IEEE Control Systems Letters paper presented at
2025 IEEE 64th Conference on Decision and Control (CDC)
December 10-12, 2025. Rio de Janeiro, Brazil

LiftProj: Physics-Informed Koopman Lifting and Projection for Nonlinear Optimal Control via First-Order Optimization

Jiwoo Choi[1] Jong-Han Kim[†]

Abstract

This paper proposes a first-order optimization framework for nonlinear optimal control problems, efficiently handling complex dynamics via projection onto a lifted, approximately linear constraint manifold constructed using a physics-informed deep Koopman operator. By circumventing repeated convex programming and avoiding penalty-based refinements, the algorithm mitigates sensitivity to hyperparameters and reduces reliance on domain-specific knowledge and manual modeling. A physics-informed loss function preserves physical consistency when mapping back to the original space, enabling fast convergence to near-optimal solutions. Experiments demonstrate improved computational efficiency and stability over established sequential programming approaches.

1 Introduction

Research on nonlinear optimal control increasingly relies on discretizing continuous problems into convex or non-convex optimization frameworks, which improve numerical stability, convergence, and constraint handling, and has been applied extensively [1, 2, 3]. Classical methods typically use direct collocation or multiple shooting with interior point methods solving the KKT conditions via Newton-type methods [4], or Riccati-based techniques to accelerate computation by leveraging problem structures [5].

Several approaches have emerged to handle nonlinear systems. Sequential Convex Programming (SCP) iteratively linearizes the problem around a reference trajectory and solves a series of convex subproblems [6]. While SCP ensures stability, it can be computationally expensive and sensitive to hyperparameters associated with trust regions and virtual control penalties. Other methods, such as problem-specific formulations, utilizes domain-specific knowledge and intuition to transform the problem via variable changes, relaxation, or linearizations [7]. While efficient in certain cases, these approaches do not scale well to complex or general problems.

A more recent approach uses first-order optimization to compute orthogonal projections onto the graph of nonlinear dynamics [8]. Unlike SCP, it solves the nonlinear problem in a single optimization pass without iterative updates. However, its use of Euler discretization and decoupled dynamic projections can cause instability or divergence in strongly nonlinear cases if hyperparameters are not carefully tuned.

Meanwhile, the Koopman operator has gained attention as a tool for addressing nonlinearities by embedding the system in a higher-dimensional latent space where it can be approximated as linear. Deep Koopman operators, which uses neural networks to learn the lifting function [9], have been studied in both linear quadratic [10] and nonlinear control contexts. Prior works such as [11] use Koopman-based prediction within a nonlinear MPC framework [12], apply spectral methods to bilinearize dynamics, and [13] propose deep Koopman networks for trajectory optimization. While effective for forward prediction and planning, they rely on full shooting or iterative convexification.

Our approach leverages a physics-informed deep Koopman operator to enable direct projection onto dynamic constraints within an ADMM-based first-order optimization framework. Unlike prior projection-based methods for nonlinear control [8], which suffer from the difficulty of projecting onto nonlinear dynamical constraint sets, our method lifts dynamics to a linear latent space, making projection tractable and improving scalability. To preserve essential physical constraints when mapping solutions back to the original state space, a physics-informed loss with higher-order consistency terms is introduced for learning the lifting function, thereby promoting alignment with the system's nonlinear dynamics.

Our approach delivers two central advantages: it avoids iterative convexification or penalty-based refinement, and significantly reduces the need for domain-specific tuning by learning the dynamics through Koopman lifting.

We provide an error bound analysis as a key step toward understanding the method's practical convergence behavior and benchmark it against a state-of-the-art SCP solver in a 6-DoF powered descent guidance problem. The results show improved computational efficiency and stability, underscoring the method's applicability to high-dimensional systems with strict constraints and complex nonlinear dynamics.

2 Preliminaries

2.1 Koopman Operator Theory

Koopman Representation of Autonomous Systems Consider an autonomous nonlinear system, $x_{k+1} = f(x_k)$, where $f : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$ is a nonlinear state transition

[1]Jiwoo Choi is with the PGM Systems R&D Center at LIG Nex1 Co., Ltd., Seongnam, South Korea. jiwoo.choi25@lignex1.com
[†]Jong-Han Kim is with the Department of Aerospace Engineering and Program in Aerospace Systems Convergence at Inha University, Incheon, South Korea. Corresponding author. jonghank@inha.ac.kr
[*]This study was supported in part by the KSLV-II Enhancement Program (RS-2022-00164702), hosted by KARI with the funding from KASA, and in part by the Theater Defense Research Center funded by DAPA under grant UD240002SD.

Copyright ©2025 IEEE

function and k is the discrete-time index. Assume the state space $\mathbb{X} \subseteq \mathbb{R}^{n_x}$ is compact and forward invariant under f, *i.e.*, $f(\mathbb{X}) \subseteq \mathbb{X}$. For an observable function $\phi : \mathbb{X} \to \mathbb{R}$, the Koopman operator $\mathcal{K}$ is defined as $\mathcal{K}\phi = \phi \circ f$, and then for any $x_k \in \mathbb{X}$, it follows that [14]:

$$\mathcal{K}\phi(x_k) = \phi \circ f(x_k) = \phi(x_{k+1}). \tag{1}$$

The Koopman operator acts on an infinite-dimensional function space $\mathcal{F}$ [15, 16], but is approximated in a finite-dimensional invariant subspace $\mathcal{F}_{n_f} \subseteq \mathcal{F}$ for practical use. In this subspace, it can be expressed as,

$$\mathcal{K}\phi_j = \textstyle\sum_{i=1}^{n_f} K_{ij}\phi_i, \tag{2}$$

with the basis function $\Phi = \begin{bmatrix}\phi_1 & \cdots & \phi_{n_f}\end{bmatrix}$, and $A = K^T$, giving $\Phi(x_{k+1}) = A\Phi(x_k)$.

Koopman Representation of Systems with Inputs Now we consider a nonlinear system with control inputs, $x_{k+1} = f(x_k, u_k)$ where $u_k \in \mathbb{U} \subseteq \mathbb{R}^{n_u}$ is the control input and $f : \mathbb{R}^{n_x} \times \mathbb{U} \to \mathbb{R}^{n_x}$. If $\Phi(f(\cdot)) \in \operatorname{range}\Phi$, there exist a finite-dimensional lifting function Φ and input matrix $B(x_k, u_k)$ such that

$$\Phi(x_{k+1}) = A\Phi(x_k) + B(x_k, u_k)u_k,$$

as shown in [14]. In practice, the approximation $B(x_k, u_k) \approx B$ simplifies this relation, thus retaining a similar linear-propagation structure in the lifted space. This introduces approximation error, as $B(x_k, u_k)$ may vary across the state-input space; it remains effective if $B(x_k, u_k)$ varies smoothly or the system operates near nominal conditions.

Deep Koopman Operators Recent work has approximated Koopman lifting functions using deep learning models [9, 16, 17]. In particular, an autoencoder-based model was proposed in [9], designed to learn mappings between the original and the lifted space. These models are trained by jointly optimizing the encoder-decoder pair Φ and Φ^{-1} along with the Koopman matrices A and B, to minimize reconstruction error, prediction mismatch, and deviation from linear propagation in the lifted space:

$$\begin{aligned}
\mathcal{L} &= \mathcal{L}_{\text{AE}} + \mathcal{L}_{\text{dyn}} + \mathcal{L}_{\text{lin}} + \gamma\|\mathbf{W}\|_F^2, \\
\mathcal{L}_{\text{AE}} &= \textstyle\sum_{i=1}^{N_d} \|x^{(i)} - \Phi^{-1}(\Phi(x^{(i)}))\|^2, \\
\mathcal{L}_{\text{dyn}} &= \textstyle\sum_{i=1}^{N_d} \|f(x^{(i)}, u^{(i)}) - \Phi^{-1}(A\Phi(x^{(i)}) + Bu^{(i)})\|^2, \\
\mathcal{L}_{\text{lin}} &= \textstyle\sum_{i=1}^{N_d} \|\Phi(f(x^{(i)}, u^{(i)})) - (A\Phi(x^{(i)}) + Bu^{(i)})\|^2,
\end{aligned}$$

where $x^{(i)}$ and $u^{(i)}$ are training samples from the data set of size N_d. The term $\mathcal{L}_{\text{AE}}$ captures the reconstruction error of the autoencoder, $\mathcal{L}_{\text{dyn}}$ penalizes the prediction mismatch between the lifted and the original dynamics, and $\mathcal{L}_{\text{lin}}$ quantifies the deviation from linear evolution in the lifted space. The regularizer term $\|\mathbf{W}\|_F^2$ reduces overfitting.

2.2 First-Order Optimization

First-order methods solve optimization problems employing gradient or subgradient information and address constraints via proximal operations [18]. The ADMM, a popular first-order method, introduces auxiliary variables to handle constraints through projection operations [19]. Given a general optimization problem:

$$\text{minimize}_{x \in \mathcal{C}} \quad g(x),$$

it can be formulated as the following equivalent problem:

$$\begin{array}{ll} \text{minimize}_{x,z} & g(x) + I_{\mathcal{C}}(z) \\ \text{subject to} & x = z, \end{array} \tag{3}$$

with the augmented Lagrangian defined by $\mathcal{L}_\rho(x, z, u) = g(x) + I_{\mathcal{C}}(z) + (1/2)\rho\|x - z + u\|^2$, where ρ is the penalty parameter, u is the scaled dual variable, and $I_{\mathcal{C}}(\cdot)$ is the indicator function (0 for $x \in \mathcal{C}$, ∞ otherwise). Problem (3) is solved by the following iterative updates:

$$\begin{aligned}
x^{j+1} &= \arg\min_x \; g(x) + (1/2)\rho\|x - z^j + u^j\|, \\
z^{j+1} &= \Pi_{\mathcal{C}}(x^{j+1} + u^j), \\
u^{j+1} &= u^j + x^{j+1} - z^{j+1},
\end{aligned}$$

where $\Pi_{\mathcal{C}}(\cdot)$ denotes the orthogonal projection onto $\mathcal{C}$.

Projections onto nonconvex sets can violate nonexpansivity and do not guarantee convergence in general; however, we can frequently find practically good solutions in many cases [3, 8]. Although general convergence theory is incomplete, partial analyses appear in [20, 21, 22]

3 Proposed Method: LiftProj

3.1 ADMM-Based Nonlinear Optimal Control

We consider a finite-horizon discrete-time nonlinear optimal control where the system evolves by $x_{t+1} = f(x_t, u_t)$ and is subject to initial, terminal, and pointwise constraints:

$$\begin{array}{ll} \text{minimize}_{x,u} & g(x_1, \ldots, x_N, u_0, \ldots, u_{N-1}) \\ \text{subject to} & x_0 = x_{\text{init}}, \quad x_N = x_{\text{des}}, \\ & (x_{t+1}, u_t) \in \mathcal{C}_t^{\text{con}}, \\ & x_{t+1} = f(x_t, u_t), \quad t \in \{0, \ldots, N-1\}, \end{array} \tag{4}$$

where x_t and u_t are the state and control input at time step t, and $f(x_t, u_t)$ denotes discrete-time dynamics, implemented via a 4th-order Runge-Kutta (RK4) scheme for higher accuracy. A stack notation $(a, b) = \begin{bmatrix}a^T & b^T\end{bmatrix}^T$, with $x = (x_1, \ldots, x_N)$ and $u = (u_0, \ldots, u_{N-1})$, is used here. The set $\mathcal{C}_t^{\text{con}}$ encodes time-indexed constraints (*e.g.*, input bounds or safety limits), and the objective function g penalizes the entire state-control trajectory.

Letting $y = (x, u)$, the problem becomes:

$$\begin{array}{ll} \text{minimize}_y & g(y) \\ \text{subject to} & Cy - d \in \mathcal{C} \end{array} \tag{5}$$

where a matrix $C = \begin{bmatrix}(C^{\text{con}})^T & (C^{\text{dyn}})^T\end{bmatrix}^T$ and a vector $d = \begin{bmatrix}(d^{\text{con}})^T & (d^{\text{dyn}})^T\end{bmatrix}^T$ are defined such that $C^{\text{con}}y - d^{\text{con}} \in \mathcal{C}^{\text{con}}$ encodes pointwise control and state constraints at each time step, and $C^{\text{dyn}}y - d^{\text{dyn}} \in \mathcal{C}^{\text{dyn}}$ encodes the discrete dynamics equations. The calligraphic symbols $\mathcal{C}^{\text{dyn}}$ and $\mathcal{C}^{\text{con}}$ denote the corresponding feasible sets used for expressing projections. Also, we have $\mathcal{C} = \mathcal{C}^{\text{con}} \cap \mathcal{C}^{\text{dyn}}$, $\mathcal{C}^{\text{con}} = \bigcap_{t=0}^{N-1} \mathcal{C}_t^{\text{con}}$, and $\mathcal{C}^{\text{dyn}} = \{(x, u) \mid x_{t+1} = f(x_t, u_t),\ \text{for } t = 0, \ldots, N-1\}$.

To enable ADMM updates, we introduce auxiliary variables $s = (s^{\text{con}}, s^{\text{dyn}})$, reformulating (5) as:

$$\begin{array}{ll} \text{minimize}_{y,s} & g(y) + I_{\mathcal{C}}(s) \\ \text{subject to} & Cy = d + s, \end{array} \tag{6}$$

with the augmented Lagrangian $\mathcal{L}_\rho(y, s, \lambda) = g(y) +$

$I_{\mathcal{C}}(s) + \frac{\rho}{2}\|Cy - d - s + \lambda\|_2^2$, where ρ is the penalty parameter, λ is the dual variable, and $I_{\mathcal{C}}(\cdot)$ denotes the indicator function. The optimal solution to Problem (6) is obtained by iteratively computing the following ADMM updates:

$$\begin{aligned} y^{j+1} &= \arg\min_y \ \mathcal{L}_\rho(y, s^j, \lambda^j), \\ s^{j+1} &= \arg\min_s \ \mathcal{L}_\rho(y^{j+1}, s, \lambda^j), \\ &= \begin{bmatrix} \Pi_{\mathcal{C}^{\text{con}}}(C^{\text{con}} y^{j+1} - d^{\text{con}} + \lambda^{\text{con},j}) \\ \Pi_{\mathcal{C}^{\text{dyn}}}(C^{\text{dyn}} y^{j+1} - d^{\text{dyn}} + \lambda^{\text{dyn},j}) \end{bmatrix}, \\ \lambda^{j+1} &= \lambda^j + Cy^{j+1} - d - s^{j+1}. \end{aligned}$$

The key computational difference arises in the projection step: The projection $\Pi_{\mathcal{C}^{\text{con}}}$ enforces pointwise bounds (*e.g.*, thrust limits, gimbal angles, velocity bounds) and is decomposable across time steps:

$$\Pi_{\mathcal{C}^{\text{con}}}(x, u) = \left(\Pi_{\mathcal{C}_0^{\text{con}}}(x_1, u_0), \dots, \Pi_{\mathcal{C}_{N-1}^{\text{con}}}(x_N, u_{N-1})\right),$$

enabling parallel computation and improved efficiency.

In contrast, the projection $\Pi_{\mathcal{C}^{\text{dyn}}}$ enforces intertemporal consistency via nonlinear dynamics (*e.g.*, $x_{t+1} = f(x_t, u_t)$), and is inherently non-decomposable across time steps. Because it couples variables across adjacent steps, $\Pi_{\mathcal{C}^{\text{dyn}}}$ requires solving a coupled optimization problem that spans the entire horizon. As a result, $\Pi_{\mathcal{C}^{\text{dyn}}}$ becomes the primary computational bottleneck in ADMM iterations. Moreover, computing $\Pi_{\mathcal{C}^{\text{dyn}}}$ for nonlinear f does not admit a closed-form solution in general, requiring the following projection subproblem:

$$\begin{aligned} &\text{minimize}_{x,u} && \|x - \hat{x}\|^2 + \|u - \hat{u}\|^2 \\ &\text{subject to} && x_{t+1} = f(x_t, u_t), \quad t \in \{0, \dots, N-1\}, \end{aligned} \tag{7}$$

where $(\hat{x}, \hat{u})$ is the point to be projected via $\Pi_{\mathcal{C}^{\text{dyn}}}$.

This yields an $N(n_x + n_u)$-dimensional problem that must be solved at each ADMM iteration, making (7) computationally intensive and motivating efficient alternatives for $\Pi_{\mathcal{C}^{\text{dyn}}}$. To address this, LiftProj approximates this nonlinear dynamics projection within a tractable, lifted linear space.

3.2 LiftProj Method

The projection $\Pi_{\mathcal{C}^{\text{dyn}}}$ that solves Problem (7) can be recast in terms of $y = (x, u)$ as follows:

$$y^* = \Pi_{\mathcal{C}^{\text{dyn}}}(\hat{y}) = \arg\min_{y \in \mathcal{C}^{\text{dyn}}} \|y - \hat{y}\|, \tag{8}$$

which computes the Euclidean projection of $\hat{y} = (\hat{x}, \hat{u})$ onto the feasible set $\mathcal{C}^{\text{dyn}}$ in the original space.

Instead of solving (8) directly, this paper proposes the LiftProj method, which computes the projection in the lifted linear space. Note that the feasible set in the lifted space becomes a hyperplane, making projection more tractable.

$$\begin{aligned} &\text{minimize}_{z \in \mathcal{C}^{\text{dyn}}} \ \|\Phi(x) - \Phi(\hat{x})\|^2 + \|u - \hat{u}\|^2 \\ \iff \quad &\text{minimize}_z \ \|z - \hat{z}\|^2 \ \text{subject to} \ \tilde{A}z = \tilde{b} \end{aligned}$$

With $\Phi(x) = (\Phi(x_1), \dots, \Phi(x_N))$ and $z = (\Phi(x), u)$, the linear operators $\tilde{A}$ and $\tilde{b}$ describe the dynamics of the lifted linear system, are derived from the state transition matrix A and the input matrix B.

As shown in Fig. 1, the LiftProj procedure comprises:

1. *Embedding:* Map $\hat{y} = (\hat{x}, \hat{u})$ into $\hat{z} = (\Phi(\hat{x}), \hat{u})$.
2. *Projection:* Project $\hat{z} = (\Phi(\hat{x}), \hat{u})$ onto the hyperplane where the lifted linear system dynamics hold, yielding
$$z^* = (z_x^*, z_u^*) = P\hat{z} = \hat{z} + \tilde{A}^T(\tilde{A}\tilde{A}^T)^{-1}(\tilde{b} - \tilde{A}\hat{z}),$$
where P is the orthogonal projection onto $\tilde{A}z = \tilde{b}$.
3. *Inverse embedding:* Apply the inverse mapping Φ^{-1} to return to the original state and input space:
$$y^*_{\text{LiftProj}} = (x^*_{\text{LiftProj}}, u^*_{\text{LiftProj}}) = (\Phi^{-1}(z_x^*), z_u^*).$$

The LiftProj result y^*_{LiftProj} may differ from y^* obtained via (8) in the original space, *i.e.*, $y^*_{\text{LiftProj}} \neq y^*$. However, ADMM tolerates certain level of inexact projection [19], and similar strategies have been analyzed in various optimization methods [23, 24, 25].

Importantly, the LiftProj error is provably bounded by the Lipschitz constants of Φ and Φ^{-1}.

Lemma 1 (Error bound of LiftProj)**.** *Let $y = (x, u)$ and define the lifted map $\tilde{\Phi}(y) = (\Phi(x), u)$, where Φ is bi-Lipschitz continuous with constants L_Φ and $L_{\Phi^{-1}}$. Then $\tilde{\Phi}$ is bi-Lipschitz with constants $L_{\tilde{\Phi}} = \max(L_\Phi, 1)$ and $L_{\tilde{\Phi}^{-1}} = \max(L_{\Phi^{-1}}, 1)$. Let $y^* = \Pi_{\mathcal{C}^{\text{dyn}}}(y)$ denote the exact projection, and let $y^*_{\text{LiftProj}} = \tilde{\Phi}^{-1}(P\tilde{\Phi}(y))$ be the lifted-space projection. Then we have that:*

$$\|y^*_{\text{LiftProj}} - y^*\| \le L_{\tilde{\Phi}^{-1}} L_{\tilde{\Phi}} \|y - y^*\|.$$

Proof. By construction, P is the Euclidean projection onto a hyperplane in the lifted space and is nonexpansive. Applying the Lipschitz continuity of $\tilde{\Phi}$ and its inverse yields:

$$\begin{aligned} \|y^*_{\text{LiftProj}} - y^*\| &= \|\tilde{\Phi}^{-1}(P\tilde{\Phi}(y)) - \tilde{\Phi}^{-1}(P\tilde{\Phi}(y^*))\| \\ &\le L_{\tilde{\Phi}^{-1}} \|P\tilde{\Phi}(y) - P\tilde{\Phi}(y^*)\| \le L_{\tilde{\Phi}^{-1}} \|\tilde{\Phi}(y) - \tilde{\Phi}(y^*)\| \\ &\le L_{\tilde{\Phi}^{-1}} L_{\tilde{\Phi}} \|y - y^*\|, \end{aligned}$$

as desired. ■

This result shows that the projection error introduced by lifting is provably bounded by the Lipschitz constants of the encoder and the decoder, providing a key step toward understanding the practical convergence behavior of the overall ADMM scheme under inexact projections.

3.3 Physics-Informed Deep Koopman Operator

To implement the LiftProj method, we construct a physics-informed deep Koopman operator using an autoencoder.

LiftProj operates by projecting onto the Koopman invariant subspace defined by A and B. To ensure consistency with the original nonlinear dynamics, the following

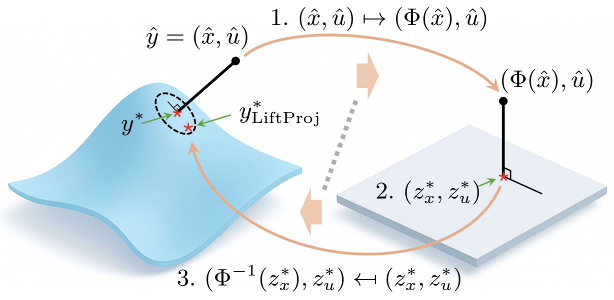
1. $(\hat{x}, \hat{u}) \mapsto (\Phi(\hat{x}), \hat{u})$
$\hat{y} = (\hat{x}, \hat{u})$
$(\Phi(\hat{x}), \hat{u})$
y^*
y^*_{LiftProj}
2. (z_x^*, z_u^*)
3. $(\Phi^{-1}(z_x^*), z_u^*) \leftarrow (z_x^*, z_u^*)$

Figure 1: The LiftProj procedures involving three steps.

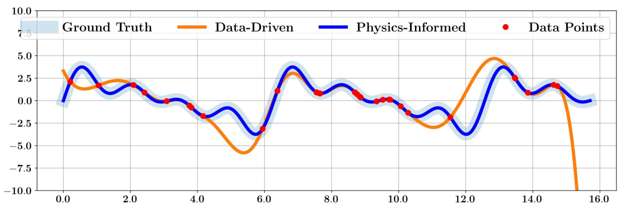

Figure 2: Continuous approximation. Generalization improves with physics-informed high-order consistency terms.

must hold:

$$\Phi(f_{\text{RK}}(x_t,u_t)) = A\Phi(x_t) + Bu_t, \tag{10}$$

where f_{RK} denotes the nonlinear propagation implemented by Runge-Kutta integration. The Koopman operator should provide a continuous approximation to this condition.

Previous deep Koopman methods have mainly used data-driven approaches [9, 17] and often fail to generalize or ensure continuity. To overcome these limitations, we propose a physics-informed loss function incorporating higher-order consistency terms, motivated by research showing improved generalization when physics-informed terms are integrated into deep learning models [26, 27]. See Fig. 2.

In this paper, the neural network model f_{NN}, which includes the encoder $\Phi(x_t;\theta_{\text{ENC}})$, decoder $\Phi^{-1}(x_t;\theta_{\text{DEC}})$, and matrices A and B, must approximate the nonlinear propagation function $f_{\text{RK}}(x_t,u_t)$:

$$f_{\text{NN}}(x_t,u_t;\theta) = \Phi^{-1}\left(A\Phi(x_t;\theta_{\text{ENC}}) + Bu_t;\theta_{\text{DEC}}\right),$$

and additionally, the composition of the encoder and decoder must approximate the identity: $\Phi^{-1}\circ\Phi \equiv I$.

We train the model using a composite loss that enforces these structural equivalences:

$$\theta^* = \arg\min_{\theta}\left(\mathcal{L}^{\text{DD}}_{\text{dyn}} + \mathcal{L}^{\text{DD}}_{\text{AE}} + \mathcal{L}^{\text{DD}}_{\text{lin}} + \mathcal{L}^{\text{PI}}_{\text{dyn}} + \mathcal{L}^{\text{PI}}_{\text{AE}} + \mathcal{L}^{\text{PI}}_{\text{lin}}\right),$$

with the data-driven (DD) losses given by:

$$\begin{aligned}
\mathcal{L}^{\text{DD}}_{\text{dyn}} &= \textstyle\sum_{i=1}^{N_d}\|f_{\text{NN}}(x^{(i)},u^{(i)};\theta) - f_{\text{RK}}(x^{(i)},u^{(i)})\|^2\\
\mathcal{L}^{\text{DD}}_{\text{AE}} &= \textstyle\sum_{i=1}^{N_d}\|\Phi^{-1}(\Phi(x^{(i)})) - x^{(i)}\|^2\\
\mathcal{L}^{\text{DD}}_{\text{lin}} &= \textstyle\sum_{i=1}^{N_d}\|\Phi(f_{\text{RK}}(x^{(i)},u^{(i)})) - (A\Phi(x^{(i)}) + Bu^{(i)})\|^2
\end{aligned}$$

and the physics-informed (PI) losses given by:

$$\begin{aligned}
\mathcal{L}^{\text{PI}}_{\text{dyn}} &= \textstyle\sum_{i=1}^{N_d}\|J_{f_{\text{NN}}}(x^{(i)},u^{(i)}) - J_{f_{\text{RK}}}(x^{(i)},u^{(i)})\|_F^2,\\
\mathcal{L}^{\text{PI}}_{\text{AE}} &= \textstyle\sum_{i=1}^{N_d}\|J_{\Phi^{-1}\cdot\Phi}(x^{(i)}) - I\|_F^2,\\
\mathcal{L}^{\text{PI}}_{\text{lin}} &= \textstyle\sum_{i=1}^{N_d}\|J_{\Phi\circ f_{\text{NN}}}(x^{(i)},u^{(i)}) - J_{\Phi\circ f_{\text{RK}}}(x^{(i)},u^{(i)})\|_F^2,
\end{aligned}$$

where J denotes the Jacobian operator, and $x^{(i)}$ and $u^{(i)}$ represent randomly sampled collocation points.

3.4 Discussion on Convergence Properties

While LiftProj performs first-order optimization with projections in a lifted space, the overall problem remains

Table 1: Architecture of the deep Koopman operator.

	Encoder		Decoder	
Layer Type	Input	Layer 1	Layer 2	Output
Operator	FC/GELU	FC	FC/GELU	FC
Dimension	17×1000	1000×50	50×1000	1000×17

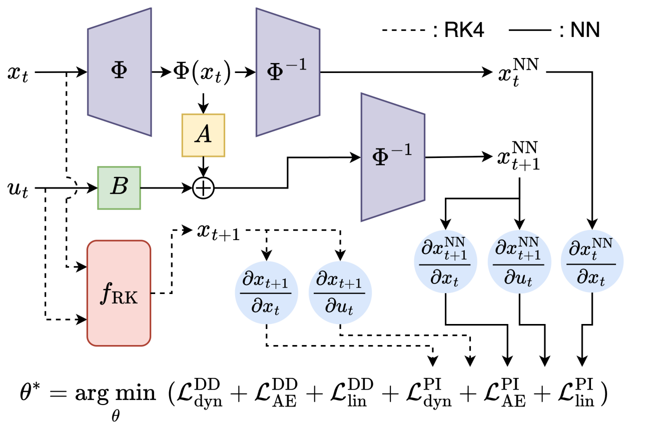
- - - - : RK4
—— : NN
x_t
Φ
$\Phi(x_t)$
Φ^{-1}
x_t^{NN}
A
u_t
B
Φ^{-1}
x_{t+1}^{NN}
x_{t+1}
f_{RK}
$\frac{\partial x_{t+1}}{\partial x_t}$
$\frac{\partial x_{t+1}}{\partial u_t}$
$\frac{\partial x_{t+1}^{\text{NN}}}{\partial x_t}$
$\frac{\partial x_{t+1}^{\text{NN}}}{\partial u_t}$
$\frac{\partial x_t^{\text{NN}}}{\partial x_t}$
$\theta^* = \arg\min_{\theta}(\mathcal{L}^{\text{DD}}_{\text{dyn}} + \mathcal{L}^{\text{DD}}_{\text{AE}} + \mathcal{L}^{\text{DD}}_{\text{lin}} + \mathcal{L}^{\text{PI}}_{\text{dyn}} + \mathcal{L}^{\text{PI}}_{\text{AE}} + \mathcal{L}^{\text{PI}}_{\text{lin}})$

Figure 3: Proposed physics-informed deep Koopman operator.

nonconvex due to the nonlinear dynamics and the use of learned approximate lifting.

LiftProj can be interpreted as a variant of mirror descent, where updates are performed in a transformed (Koopman-invariant) space and mapped back via the inverse encoder. This mirrors mirror descent schemes [28], which exploit geometric structure through updates in dual spaces.

A key to convergence in mirror descent is controlling distortion from the mapping and its inverse. In LiftProj, the lifting Φ and its inverse Φ^{-1} are trained to be bi-Lipschitz continuous, and this ensures that the projection error of LiftProj can be bounded by a constant multiple of the residual norm.

To promote this structure, we apply several techniques: (i) a reconstruction loss to enforce encoder-decoder consistency, (ii) Jacobian-based penalties to regulate local distortion, and (iii) spectral normalization to control the Lipschitz constants of the networks (regularizing the spectral norm of each layer's weight matrix). These encourage well-conditioned and stable mappings that satisfy the assumptions for convergence.

This bounded inexactness is critical. As shown in recent works [20, 21], ADMM with inexact projections still converges under certain conditions, provided the projection error remains proportional to the current residual.

While a formal proof is left for future work, our analysis provides theoretical support for the convergence of LiftProj. Under mild assumptions (*e.g.*, Lipschitz continuity and low distortion), its inexact ADMM iterations behave similarly to mirror descent in nonconvex optimization.

4 Numerical Examples

We illustrate the effectiveness of the proposed method on a powered descent guidance problem for reusable rockets.

Table 2: Initial conditions and simulation parameters.

Param.	**Value**	**Param.**	**Value**
q_{init}	$(1,0,0,0)$	q_{des}	$(1,0,0,0)$
ω_{init}	$(0,0,0)$	ω_{des}	$(0,0,0)$
r_{init}	$(1,1,0)$	r_{des}	$(0,0,0)$
v_{init}	$(-0.2,0,0.1)$	v_{des}	$(-0.1,0,0)$
g	$(-1,0,0)$	$r_{T,\mathcal{B}}$	$(-1,0,0)$
J	diag$(1,1,1)$	$(m_{\text{wet}},\, m_{\text{dry}})$	$(2,0.5)$
α	0.05	$(T_{\min},\, T_{\max})$	$(0.5,\, 2.6)$
t_f	5	$(\theta_{\max},\, \delta_{\max}, \gamma_{\text{gs}})$	$(45°,20°,10°)$

4.1 6-DoF Powered Descent Guidance

The continuous-time 6-DoF powered descent guidance problem is formulated as follows [6]:

$$\begin{aligned} \text{minimize}_{x(t)} \quad & -m(t_f) \\ \text{subject to} \quad & \dot{\omega}(t) = J^{-1}(r_{T,\mathcal{B}} \times T^{\mathcal{B}}(t) - \omega(t) \times J\omega(t)), \\ & \dot{q}(t) = (1/2)\Omega(\omega(t))q(t), \\ & \dot{v}^{\mathcal{I}}(t) = C^{\mathcal{I}}_{\mathcal{B}}(q(t))T^{\mathcal{B}}(t)/m(t) + g^{\mathcal{I}}, \\ & \dot{r}^{\mathcal{I}}(t) = v^{\mathcal{I}}(t), \qquad \dot{m}(t) = -\alpha\|T^{\mathcal{B}}(t)\|, \\ & \dot{T}^{\mathcal{B}} = \dot{T}_{\text{cmd}}, \qquad T_{\text{lb}} \le \|T^{\mathcal{B}}(t)\| \le T_{\text{ub}}, \\ & \tan(\gamma_{\text{gs}})\, r^{\mathcal{I}}_{e,n}(t) \le r^{\mathcal{I}}_{u}(t), \\ & \cos(\theta_{\max}) \le 1 - 2q_{2,3}(t)^T q_{2,3}(t), \\ & \cos(\delta_{\max})\|T^{B}(t)\| \le T^{B}_{u}, \\ & x(0) = x_{\text{init}}, \qquad x(t_f) = x_{\text{des}}, \end{aligned} \tag{11}$$

where $x = (q, \omega, r^{\mathcal{I}}, v^{\mathcal{I}}, m, T^{\mathcal{B}})$ and $\dot{T}_{\text{cmd}}$ denote quaternion, angular rate, position, velocity, mass, thrust, and thrust rate command, respectively. The formulation captures coupled nonlinear rotational and translational dynamics, along with constraints on thrust magnitude, gimbal angle, tilt angle, and glide slope to ensure safe landing. Refer to [6] for details on variables and physical constants.

4.2 Deep Koopman Operator

The deep Koopman operator architecture is shown in Table 1. The model was trained using the ADAM optimizer on an NVIDIA A100 GPU with a batch size of 512 and a learning rate of 0.001 decayed by a factor of 0.1. While LiftProj enables fast optimization via lifted projections, it incurs a one-time offline cost for Koopman model training. For Section IV experiments, training the PI model with 10,000 samples took approximately 1.7 hours. Inference is fast, with the Koopman forward-inverse passes taking less than 9 ms per ADMM iteration on an Apple M2 processor.

4.3 Optimization Results

We compare LiftProj against SCP using a prediction horizon $N = 40$. Scaled initial conditions and parameters for (11) are listed in Table 2. All algorithms used Python 3.10.13 and the SCP problems were solved with `MOSEK`.

The SCP implementation used in our experiments follows [6] and employs the RK4 discretization scheme. Constraint drift is controlled with virtual-control penalties (10^1–10^5) and convexification is stabilized with trust-region weights (10^{-2}–10^2). Large penalties tighten feasibility but may slow convergence; very small trust regions have a similar effect. Iterations stop when the virtual-control norm drops below 2.0×10^{-3} and the trust-region radius below 1.7×10^1. The iteration cap was 20. This well-established setup matches our problem structure and is widely used in the literature.

Table 3 reports computation time, propagation error, and fuel consumption. Propagation error represents the terminal position error from applying the optimized control to the full nonlinear dynamics, reflecting solution fidelity. The LiftProj with the physics-informed deep Koopman model achieved the shortest computation time and smallest propagation error. The 10k-sample data-driven model performed worse than the 50k version due to poor

Table 3: Performance comparison of LiftProj with physics-informed (PI), data-driven (DD) models, and SCP.

Model	**Compute Time** [sec]	**Propagation Error**	**Fuel Use**
LiftProj (PI, 10k)	1.5	5.26×10^{-3}	0.0492
LiftProj (DD, 10k)	20.2	3.15×10^{-2}	0.0421
LiftProj (DD, 50k)	2.6	1.78×10^{-2}	0.0488
SCP	2.4	5.49×10^{-3}	0.0495

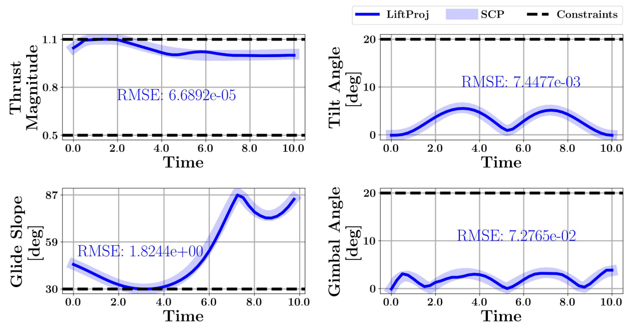

Figure 4: Optimization results displaying constraint satisfaction and state/control trajectories.

accuracy and generalizability, resulting in more ADMM iterations. Still, the physics-informed model remained effective even with fewer data, showing its robustness. Fig. 4 shows constraints satisfaction, and Fig. 6 highlights LiftProj's significantly lower sensitivity compared to SCP.

4.4 Computational Cost and Hyperparameter Sensitivity

LiftProj solves the problem in 1.5 seconds, only modestly faster than SCP's 2.4 seconds. However, this comparison significantly understates the practical efficiency of LiftProj. SCP required extensively careful hyperparameter tuning, while LiftProj required none. Fig. 6 demonstrates LiftProj's robustness to parameter variations; among all SCP runs, only the red box achieves a propagation error comparable to LiftProj.

The Koopman model is trained once and reused across multiple instances, making LiftProj ideal for repeated use in replanning, MPC, or batch policy settings.

Crucially, SCP's performance deteriorates without tuning, often taking 5-15 times longer or diverging, while LiftProj remains stable. This robustness makes it attractive for deployment in uncertain or automated pipelines.

5 Concluding Remarks

In this paper, we proposed LiftProj, a first-order optimization framework for nonlinear optimal control that performs projection in a lifted, linearized space while enforcing physical constraints in the original space. Numerical results show that LiftProj achieves performance competitive with state-of-the-art SCP, offering improved scalability and parameter robustness. By leveraging a physics-informed deep Koopman operator, LiftProj reduces dependence on problem-specific knowledge and enhances generalization. This makes it well-suited for complex, high-dimensional control tasks.

LiftProj has some limitations. Although our error analysis provides partial support, theoretical convergence is not guaranteed due to nonconvexity and approximate lifting. Bi-Lipschitz continuity is promoted during training,

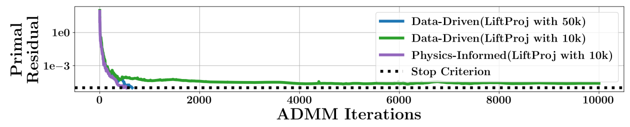

Figure 5: Convergence of LiftProj. Iterations terminates once the squared norm of the primal residual drops below 10^{-5}.

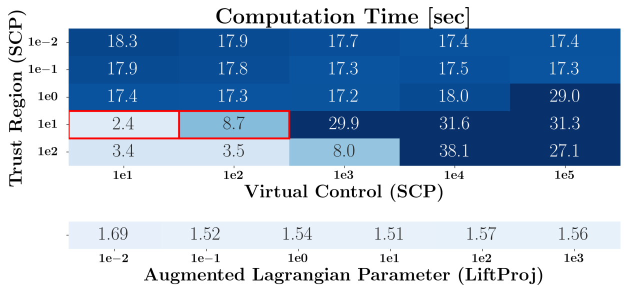

Figure 6: Sensitivity of computation time to hyperparameter variations. Note that LiftProj (Bottom) shows lower sensitivity than SCP (Top).

though not strictly enforced. Generalization may degrade for out-of-sample data, and training can be numerically expensive. Still, the trained Koopman model is reusable across different tasks, making LiftProj suitable for repeated or real-time control.

References

[1] L. Blackmore, "Autonomous precision landing of space rockets," in *Frontiers of Engineering*, vol. 46, pp. 15–20, The Bridge, 2016.

[2] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake, "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," *Auton. Robots*, vol. 40, pp. 429–455, 2016.

[3] J. Choi and J.-H. Kim, "Powered descent guidance via first-order optimization with expansive projection," *IEEE Access*, vol. 12, pp. 46232–46240, 2024.

[4] M. Diehl, H. J. Ferreau, and N. Haverbeke, "Efficient numerical methods for nonlinear MPC and moving horizon estimation," *Nonlinear Model Predictive Control*, pp. 391–417, 2009.

[5] L. Vanroye, A. Sathya, J. De Schutter, and W. Decré, "FATROP: A fast constrained optimal control problem solver for robot trajectory optimization and control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pp. 10036–10043, 2023.

[6] M. Szmuk, U. Eren, and B. Açıkmeşe, "Successive convexification for mars 6-dof powered descent landing guidance," in *Proc. AIAA Guidance, Navigation, and Control Conf.*, p. 1500, 2017.

[7] B. Açıkmeşe, J. M. Carson, and L. Blackmore, "Lossless convexification of nonconvex control bound and pointing constraints of the soft landing optimal control problem," *IEEE Trans. Control Syst. Technol.*, vol. 21, no. 6, pp. 2104–2113, 2013.

[8] Y.-J. Kim, J. Choi, J. Choi, and J.-H. Kim, "A first-order approach for optimal control of nonlinear dynamical systems," in *Proc. 10th Int. Conf. Control Decis. Inf. Technol. (CoDIT)*, pp. 2290–2295, IEEE, 2024.

[9] B. Lusch, J. N. Kutz, and S. L. Brunton, "Deep learning for universal linear embeddings of nonlinear dynamics," *Nature Commun.*, vol. 9, no. 1, p. 4950, 2018.

[10] Y. Han, W. Hao, and U. Vaidya, "Deep learning of Koopman representation for control," in *Proc. 59th IEEE Conf. Decis. Control (CDC)*, pp. 1890–1895, 2020.

[11] C. Folkestad and J. W. Burdick, "Koopman NMPC: Koopman-based learning and nonlinear model predictive control of control-affine systems," in *Proc. IEEE Int. Conf. Robot. Autom.*, pp. 7350–7356, 2021.

[12] D. Goswami and D. A. Paley, "Bilinearization, reachability, and optimal control of control-affine nonlinear systems: A Koopman spectral approach," *IEEE Trans. Autom. Control*, vol. 67, pp. 2715–2728, 2021.

[13] H. Shi and M. Q.-H. Meng, "Deep Koopman operator with control for nonlinear systems," *IEEE Robot. Autom. Lett.*, vol. 7, no. 3, pp. 7700–7707, 2022.

[14] L. C. Iacob, R. Tóth, and M. Schoukens, "Koopman form of nonlinear systems with inputs," *Automatica*, vol. 162, p. 111525, 2024.

[15] P. Bevanda, S. Sosnowski, and S. Hirche, "Koopman operator dynamical models: Learning, analysis and control," *Annu. Rev. Control*, vol. 52, pp. 197–212, 2021.

[16] G. Nehma and M. Tiwari, "Leveraging KANs for enhanced deep Koopman operator discovery," *arXiv preprint arXiv:2406.02875*, 2024.

[17] E. Yeung, S. Kundu, and N. Hodas, "Learning deep neural network representations for Koopman operators of nonlinear dynamical systems," in *Proc. American Control Conf. (ACC)*, pp. 4832–4839, 2019.

[18] N. Parikh and S. Boyd, "Proximal algorithms," *Found. Trends Optim.*, vol. 1, no. 3, pp. 127–239, 2014.

[19] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, 2011.

[20] Y. Wang, W. Yin, and J. Zeng, "Global convergence of ADMM in nonconvex nonsmooth optimization," *J. Sci. Comput.*, vol. 78, pp. 29–63, 2019.

[21] X. Yi, S. Zhang, T. Yang, T. Chai, and K. H. Johansson, "Linear convergence of first-and zeroth-order primal–dual algorithms for distributed nonconvex optimization," *IEEE Trans. Autom. Control*, vol. 67, no. 8, pp. 4194–4201, 2021.

[22] L. Yang, "Proximal gradient method with extrapolation and line search for a class of non-convex and non-smooth problems," *J. Optim. Theory Appl.*, vol. 200, no. 1, pp. 68–103, 2024.

[23] F. R. de Oliveira, O. P. Ferreira, and G. N. Silva, "Newton's method with feasible inexact projections for solving constrained generalized equations," *Comput. Optim. Appl.,*, vol. 72, pp. 159–177, 2019.

[24] D. S. Gonçalves, M. L. Gonçalves, and F. R. Oliveira, "An inexact projected lm type algorithm for solving convex constrained nonlinear equations," *J. Comput. Appl. Math.*, vol. 391, p. 113421, 2021.

[25] A. Patrascu and I. Necoara, "On the convergence of inexact projection primal first-order methods for convex minimization," *IEEE Trans. Autom. Control*, vol. 63, no. 10, pp. 3317–3329, 2018.

[26] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018.

[27] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *J. Comput. Phys.*, vol. 378, pp. 686–707, 2019.

[28] A. Beck and M. Teboulle, "Mirror descent and nonlinear projected subgradient methods for convex optimization," *Oper. Res. Lett.*, vol. 31, no. 3, pp. 167–175, 2003.