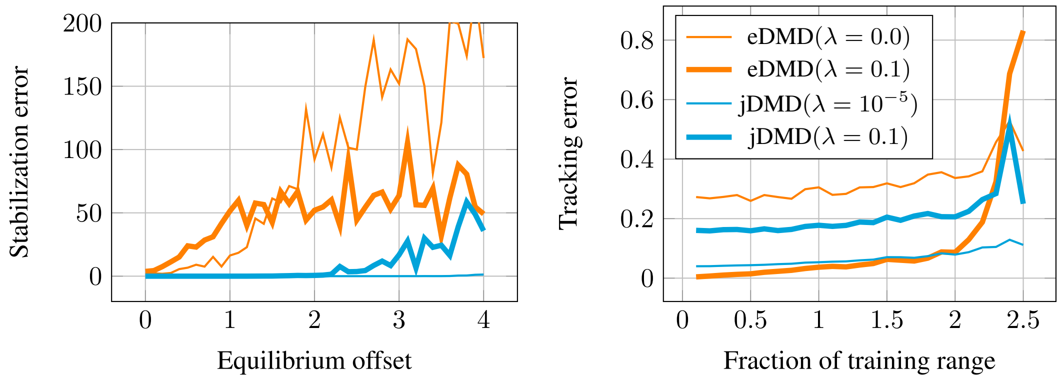
We demonstrate the generalizability of the proposed method to tasks outside of its training domain in Figure 4. In both the planar quadrotor stabilization (Figure 3a) and trajectory tracking (Figure 3b) tasks, we trained the models by sampling uniformly from a given window of offsets, centered about the origin. To test the generalizability of the methods we increased the relative size of the window from which the test data was sampled, e.g. if the initial lateral position was trained on data in the interval $[-1.5, +1.5]$, we sampled the test initial condition from the window $[-\gamma 1.5, +\gamma 1.5]$. As shown in the results, while the performance of the proposed algorithm remains relatively constant even when $\gamma = 2.5$, whereas the classic eDMD approach looses performance and fails to generalize at all for the stabilization task using and LQR controller (like due to poor derivative information), and up to $\gamma = 2$ for the tracking task using a linear MPC controller.

In Figure 3a we show the effect of changing the equilibrium position for the planar quadrotor, but keeping the delta initial conditions within the training window. As shown, eDMD doesn't generalize to other equilibrium points, despite the fact that the underlying dynamics are invariant to the equilibrium position. Our proposed approach, however, easily learns this from the derivative information provided by the nominal model.

	nominal MPC	**eDMD**	**jDMD**
Mean Tracking Err.	0.30	0.63	**0.11**
Success Rate	**82%**	18%	80%

Table 1: Performance summary of MPC tracking of 6-DOF quadrotor

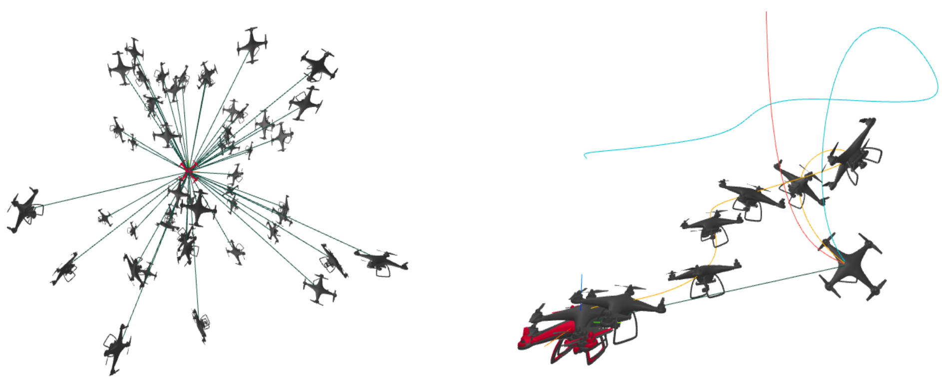
To demonstrate that this generalizability extends to more complex dynamics, we additionally show the tracking capabilities of



(a) LQR stabilization error over increasing equilibrium offset

(b) Tracking error for the quadrotor MPC reference trajectory tracking task.

Figure 3: Generalizability with respect to initial conditions sampled outside of the training domain. The initial conditions are sampled from a uniform distribution, whose limits are determined by a scaling of the limits used for the training distribution. A training range fraction greater than 1 indicates the distribution range is beyond that used to generate the training trajectories. The thick lines represent the algorithm with a heavy regularization parameter.



(a) Sampled point-to-point trajectories and initial conditions for test tracking MPC of 6-DOF quadrotor.

(b) Generated MPC tracking trajectories for nominal MPC (blue), eDMD (red), and jDMD (orange) for an example reference.

Figure 4: Generalizability with respect to initial conditions sampled outside of the training domain. The initial conditions are sampled from a uniform distribution, whose limits are determined by a scaling of the limits used for the training distribution. A training range fraction greater than 1 indicates the distribution range is beyond that used to generate the training trajectories. The thick lines represent the algorithm with a heavy regularization parameter.

infeasible, point-to-point trajectories for the 6-DOF quadrotor with full attitude dynamics. As seen in Figure 4a, the generated reference trajectories have a wide scope in sampling despite being sparse in nature. As shown in Table 2, jDMD has the best tracking performance and closely matches nominal MPC's ability to successfully reach the equilibrium. Meanwhile, eDMD has significantly worse performance in tracking and succeeding in stabilizing about the goal state. Despite the aggressive starting position, jDMD is still able to successfully account for the attitude dynamics, reach the goal state, and stabilize, as seen in Figure 4b.

6.3 Lifted versus Projected MPC

Friction (μ)	0.0	0.1	0.2	0.3	0.4	0.5	0.6
Nominal	✓	✓	✗	✗	✗	✗	✗
eDMD	3	19	6	14	✗	✗	✗
jDMD	2	2	2	2	3	7	12

Table 3: Training trajectories required to stabilize the cartpole with the given friction coefficient

MPC	**eDMD**	**jDMD**
Lifted	17	15
Projected	18	**2**

Table 2: Training trajectories required to beat nominal MPC

We performed a simple experiment to highlight the value of the proposed "projected" MPC, outlined in . We trained eDMD and jDMD models with an increasing number of training trajectories, and recorded the first sample size at which the "lifted" and "projected" MPC controllers consistently stabilized the system (i.e. stabilized 95% of the test initial conditions for the cartpole system for that sample size and subsequent ones). The results are summarized in Table 2. The results qualitatively show what we quantitatively observed while training and testing these various examples: the projected MPC approach usually required far fewer samples to "train" and usually had better performance than it's "lifted" counterpart that used the bilinear "lifted" dynamics. This was especially pronounced when combined with the proposed jDMD approach, which makes sense given that the approach explicitly encourages these Jacobians to match the analytical ones, so quickly converge to reasonable values with just a few training examples.

6.4 Sensitivity to Model Mismatch

While we've introduced a significant mount of model mismatch in all of the examples so far, a natural argument against model-based methods is that they're only as good as your model is at capturing the salient dynamics of the system. We investigated the effect of increasing model mismatch by incrementally increasing the Coulomb friction coefficient between the cart and the floor for the cartpole stabilization task (recall the nominal model assumed zero friction). The results are shown in Figure 3. As expected, the number of training trajectories required to find a good stabilizing controller increases for the proposed approach. We achieved the results above by setting $\alpha = 0.01$, corresponding to a decreased confidence in our model, thereby placing greater weight on the experimental data. The standard eDMD approach always required more samples, and was unable to find a good enough model above friction values of 0.4. While this could likely be remedied by adjusting the nonlinear mapping ϕ, the proposed approach works well with the given bases. Note that the nominal MPC controller failed to stabilize the system above friction values of 0.1, so again, we demonstrate that we can improve MPC performance substantially with just a few training samples by combining analytical gradient information and data sampled from the true dynamics.

7 Limitations

As with most data-driven techniques, it is hard to definitively declare that the proposed method will increase performance in all cases. It is possible that having an extremely poor analytical model may hurt rather than help the training process. However, we found that even when the α parameter is extremely small (placing little weight on the Jacobians during the learning process), it still dramatically improves the sample efficiency. It is also quite possible that the performance gaps between eDMD and jDMD shown here can be reduced through better selection of basis functions and better training data sets; however, given that the proposed approach converges to eDMD as $\alpha \to 0$, we see no reason to not adopt the proposed methodology as simply tune α based on the confidence of the model and the quantity (and quality) of training data.

8 Conclusion and Future Work

We have presented a simple but powerful extension to eDMD, a model-based method for learning a bilinear representation of arbitrary dynamical systems, that incorporates derivative information from an analytical model. When combined with a simple linear MPC policy that projects the learned dynamics back into the original state space, we have shown that the resulting pipeline can dramatically increase sample efficiency, often improving over a nominal MPC policy with just a few sample

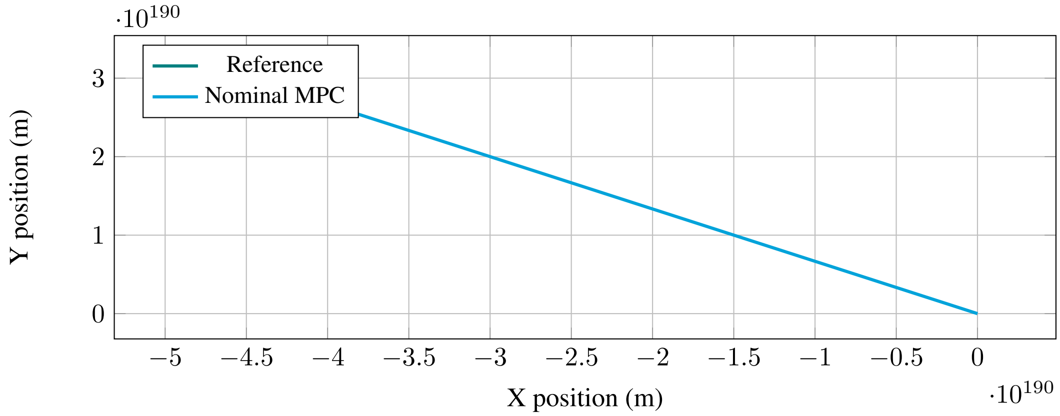


Figure 5: Effect of increasing model mismatch. Displays the number of training trajectories required to consistently stabilize the cartpole system, given an increasing friction coefficient, which the nominal model does not include at all. An entry of 0 signifies that a stabilizing controller wasn't found with less than 100 training trajectories. The nominal MPC controller failed to stabilize once the friction coefficient went above 0.1.

trajectories. Substantial areas for future work remain: most notably testing the proposed pipeline on hardware. Additional directions include lifelong learning or adaptive control applications, combining simulated and real data through the use of modern differentiable physics engines [27, 28] residual dynamics learning, as well as the development of specialized numerical methods for solving nonlinear optimal control problems using the learned bilinear dynamics.

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. pages 2589–2594. ISSN 9781479936854. 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. pages 8388–8395. ISSN 9781665417143. 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. 4(2):309–320. 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. pages 1–14. 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. 100:990–1004. 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. 2021-May:2811–2817. ISSN 9781728190778. 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. 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. 6(3):4369–4376. doi:10.1109/LRA.2021.3068117.

[10] M. Korda and I. Mezi. Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. 93:149–160. 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. pages 9216–9222. ISSN 9781728173955. doi:10.1109/ICRA40945.2020.9197510.

[12] H. J. Suh and R. Tedrake. The Surprising Effectiveness of Linear Models for Visual Foresight in Object Pile Manipulation. 17:347–363. doi:10.48550/arxiv.2002.09093. URL https://arxiv.org/abs/2002.09093v3.

[13] 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. ISBN 978-1-72819-077-8. doi:10.1109/ICRA48506.2021.9562002.

[14] C. Folkestad, S. X. Wei, and J. W. Burdick. Quadrotor Trajectory Tracking with Learned Dynamics: Joint Koopman-based Learning of System Models and Function Dictionaries. URL http://arxiv.org/abs/2110.10341.

[15] A. Narasingam, J. Sang, and I. Kwon. Data-driven feedback stabilization of nonlinear systems: Koopman-based model predictive control. pages 1–12.

[16] SINDy with Control: A Tutorial. URL https://github.com/urban-fasel/SEIR.

[17] J. L. Proctor, S. L. Brunton, and J. Nathan Kutz. Generalizing koopman theory to allow for inputs and control. 17(1):909–930. doi:10.1137/16M1062296. URL http://www.siam.org/journals/siads/17-1/M106229.html.

[18] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley. A DataDriven Approximation of the Koopman Operator: Extending Dynamic Mode Decomposition. 25(6):1307–1346. doi:10.1007/S00332-015-9258-5/FIGURES/14. URL https://link.springer.com/article/10.1007/s00332-015-9258-5.

[19] D. C.-L. Fong and M. Saunders. LSMR: An Iterative Algorithm for Sparse Least-Squares Problems. 33(5):2950–2971. ISSN 1064-8275. doi:10.1137/10079687X. URL https://epubs.siam.org/doi/abs/10.1137/10079687X.

[20] C. C. Paige and M. A. Saunders. LSQR: An Algorithm for Sparse Linear Equations and Sparse Least Squares. 8(1):43–71. ISSN 0098-3500, 1557-7295. doi:10.1145/355984.355989. URL https://dl.acm.org/doi/10.1145/355984.355989.

[21] P. Strobach. Recursive Least-Squares Using the QR Decomposition. In P. Strobach, editor, *Linear Prediction Theory: A Mathematical Basis for Adaptive Systems*, Springer Series in Information Sciences, pages 63–101. Springer. ISBN 978-3-642-75206-3. doi:10.1007/978-3-642-75206-3_4. URL https://doi.org/10.1007/978-3-642-75206-3_4.

[22] A. Sayed and T. Kailath. *Recursive Least-Squares Adaptive Filters*, volume 20094251 of *Electrical Engineering Handbook*, pages 1–40. CRC Press. ISBN 978-1-4200-4606-9 978-1-4200-4607-6. doi:10.1201/9781420046076-c21. URL http://www.crcnetbase.com/doi/abs/10.1201/9781420046076-c21.

[23] A. Ghirnikar and S. Alexander. Stable recursive least squares filtering using an inverse QR decomposition. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 1623–1626 vol.3. doi:10.1109/ICASSP.1990.115736.

[24] T. A. Howell, B. E. Jackson, and Z. Manchester. ALTRO: A Fast Solver for Constrained Trajectory Optimization. pages 7674–7679. ISSN 9781728140049. doi:10.1109/IROS40897.2019.8967788.

[25] 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. ISSN 9781728190778. doi:10.1109/ICRA48506.2021.9561438. URL https://github.com/.

[26] Z. Manchester, J. Lipton, R. Wood, and S. Kuindersma. A Variable Forward-Sweep Wing Design for Enhanced Perching in Micro Aerial Vehicles. In *AIAA Aerospace Sciences Meeting*. URL https://rexlab.stanford.edu/papers/Morphing_Wing.pdf.

[27] T. A. Howell, S. Le Cleac', J. Z. Kolter, M. Schwager, and Z. Manchester. Dojo: A Differentiable Simulator for Robotics.

[28] E. Todorov, T. Erez, and Y. Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. doi: 10.1109/IROS.2012.6386109.