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

Anonymous Author(s)
Affiliation
Address
email

Abstract: We present a data-efficient algorithm for learning models for model-predictive control (MPC). Our approach, Jacobian-Regularized DMD (JDMD), offers improved sample efficiency over traditional Koopman approaches based on Dynamic-Mode Decomposition (DMD) by leveraging Jacobian information from an approximate prior model of the system, and improved tracking performance over traditional model-based MPC. We demonstrate JDMD's ability to quickly learn bilinear Koopman dynamics representations across several realistic examples in simulation, including a perching maneuver for a fixed-wing aircraft with an experimentally derived high-fidelity physics model. In all cases, we show that the models learned by JDMD provide superior tracking and generalization performance in the presence of significant model mismatch within a model-predictive control framework, when compared to 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 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 of nonlinear dynamical systems for use in model-predictive control that leverages derivative information from an approximate prior dynamics model of the system in the training process. Given the increased availability of differentiable simulators [8, 9], this approximate derivative information is readily available for many systems of interest. Our new algorithm builds on extended Dynamic Mode Decomposition (EDMD), which learns Koopman models from trajectory data [10, 11, 12, 13, 14], 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 MPC controllers using the approximate analytical model as well models learned using both traditional Koopman approaches and multi-layer perceptrons (MLPs).

Submitted to the 6th Conference on Robot Learning (CoRL 2022). Do not distribute.

While our proposed Koopman-based approached is significantly more sample efficient, we also demonstrate the utility of incorporating gradient information for learning a simple model using a two-layer MLP.

Our work is most closely related to the recent work of Folkestad et. al. [13, 15, 16], which learn bilinear models and apply nonlinear model-predictive control directly on the learned bilinear dynamics. Other recent works have combined linear Koopman models with model-predictive control [12] and Lyapunov control techniques with bilinear Koopman [17]. 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

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. Section 5 then provides simulation results and analysis of the proposed algorithm applied to control tasks on a cartpole, a quadrotor, and a small foam airplane with an experimentally determined aerodynamics model, all subject to significant model mismatch. It also includes a comparison of the current approach to model-learning via a multi-layer perceptron, for the canonical cartpole problem. In Section 6 we discuss the limitations of our approach, followed by some concluding remarks in Section 7.

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 [18, 19, 11, 20, 21]. 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. Assuming the dynamics are control-affine, the nonlinear finite-dimensional system (1) can be represented *exactly* by an infinite-dimensional bilinear system through the Koopman canonical transform [21]. This bilinear Koopman model follows 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, which can be either chosen heurstically based on domain expertise, or learned [22, 23, 24].

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 an "unlifting" operation by projecting a lifted state y back into the original state space $\mathcal{X}$. In this work, since we embed the original state within the nonlinear mapping [15], ϕ is constructed in such a way that this unlifting is linear:

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

We note that our proposed method does not rely on this assumption: any mapping could be used. The problem of finding an optimal mapping is itself a major area of research, and many recent studies have focused on jointly learning both the model and the mapping [22, 23, 25, 26, 24]. While clearly advantageous, learning an optimal embedding is orthogonal to the main focus of the current paper, which focuses on a straightforward way of incorporating analytical derivative information from an approximate model, which is equally applicable whether the embedding function is learned or chosen heuristically. The mappings in the current work are chosen heuristically based on problem insight and experience.

2.2 Extended Dynamic Mode Decomposition

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) [20, 15]. We first define the following data matrices:

$$Z_{1:P} = \begin{bmatrix} y_1 & y_2 & \dots & y_P \\ u_1 & u_2 & \dots & u_P \\ u_{1,1}y_1 & u_{2,1}y_2 & \dots & u_{P,1}y_P \\ \vdots & \vdots & \ddots & \vdots \\ u_{1,m}y_1 & u_{2,m}y_2 & \dots & u_{P,m}y_P \end{bmatrix}, \quad Y_{1:P}^+ = \begin{bmatrix} y_1^+ & y_2^+ & \dots & y_P^+ \end{bmatrix}, \tag{4}$$

We then concatenate all of the model coefficient matrices as follows:

$$E = [A \quad B \quad C_1 \quad \dots \quad C_m] \in \mathbb{R}^{N_y \times N_z}, \tag{5}$$

The model learning problem can then be written as the following linear least-squares problem:

$$\underset{E}{\text{minimize}} \; \left\| E Z_{1:P} - Y_{1:P}^+ \right\|_2^2 \tag{6}$$

3 Jacobian-Regularizated Dynamic Mode Decomposition

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

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

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

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

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

We then solve the following linear least-squares problem:

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

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 Section 4, 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 its 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 [27] or LSQR [28] 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 [29, 30, 31].

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 [32]:

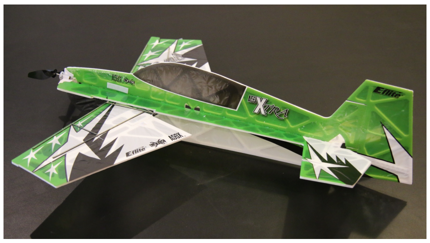
$$U_{i+1} = \sqrt{U_i^T U_i + F_{i+1}^T F_{i+1}} = \text{QR}_\text{R}\left(\begin{bmatrix} U_i \\ F_{i+1} \end{bmatrix} \right), \tag{14}$$

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.

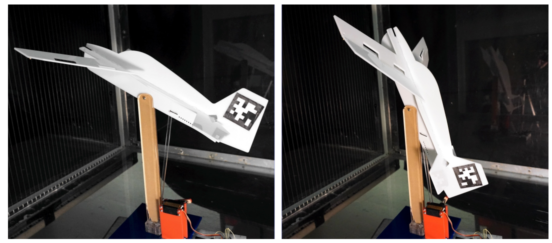
We also handle regularization of the normal equations, equivalent to adding Tikhonov regularization to the original least squares problem, during the base case of our recursion. If we want to add an L2 regularization with weight λ, we calculate U_1 as:

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

(a) Expert perching demonstration, a high angle-of-attack maneuver that minimizes velocity at the goal position with complex, post-stall aerodynamic forces



(b) E-Flite AS3Xtra airplane model used in hardware data collection



(c) Experiment setup configurations for collecting flight data

Figure 1: Complex dynamics of a perching fixed-wing airplane. High-angle-of-attack perching maneuvers (top) require the modeling of complex post-stall aerodynamic effects. The simulated aerodynamic forces were modeled as functions using flight data collected from real-world hardware experiments (bottom).

5 Experimental Results

This section presents the results of several simulation experiments to evaluate the performance of JDMD. For each simulated system we specify two models: a *nominal* model, which is simplified and contains both parametric and non-parametric model error, and a *true* model, which is used exclusively for simulating the system and evaluating algorithm performance.

All models were trained by simulating the "true" system with a nominal 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 20-25 Hz. The bilinear EDMD model was trained using the same approach introduced by Folkestad and Burdick [15]. When applying MPC to the learned Koopman models, the projected Jacobians (9) were used, since this projected system is much more likely to be controllable than the lifted one and reduces the computational complexity back to that of the nominal MPC controller. This results in a nonlinear model in the original state space, which is linearized about the reference trajectory to create a linear MPC controller. All continuous dynamics were discretized with an explicit fourth-order Runge Kutta integrator. Code for all experiments is available at TODO: removed for anonymous review.

5.1 Systems and Tasks

Cartpole: We perform a swing-up task on a cartpole system. The *true* model includes Coulomb friction between the cart and the floor, viscous damping at both joints, and a deadband in the control input that were 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 [32, 33].

Quadrotor: We track point-to-point linear reference trajectories from various initial conditions on both planar and full 3D quadrotor models. For both systems, the *true* model includes aerodynamic drag terms not included in the *nominal* model, as well as parametric er-

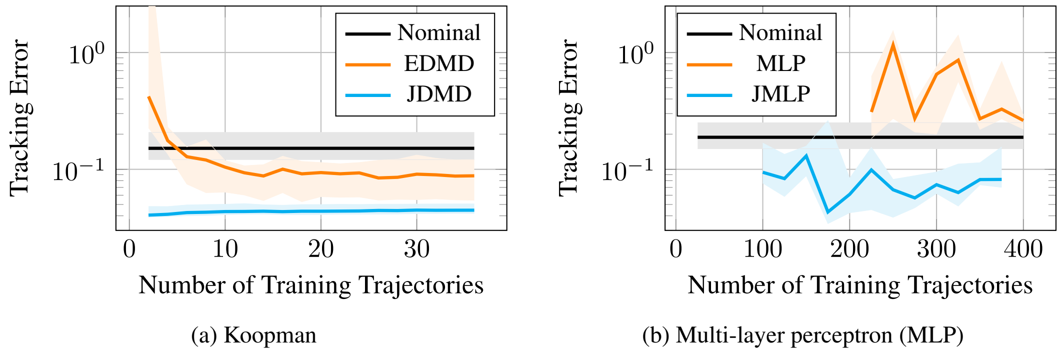


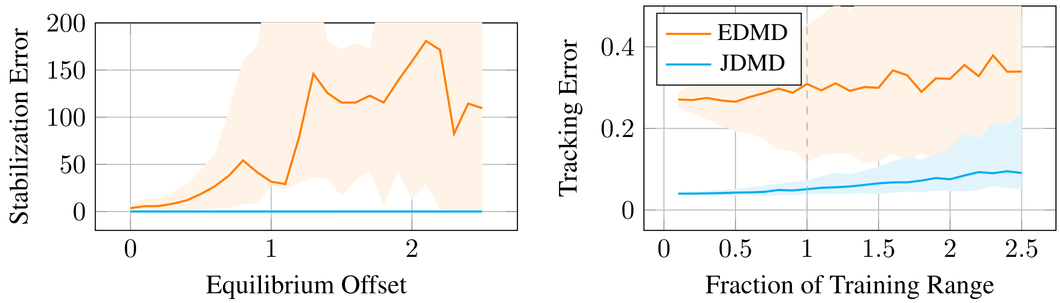
Figure 2: Cartpole swingup MPC tracking error vs training trajectories for Koopman methods (left) and a multi-layer perceptron (right). The sample efficiency of both methods is significantly improved when derivative information is included in the loss function. Note that Koopman approaches require an order of magnitude fewer trajectories to stabilize compared the MLP-based approach. The median error is shown as a thick line, while the shaded regions represent the 5% to 95% percentile bounds on the 10 test trajectories.

ror of roughly 5% on the system parameters (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 trained 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.

Airplane: We perform a post-stall perching maneuver on a high-fidelity model of a fixed-wing airplane. The perching trajectory is produced using trajectory optimization (see Figure 1a) and tracked using MPC. Perching involves flight at high angles of attack, where the aerodynamic lift and drag forces are extremely complex and difficult to model from first principles. We look to previous works where the simulated aerodynamics were fitted using empirical data from in-person, wind-tunnel experiments (see Figure 1b and 1c) before being demonstrated on hardware platforms [34, 35]. The *true* model includes the empirically-modeled, nonlinear flight dynamics [35], while the *nominal* model uses a simple flat-plate wing model with linear lift and quadratic drag coefficient approximations. The bilinear models use a 68-dimensional nonlinear mapping ϕ including 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 of the states.

5.2 Sample Efficiency

We compare the sample efficiency of several algorithms on the cartpole swing-up task in Fig. 2, including a simple two-layer multi-layer perceptron trained using the a loss function equivalent to (11) with $\alpha = 1$ (MLP) and $\alpha \in (0, 1)$ (JMLP). The derivatives of the model with respect to the inputs are calculated automatically using backward propagation of the partial derivatives for usage in the loss function, resulting in second-order derivatives of the $\tanh$ activation functions when calculating the gradient with respect to the model parameters. As shown, the proposed method achieves the best performance overall, and does so with only two training trajectories. In comparison, traditional EDMD requires about 10 iterations to achieve consistent performance, whereas the MLP methods require hundreds of training trajectories. It's also important to note that by applying the proposed approach to an MLP we were able to dramatically improve both the performance and sample efficiency of the MLP-based approach. Similar results were obtained for the airplane perching example (Fig. 5b), where EDMD requires about 4x the number of samples (20 vs 5) compared to the proposed approach.



(a) LQR stabilization error over increasing equilibrium offset for 100 random initial conditions.

(b) MPC Tracking error over increasing scope of test distribution for 50 random initial conditions.

Figure 3: Generalizability with respect to final or initial conditions sampled outside of the training domain, studied on planar quadrotor performing an LQR stabilization (left) and MPC tracking task (right). For the stabilization task, 100 equilibrium positions are sampled uniformly within an offset value. For the tracking task, 50 initial conditions are sampled from a uniform distribution, whose limits are determined by a scaling of those of the training distribution. A training range fraction greater than 1 (vertical gray dashed line) indicates the distribution range is beyond that used to generate the training trajectories. The median error is shown as a thick line, while the shaded regions represent the 5% to 95% percentile bounds.

5.3 Generalization

We demonstrate the generalizability of the proposed method on both the planar and 3D quadrotor. In all tasks, the goal is to return to the origin, given an initial condition sampled from some uniform distribution centered at the origin. To test the generalizability of the algorithms, we scale the size of the sampling "window" relative to the window on which it was trained, 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 $[-\gamma 1.5, +\gamma 1.5]$. The results for the planar quadrotor are shown in Figure 3b, with γ up to 2.5. As shown, JDMD generalizes well outside of the training window, where the performance of EDMD varies significantly even within the training window, as shown by the growing region that bounds the 5% to 95% percentile of the tracking performance over the 50 test cases. Additionally, in Figure 3a we show the effect of changing the equilibrium position away from the origin: while the true dynamics should be invariant to this change, EDMD fails to learn this whereas JDMD does.

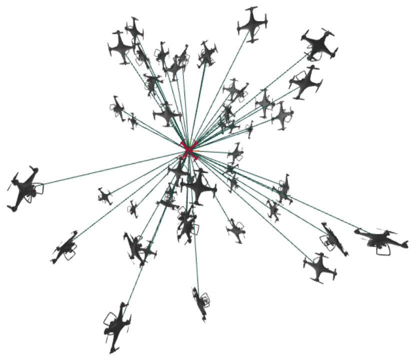
For the full quadrotor, given the goal of tracking a straight line back to the origin, we test 50 initial conditions, many of which are far from the goal, have large velocities, or are nearly inverted (see Figure 4a). The results using an MPC controller are shown in Table 1, demonstrating the excellent generalizability of the algorithm, given that the algorithm was only trained on 30 initial conditions, sampled relatively sparsely given the size of the sampling window. EDMD only successfully brings about 18% of the samples to the origin, while the majority of the time resulting in trajectories like those in Figure 4b. JDMD improves the tracking performance of nominal MPC, which is subject to a constant error bias due to model mismatch, as shown in Fig. 4b.

	Nominal	**EDMD**	**JDMD**
Success Rate	**82%**	18%	80%
Median	0.30	0.63	**0.11**
5% Quantile	0.13	0.08	**0.03**
95% Quantile	0.38	2.62	**0.23**

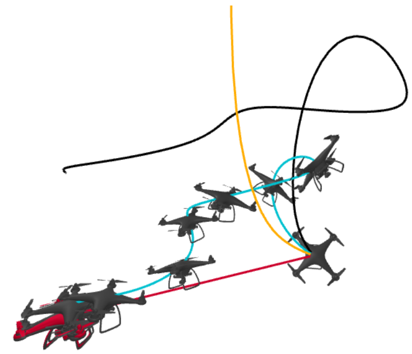
Table 1: Performance summary of MPC tracking of 6-DOF quadrotor. Other than success rate, all values are the tracking error of the successfully stabilized trajectories.

5.4 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



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



(b) Performed trajectories of nominal MPC (black), EDMD (orange), and JDMD (cyan) for tracking infeasible, point-to-point trajectory (red).

Figure 4: Point-to-point, test trajectory generation and example tracking performance of full, 6-DOF quadrotor. The test trajectories generated include a wide scope of initial conditions beyond that of the training set, such as high position offset, large velocities, and near-inverted attitude. JDMD often had the best tracking performance while successfully reaching the goal state, with a similar success rate as nominal MPC within a tighter distribution.

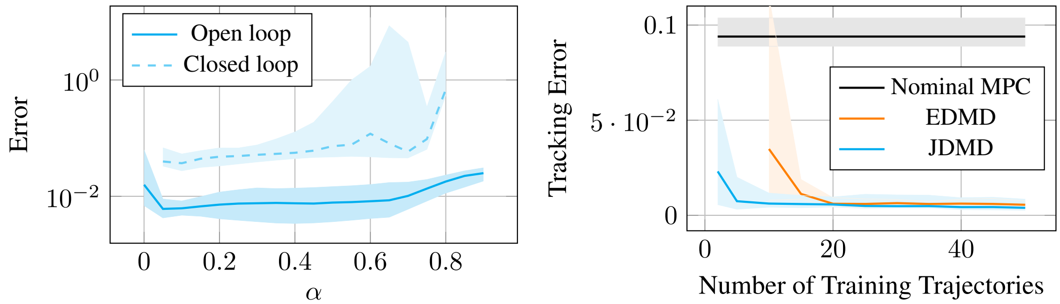
Friction (μ)	0.0	0.1	0.2	0.3	0.4	0.5	0.6
Nominal	✓	✓	✗	✗	✗	✗	✗
EDMD	3	19	6	14	✗	✗	✗
JDMD	2	2	2	2	3	7	12

Table 2: Training trajectories required to stabilize the cartpole with the given friction coefficient

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 Table 2. 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.

5.5 Model Prediction Error vs. Controller Performance

Much of the previous literature on model learning focuses on open-loop dynamics prediction error. While intuitive, we argue that this is a poor metric when the end goal is closed-loop control performance. In Figure 5a we show that decreasing confidence in the analytical model (by increasing α) increases open-loop dynamics prediction error significantly while having minimal impact on closed loop performance below $\alpha = 0.7$. We found we can often quickly find models "good enough" for control with just a few training trajectories (typically with a higher value of α), that predicted the open-loop dynamics very poorly. For example, in Fig. 5a at the extremes of $\alpha = 0$ (EDMD) and $\alpha \geq 0.8$, the open-loop predictions were unstable and diverged, while the closed-loop system still successfully tracked the reference trajectory. This may be unsurprising due to the presence of the Jacobians in the feedback-policy of closed-loop controllers.



(a) Median JDMD model prediction error (open-loop) and MPC tracking error (closed-loop) for perching airplane over varying α values. Closed-loop behavior changes little with respect to open-loop prediction error. The missing open-loop values are points there the states of the open-loop system diverged to infinity.

(b) Sample efficiency for the airplane perching problem. JDMD learns the model with only 5 training trajectories, whereas EDMD requires about 20 to achieve the same performance. Both models perform significantly better than nominal MPC due to significant model mismatch at high angles of attack.

Figure 5: Results on the airplane perching task

6 Limitations

Many of the limitations of the proposed approach derive from the limitations of Koopman approaches more broadly. Foremost among these is the sensitivity of performance to the selections of the nonlinear mapping and respective unlifting operation; the current study has not investigated the incorporation of the proposed method in methods which jointly learn both the model and the nonlinear mapping, although the extension should be fairly straightforward. In addition, the bilinear Koopman model assumes the original, nonlinear dynamics to be control-affine, limiting its application to broad dynamical systems in general. Another significant limitation of the current work is lack of demonstration on hardware, something we plan to remedy in the future. Better, in-depth comparisons of the given approach to other approaches beyond a simple MLP would also be enlightening, which were left out due to scope limitations. Additionally, while the presented single rigid-body systems such as a quadrotor or airplane have similar dimensionality to many autonomous systems of interest, extensions to systems with many degrees of freedom may be difficult computationally, given derivative information grows with the square of the state dimension. In addition, the relationship between closed-loop performance and open-loop dynamics prediction error should be studied futher, given we have demonstrated good MPC performance that has not translated directly to model prediction error. As with most data-driven techniques, it is difficult to claim that our method will increase performance in all cases. It is possible that having an extremely poor prior 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 over standard EDMD. 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 and simply tune α based on the confidence of the model and the quantity (and quality) of training data.

7 Conclusions and Future Work

We have presented JDMD, a simple but powerful extension to EDMD that incorporates derivative information from an approximate prior model. We have tested JDMD in combination with a simple linear MPC control policy across a range of systems and tasks, and have found that the resulting combination can dramatically increase sample efficiency over EDMD, often improving over a nominal MPC policy with just a few sample trajectories. We also showed that the proposed approach is more efficient than a simple multi-layer perception by one or two orders of magnitude. Substantial areas for future work remain: most notably, demonstrating the proposed pipeline on hardware.

Additional directions include applications on sytems with many degrees of freedom such as those whose dynamics are governed by discretized PDEs, lifelong learning or adaptive control applications, combining simulated and real data through the use of modern differentiable physics engines [9, 8], 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] T. A. Howell, S. L. Cleac'h, J. Z. Kolter, M. Schwager, and Z. Manchester. Dojo: A differentiable simulator for robotics. *arXiv preprint arXiv:2203.00806*, 2022.

[9] 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. IEEE, 2012. doi:10.1109/IROS.2012.6386109.

[10] 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`.

[11] 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.

[12] M. Korda and I. Mezić. 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`.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

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

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

[19] 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.

[20] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley. A Data–Driven 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.

[21] A. Surana. *Koopman Operator Based Observer Synthesis for Control-Affine Nonlinear Systems; Koopman Operator Based Observer Synthesis for Control-Affine Nonlinear Systems*. ISBN 978-1-5090-1837-6. doi:10.1109/CDC.2016.7799268.

[22] 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. In *2020 American Control Conference (ACC)*, pages 3906–3913. IEEE, 2020.

[23] C. Folkestad, S. X. Wei, and J. W. Burdick. Koopnet: Joint learning of koopman bilinear models and function dictionaries with application to quadrotor trajectory tracking. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 1344–1350. IEEE, 2022.

[24] Q. Li, F. Dietrich, E. M. Bollt, and I. G. Kevrekidis. Extended dynamic mode decomposition with dictionary learning: A data-driven adaptive spectral decomposition of the koopman operator. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(10):103111, 2017.

[25] R. Wang, Y. Han, and U. Vaidya. Deep koopman data-driven optimal control framework for autonomous racing. *Early Access*, 5, 2021.

[26] E. Kaiser, J. N. Kutz, and S. L. Brunton. Data-driven discovery of koopman eigenfunctions for control. *Machine Learning: Science and Technology*, 2(3):035023, 2021.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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/.

[34] J. Moore, R. Cory, and R. Tedrake. Robust post-stall perching with a simple fixed-wing glider using LQR-Trees. *Bioinspiration &amp$\mathsemicolon$ Biomimetics*, 9(2):025013, May 2014. doi:10.1088/1748-3182/9/2/025013. URL https://doi.org/10.1088/1748-3182/9/2/025013. Publisher: IOP Publishing.

[35] 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.