A Neural Koopman Framework for CubeSat Modeling and Control

Omar Shouman[1], Mohamed Mabrok[2] and Tamer Khattab[1]

***Abstract*— CubeSats have gained significant attention due to their low cost and versatility, yet their low inertia makes them particularly vulnerable to disturbances, such as magnetic torques and residual atmospheric drag, that induce strong nonlinearities in their rotational dynamics. While nonlinear controllers can address these challenges, their high computational burden and intricate stability analyses often forces practitioners to rely on linearization-based techniques, which fail to capture the full extent of the system's nonlinear behavior. Moreover, existing Koopman-based approaches for attitude control have not exploited the quaternion representation intrinsic to CubeSat dynamics, nor have they adequately addressed the impact of environmental disturbances and sensor noise, an important gap in current research. In this paper, we propose a neural Koopman framework that lifts the nonlinear dynamics, represented in quaternion form, into a higher-dimensional linear space. By training deep neural networks as lifting functions using a control-aware loss function, our method enables the design of efficient linear controllers such as LQR. Comparative results demonstrates that the Koopman model preserves the unit-norm property of quaternions under sensor noise and disturbances and delivers superior reference tracking performance compared to the conventional linearized model, which suffers from renormalization issues. These findings highlight the potential of the proposed framework for achieving robust and computationally efficient CubeSat attitude control.**

***Index Terms*—** CubeSat, Deep Learning, Koopman Theory, Quaternions, Data-driven model

I. Introduction

Nowadays, CubeSats have become an essential advancement in satellite technology and space exploration, altering how humans conduct scientific study and Earth observation. They have gained popularity due to their cost-effectiveness and versatility, serving a wide range of applications, including scientific research, technology demonstrations, communication experiments and remote sensing, which broadens the scope of space missions [1].

Maintaining a stable and predictable orientation is critical for mission success, which is the role of the Attitude Determination and Control System (ADCS). The ADCS continuously monitors and adjusts the satellite's orientation, ensuring that key components, such as antennas, sensors and solar panels, remain properly aligned towards their targets. By compensating for the complex interplay of forces that affect the rotational dynamics, the ADCS enables precise control of the CubeSat, thereby enhancing performance and reliability in the face of varying external disturbances and internal structural complexities [2], [3].

[1]Omar Shouman and Tamer Khattab are with IIPL Lab, Department of Electrical Engineering, Qatar University, Doha, Qatar os1902986@qu.edu.qa, tkhattab@ieee.org

[2]Mohamed Mabrok is with the Department of Mathematics and Statistics, Qatar University, Doha, Qatar m.a.mabrok@qu.edu.qa

There have been different methods for controlling the CubeSat's attitude. While nonlinear methods work accurately, they are computationally expensive and complex [4]. Moreover, they involve intricate stability analyses of the closed-loop system and the difficulty of ensuring global stability and robust performance further complicates the control design. Therefore, researchers have gone to linearize the system around the operating point [5], [6]. However, linearization is not able to capture the full dynamics and may underperform during large-angle maneuvers or in high disturbance conditions.

The key contributions of our framework are:

- We propose a framework that leverages Koopman theory applied to a quaternion representation to capture nonlinear dynamics within a higher-dimensional linear framework while integrating deep learning techniques for enhanced accuracy and efficiency.
- We derive an extended loss function that explicitly incorporates the control term, ensuring improved control performance.
- We compare the koopman-based control with the traditional linearization for CubeSat dynamics while accounting for real system constraints and disturbances.

In the upcoming sections, Section II discusses the related work. Section III describes the proposed framework, and Section IV shows the results. Finally, Section V concludes the work and suggests future directions.

II. Related Work

Spacecraft attitude control is critical for mission success, especially for CubeSats in low Earth orbit (LEO) with highly nonlinear dynamics. Both nonlinear and linear control strategies have been explored, each with their strengths and limitations.

Nonlinear methods, such as finite-time control, sliding mode, and Lyapunov-based approaches, offer robustness and can handle large disturbances, however, at the cost of higher computational demands and complex tuning. For instance, Cortés et al. [7] compared a Lyapunov-based method and a sliding mode controller (SMC) for large maneuvers, finding that SMC provides superior robustness while the Lyapunov-based method offers smooth operation with lower computational cost. Other techniques, like Prescribed Performance Control (PPC) [8] and neural adaptive control [9], further enhance disturbance rejection and adaptability but typically require more onboard processing. These methods remain promising for CubeSats, especially under bounded disturbances [10] and when performance guarantees within a finite

time are critical. However, they are computationally demanding, especially for nano satellites such as the CubeSats, and complex to tune.

Linear controllers simplify the attitude control problem by using linearized or approximate models, making them computationally efficient for resource-constrained CubeSats. Model Predictive Control (MPC) has been validated experimentally by Rodrigues et al. [11] for constraint handling, while classical PID/PD controllers, as shown by Gaber et al. [6] and Sugimura et al. [12], offer simplicity and low power consumption but may struggle with large disturbances. Similarly, Linear Quadratic Gaussian (LQG) control, which combines LQR with Kalman filtering [5], provides better noise handling at the expense of needing accurate linear models. Although linear controllers are attractive for their simplicity and efficiency, they may underperform during high disturbance conditions. In addition, when linearizing about the operating point $q = [1, 0, 0, 0]$ (which corresponds to 0 Euler angles), the quaternion normalization condition $q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$ forces the scalar component q_0 to remain constant, so that any small perturbation in q_0 is entirely determined by changes in the vector part $[q_1, q_2, q_3]$; consequently, in the linearized dynamics the row corresponding to q_0 becomes effectively zero, as detailed in [5]. This limits the controller's ability to capture the full influence of q_0 on the system's behavior and it becomes sensitive to sensor noise and disturbances.

While there are some researchers that have explored Koopman-based modeling for ADCS [13], [14], [15], [16], [17], [18], none of these studies employ a quaternion representation, nor are they specifically designed for CubeSats, a platform characterized by limited actuator resources and heightened sensitivity to disturbances. Furthermore, aside from the work by Cartocci et al. [18], most research predominantly focus on controller design, without investigating how the lifted Koopman model compares to the underlying nonlinear dynamics. Additionally, none of the studies account for environmental disturbances and sensor noise. This gap raises the question of whether a dedicated Koopman-based approach that directly encodes quaternion and angular velocity states can provide both robust performance and an accurate representation of the nonlinear dynamics inherent to CubeSat ADCS under practical considerations.

III. Proposed Framework

The Koopman operator framework is a data-driven method that transforms nonlinear dynamics into a linear representation in a high-dimensional observable space, allowing linear tools to be used for analysis and control. This approach overcomes the limitations of traditional linearization techniques that can fail under large disturbances or over a wide operating range. Figure 1 illustrates the proposed framework. In this section, we first describe the system and the fundamentals of Koopman theory, then detail the data generation process and the deep learning model.

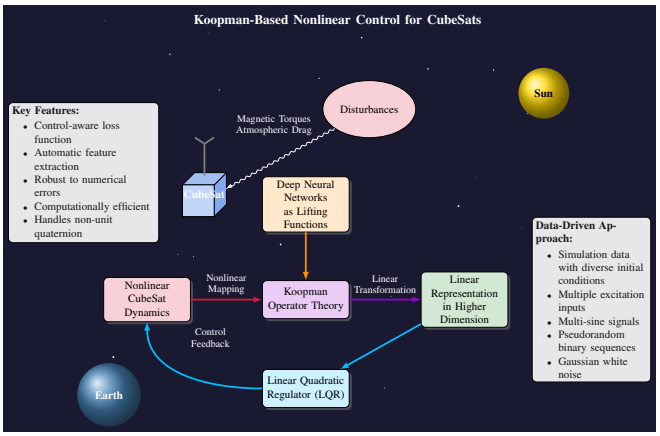

Fig. 1: Proposed Framework

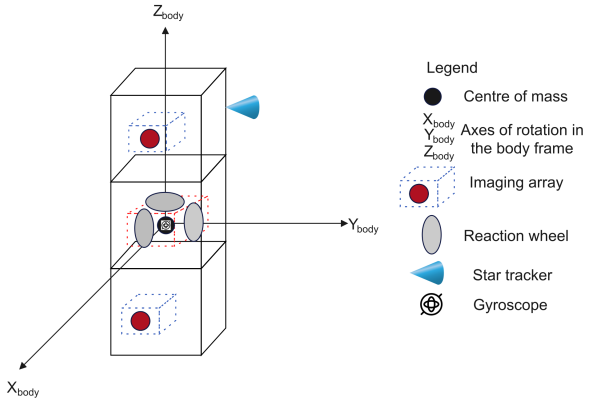

Fig. 2: Block diagram of a 3U CubeSat

A. System Dynamics Modeling

In this section, we develop a comprehensive dynamic model for a 3U CubeSat, illustrated in Figure 2, where each reaction wheel is dedicated to controlling one axis of rotation. We detail the CubeSat's overall dynamics and account for environmental disturbances, while also presenting the actuator and sensor models used for attitude determination and control.

1) CubeSat Rotational Dynamics: Quaternions offer a singularity-free, four-dimensional representation of rotations, overcoming Euler angles' limitation of gimbal lock. This ensures smooth and stable orientation tracking, which is crucial for aerospace applications like CubeSats. Despite requiring an extra parameter and a normalization step, their ability to consistently encode rotations without order dependency makes them superior for reliable and accurate attitude control.

The rotational motion of a rigid body in three-dimensional space is governed by the balance of angular momentum and external torques [19].

$$\dot{\boldsymbol{\omega}} = \mathbf{I}^{-1}\Big(\boldsymbol{\tau} - \boldsymbol{\omega} \times (\mathbf{I}\,\boldsymbol{\omega})\Big) \tag{1}$$

For many spacecraft, the spacecraft's body axes are aligned

with its principal axes of inertia, making the inertia matrix diagonal, $\mathbf{I} = \mathrm{diag}\big(I_x,\, I_y,\, I_z\big)$.
Hence, component wise, the rotational dynamics become:

$$\begin{aligned}
\dot{\omega}_x &= \frac{1}{I_x}\Big(\tau_x - (I_z - I_y)\,\omega_y\,\omega_z\Big),\\
\dot{\omega}_y &= \frac{1}{I_y}\Big(\tau_y - (I_x - I_z)\,\omega_z\,\omega_x\Big),\\
\dot{\omega}_z &= \frac{1}{I_z}\Big(\tau_z - (I_y - I_x)\,\omega_x\,\omega_y\Big)
\end{aligned} \tag{2}$$

To propagate the CubeSat's orientation without encountering the singularities of Euler angles, we use the unit quaternion $\mathbf{q} = [q_0,\ q_1,\ q_2,\ q_3]^T$. Its time derivative is:

$$\dot{\mathbf{q}} \;=\; \tfrac{1}{2}\begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix}\mathbf{q} \tag{3}$$

Using component notation $[x_1, \ldots, x_7] := [q_0, q_1, q_2, q_3, \omega_x, \omega_y, \omega_z]$, the multi-input multi-output (MIMO) system can be written as:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \\ \dot{x}_7 \end{bmatrix} = \begin{bmatrix} -\frac{1}{2}\left(x_2\,x_5 \;+\; x_3\,x_6 \;+\; x_4\,x_7\right) \\ \frac{1}{2}\left(x_1\,x_5 \;-\; x_4\,x_6 \;+\; x_3\,x_7\right) \\ \frac{1}{2}\left(x_4\,x_5 \;+\; x_1\,x_6 \;-\; x_2\,x_7\right) \\ \frac{1}{2}\left(-x_3\,x_5 \;+\; x_2\,x_6 \;+\; x_1\,x_7\right) \\ \dfrac{-\,x_6\,x_7\,(I_z - I_y) \;+\; u_1}{I_x} \\ \dfrac{-\,x_5\,x_7\,(I_x - I_z) \;+\; u_2}{I_y} \\ \dfrac{-\,x_5\,x_6\,(I_y - I_x) \;+\; u_3}{I_z} \end{bmatrix}, \tag{4}$$

where u is the control input.

2) Environmental Disturbances: The space environment imposes four main disturbance torques on CubeSat attitude [20]. Aerodynamic drag $\mathbf{F}_{\mathrm{drag}} = -\frac{1}{2}\rho V_{\mathrm{rel}}^2 C_D A \hat{V}_{\mathrm{rel}}$ (with atmospheric density ρ, relative speed V_{rel}, drag coefficient C_D, cross-sectional area A and flow direction $\hat{V}_{\mathrm{rel}}$) produces $\boldsymbol{\tau}_{\mathrm{drag}} = \mathbf{r}_{cp} \times \mathbf{F}_{\mathrm{drag}}$ if applied at a center-of-pressure offset $\mathbf{r}_{cp}$ [21], [22].

Magnetic residuals $\boldsymbol{\tau}_{\mathrm{mag}} = \mathbf{M}_r \times \mathbf{B}$ (with residual dipole moment $\mathbf{M}_r$ and geomagnetic field $\mathbf{B}$) arise from onboard magnetization [20], [23].

Gravity gradient torque $\boldsymbol{\tau}_{\mathrm{gg}} = \frac{3\mu}{r^3}\,\mathbf{r}_B \times (\mathbf{I}\,\mathbf{r}_B)$ (with Earth's gravitational parameter μ, orbital radius r, Earth-pointing unit vector $\mathbf{r}_B$ and inertia matrix $\mathbf{I}$) reflects differential gravity [20], [21].

Solar radiation pressure $\mathbf{F}_{\mathrm{srp}} = -P_{\mathrm{srp}} C_R A \hat{s}$ ($P_{\mathrm{srp}} \approx 4.5 \times 10^{-6}\,\mathrm{N/m^2}$, reflectivity C_R, Sun-vector $\hat{s}$) yields $\boldsymbol{\tau}_{\mathrm{srp}} = \mathbf{r}_{\mathrm{srp}} \times \mathbf{F}_{\mathrm{srp}}$ with SRP center-of-pressure offset $\mathbf{r}_{\mathrm{srp}}$ [23], [24].

These disturbance models, with parameters chosen based on a 3U CubeSat, 500 km polar orbits, and literature [20], [21], [22], are essential for accurate modeling of a CubeSat environment.

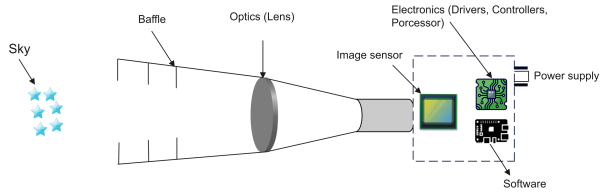

Fig. 3: A conceptual diagram of a star tracker used to determine the satellite's attitude.

3) Actuators and Sensors: A reaction wheel produces torque by exchanging angular momentum: for single-axis $\tau_{rw} = -\frac{d}{dt}(I_w\,\omega_w\,\hat{u}^w)$ (wheel inertia I_w, spin rate ω_w, spin-axis unit vector $\hat{u}^w$, so $\mathbf{h}_w = I_w\,\omega_w\,\hat{u}^w$); for n wheels $\tau_{\mathrm{total}} = -\sum_{i=1}^{n}\frac{d}{dt}(I_{w,i}\,\omega_{w,i}\,\hat{u}_i^w)$. To model the eraction wheel in simulation, a saturation $\tau_{\max} = 1 \times 10^{-4}\,\mathrm{N\cdot m}$ is imposed [23].

Gyroscopes measure angular velocity via $\omega_{\mathrm{meas}} = \omega_{\mathrm{true}} + b(t) + n(t)$, where $b(t)$ is a bias (set to $\sim 1\times10^{-4}\,\mathrm{rad/s}$) and $n(t)$ is additive white Gaussian noise (AWGN) with $\sigma_n^2 \sim (1 \times 10^{-3}\,\mathrm{rad/s})^2$ [25].

Star trackers, illustrated in Figure 3, are optical sensors that capture images of the star field and provide absolute attitude measurements, typically in the form of quaternions. They do so by collecting incoming light from stars through a baffle (which blocks stray light) and focusing it onto an image sensor (CMOS or CCD) via a lens or other optical elements. The onboard processor and software then detect star centroids in the digital image, identify them by matching against a known star catalog, and finally compute the spacecraft's attitude (orientation). This orientation is expressed as a rotation from the inertial star field to the spacecraft body frame, and is commonly output as a quaternion.

Their measurement error is modeled by $\hat{q} = \delta q \otimes q_{\mathrm{true}}$, where $\delta q \approx \begin{pmatrix} 1 \\ \frac{1}{2}\epsilon \end{pmatrix}$, $\epsilon \sim \mathcal{N}(0, \sigma_q^2 I_{3\times3})$. Off-shelf startrackers have a typical errors of 30–70 arcsec (roughly 0.008° and 0.056° in pitch/yaw and roll, respectively) [26].

B. Koopman Theory Preliminaries

In this subsection, we review the fundamentals of koopman operator theory, a detailed explanation can be found in [27]. Consider a discrete-time nonlinear system of the form:

$$\begin{cases} \mathbf{x}(k+1) \;=\; f\big(\mathbf{x}(k),\, \mathbf{u}(k)\big), \\ \mathbf{y}(k) \;=\; g\big(\mathbf{x}(k)\big), \end{cases} \tag{5}$$

where $\mathbf{x}(k) \in \mathbb{R}^n$ is the state, $\mathbf{u}(k) \in \mathbb{R}^m$ is the control input, $\mathbf{y}(k) \in \mathbb{R}^p$ is the output, and the functions $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^p$ are potentially nonlinear.
To combine states and inputs, define the *extended state*

$$\mathbf{z}(k) \;=\; \begin{bmatrix} \mathbf{x}(k) \\ \mathbf{u}(k) \end{bmatrix} \;\in\; \mathbb{R}^{n+m}$$

We assume there is a map $\mathcal{F} : \mathbb{R}^{n+m} \to \mathbb{R}^{n+m}$ such that

$$\mathbf{z}(k+1) \;=\; \mathcal{F}\big(\mathbf{z}(k)\big) \tag{6}$$

Now, let $\phi_{\text{inf}}(z)$ denote the infinite-dimensional lifting function described as

$$\phi_{\text{inf}}(z) = \begin{bmatrix} \phi_1(z) \\ \phi_2(z) \\ \vdots \end{bmatrix} \tag{7}$$

Each entry of ϕ is a function of the current extended state. Now, we introduce the Koopman operator $\mathcal{K}$ as

$$\mathcal{K}(\phi_{\text{inf}}(z)) := \phi_{\text{inf}}(\mathcal{F}(z)) \tag{8}$$

By combining eqs (6) and (8), the time evolution of $\phi_{\text{inf}}(z)$ can be described by

$$\phi_{\text{inf}}(z(k+1)) = \mathcal{K}(\phi_{\text{inf}}(z(k))) \tag{9}$$

This operator is linear with respect to ϕ, despite $\mathcal{F}$ being nonlinear. However, $\mathcal{K}$ typically resides in an infinite-dimensional space, necessitating finite-dimensional approximations for practical use.

1) Finite-dimensional approximation : Now, since the infinite-dimensional operator is purely theoretical, eq (7) can be represented as:

$$\phi(\mathbf{z}) \;=\; \begin{bmatrix} \phi_1(\mathbf{z}) \\ \phi_2(\mathbf{z}) \\ \vdots \\ \phi_N(\mathbf{z}) \end{bmatrix} \in \mathbb{R}^N \tag{10}$$

To approximate $\mathcal{K}$ numerically, we seek a matrix $\mathbf{K}$ such that

$$\phi\big(\mathbf{z}(k+1)\big) \;\approx\; \mathbf{K}\,\phi\big(\mathbf{z}(k)\big) \tag{11}$$

Now, the lifting function is defined based on the state as:

$$\phi(z) \;=\; \psi(x(k)) \in \mathbb{R}^N \tag{12}$$

where $\psi(x) : \mathbb{R}^n \to \mathbb{R}^N$ is an N-dimensional lifting function given by

$$\psi(x) \;=\; \begin{bmatrix} \psi_1(x) \\ \vdots \\ \psi_N(x) \end{bmatrix} \in \mathbb{R}^N. \tag{13}$$

Common choices for ψ_i include polynomials, radial basis functions, or neural-network-based features, all aiming to capture nonlinear dynamics in a higher-dimensional space. In terms of control, the system in eq (5) can be described in the lifted domain, where it appears linear, as

$$\begin{cases} \psi(x(k+1)) = A\psi(x(k)) + Bu(k) \\ y(k) = C\psi(x(k)) \end{cases} \tag{14}$$

Another representation is the bilinear Koopman control, where the effects of control inputs are explicitly modeled as state-dependent, resulting in a formulation of the form [28]

$$z_{k+1} = Az_k + \sum_i u_{i,k} B_i z_k, \tag{15}$$

which inherently captures the nonlinear interaction between the state and the input. Both this bilinear approach and the Koopman operator with additive control, shown in eq. (14), have been employed to model nonlinear systems in a higher linear manifold [13], [14], [28], [29]. The Koopman operator with additive control is more common in ADCS [13], [14], [30], [15] due to its simplicity and computational efficiency, which is particularly advantageous for the limited onboard processing capabilities of CubeSats, and because it integrates seamlessly with established linear control designs such as LQR and MPC [31].

To sum up, Koopman theory enables us to derive a linear approximation of the otherwise nonlinear system in a higher-dimensional space of observables. This lifted system can utilize classical linear control techniques.

C. Data Generation

To learn the nonlinear dynamics of our CubeSat system, we require a diverse set of states and inputs trajectories in ideal scenario (i.e. no disturbances). We generated the dataset by integrating the system in (4) using Simulink® with an appropriate ODE solver. The rotational dynamics were governed by a diagonal inertia matrix $\mathbf{I} = \mathrm{diag}([0.03,\ 0.03,\ 0.006])\ \mathrm{kg}\cdot\mathrm{m}^2$, reflecting the asymmetric mass distribution of a 3U CubeSat. We performed 1000 independent simulations to capture a wide variety of initial conditions and input trajectories. Each simulation had a trajectory length of 5 s and a sampling time of 0.01 s.

Each control input $\mathbf{u}(k) = \begin{bmatrix} u_1(k) & u_2(k) & u_3(k) \end{bmatrix}^\top$ is drawn at random from a set of three excitation types: Multi-Sine, PRBS (Pseudorandom Binary Sequence), and Gaussian White Noise. The Multi-Sine signal is a sum of sinusoids with randomly selected amplitudes, frequencies, and phases, providing broadband excitation. The PRBS input is a step-like signal that switches levels in a pseudo-random pattern, ensuring rich frequency content for system identification [32], while the Gaussian White Noise input offers another form of excitation. Each axis (X, Y, Z) is assigned one of these input types via a random permutation, ensuring variability across different runs.

To cover a wide region of the attitude state space, each simulation starts with random initial conditions. The orientation is initialized with random Euler angles (ϕ, θ, ψ), each sampled within $\pm 360^\circ$ and then converted to a quaternion (q_0, q_1, q_2, q_3). Additionally, the initial angular velocities (ω_x, ω_y, ω_z) are randomly sampled within $\pm 1\,\mathrm{rad/s}$. By introducing variance in both the initial orientation and spin rates, each trajectory captures distinct operating conditions.

D. Deep Learning Model

We attempted to implement Gaussian Radial Basis Functions (RBFs) and found that even using 100 RBFs was insufficient to effectively capture the system's nonlinear dynamics, leading to an optimization problem too complex to solve efficiently (even with a 4070 RTX GPU). Furthermore, even with access to more powerful hardware, the resulting high-dimensional state space would render the control process complex.

Thus, to efficiently capture the system's complex nonlinear behavior, we employ a deep neural network as a lifting function, denoted by $\psi_{NN}(\cdot) \in \mathbb{R}^N$. The network is trained to extract a rich feature representation of the system's nonlinear dynamics. Now we will derive the loss function that will be used in training.

In the standard, unforced Koopman-operator framework, we assume that the lifted state evolves as

$$\psi(x_{k+1}) = K\,\psi(x_k)$$

By iterating this recurrence, we obtain

$$\psi(x_{m+1}) = K^m\,\psi(x_1)$$

Thus, for a prediction horizon S_p, one may define the prediction loss as [33]

$$\mathcal{L}_{\text{pred}} = \frac{1}{S_p}\sum_{k=1}^{S_p}\Big\|x_{k+1} - \psi^{-1}\Big(K^k\,\psi(x_1)\Big)\Big\|_{\text{MSE}}, \quad (16)$$

where ψ^{-1} denotes the inverse mapping from the lifted space back to the original state space. However, this representation does not account for the control input. In addition instead of training a decoder NN, we train a matrix C for reconstruction of the original states for a less computationally demanding model.

We now derive the control version of the loss function in (16). When a control input u_k is present, the lifted dynamics become

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

One-step substitutions yield:

$$\begin{aligned}\psi(x_2) &= A\,\psi(x_1) + B\,u_1,\\ \psi(x_3) &= A\,\psi(x_2) + B\,u_2\\ &= A^2\,\psi(x_1) + A\,B\,u_1 + B\,u_2,\end{aligned}$$

this implies, the general relation:

$$\psi(x_{k+1}) = A^k\,\psi(x_1) + \sum_{i=1}^{k} A^{k-i}\,B\,u_i.$$

Thus, the prediction loss in the controlled setting is defined as:

$$\mathcal{L}_{\text{pred}} = \frac{1}{S_p}\sum_{k=1}^{S_p}\Big\|x_{k+1} - C\left(A^k\,\psi(x_1) + \sum_{i=1}^{k} A^{k-i}\,B\,u_i\right)\Big\|_{\text{MSE}} \quad (17)$$

Now we will define the remaining loss terms.

1) Reconstruction Loss: To ensure that the lifting function ψ_{NN} accurately captures the state, we define the reconstruction loss as:

$$\mathcal{L}_{\text{recon}} = \Big\|x_k - C\,\psi_{\text{NN}}(x_k)\Big\|_{\text{MSE}} \quad (18)$$

2) Linearity Loss: To encourage the latent dynamics to follow the linear model, we define the linearity loss as:

$$\mathcal{L}_{\text{lin}} = \frac{1}{T-1}\sum_{k=1}^{T-1}\Big\|\psi_{\text{NN}}(x_{k+1}) - \Big(A\,\psi_{\text{NN}}(x_k) + B\,u_k\Big)\Big\|_{\text{MSE}} \quad (19)$$

with T being the total number of time steps in the trajectory.

3) Total Loss Function: The overall loss is a weighted sum of the above terms along with an L_2 regularization term to control the magnitude of the weights:

$$\mathcal{L} = \alpha_1\Big(\mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{pred}}\Big) + \mathcal{L}_{\text{lin}} + \alpha_2\|\mathbf{W}\|_2^2, \quad (20)$$

where $\|\mathbf{W}\|_2^2$ is the regularization term, the squared L_2-norm of the trainable weight matrices, and α_1 and α_2 are scalar weighting coefficients.

4) Training Procedure: The datasets were partitioned into an 80/10/10 split for training, validation, and testing, yielding 800 training trajectories and 100 trajectories each for testing and validation. All matrices, A, B, and C are treated as trainable parameters during training. In addition, each weight matrix W is initialized from a uniform distribution over the interval $[-s, s]$, where $s = \frac{1}{\sqrt{a}}$, and a is the dimension of the input to the layer (i.e., number of neurons in the previous layer). This initialization strategy, which helps maintain proper scaling of the activations, was recommended in [33]. Hyperparameters were tuned using a grid search algorithm.

After training, the model is evaluated on a the test set using two performance metrics: (a) **Iterative Koopman prediction:** Starting with $z_0 = \psi(x_0)$ and given a sequence $\{u_0, u_1, u_2, \dots\}$, the lifted states are computed iteratively as $z_1 = Az_0 + Bu_0$, $z_2 = Az_1 + Bu_1$, $z_3 = Az_2 + Bu_2$, etc. (b) **First-order linear state-space prediction:** Directly in state space, starting with x_0, the prediction is given by $x_1 = Fx_0 + Gu_0$, then $x_2 = Fx_1 + Gu_1$, then $x_3 = Fx_2 + Gu_2$, etc. These predicted trajectories are compared to the true trajectory to assess model performance.

These two metrics serve distinct yet complementary roles. The iterative Koopman prediction evaluates the model's ability to propagate the system's inherently nonlinear dynamics over multiple steps solely via the learned linear operators in the lifted space, thereby revealing its long-horizon accuracy, stability under compounding errors, and its utility for receding-horizon predictive control design. In contrast, the first-order linear state-space prediction offers a direct benchmark against a classical linearization, quantifying the performance improvement afforded by the Koopman lifting and ensuring that the additional modeling complexity yields tangible gains over standard linear approximations on the same test trajectories.

IV. Results

Various architectures, activation functions, and hyperparameters were evaluated, though only the final results are presented due to space constraints. Our proposed neural network is a fully connected feedforward model that begins with a 7-dimensional input, followed by five dense hidden layers with 200 neurons each and tanh activations to introduce nonlinearity, before branching into the two variants (N32 and N64), as shown in Figure 4.

Table I shows the test-set multi-step prediction error (MSE) for each model over 20, 50, and 100-step horizons, as well as the final hyperparameter settings used in training. The N=64 variant attains the best performance, with MSEs

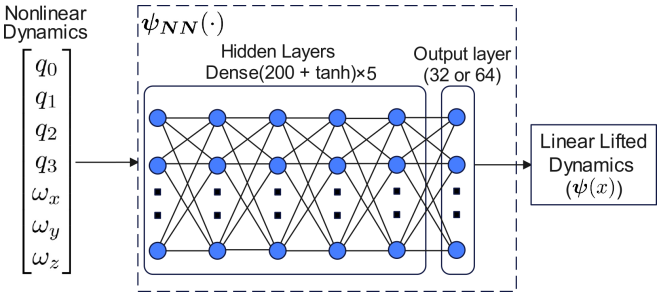

Fig. 4: Neural Network architecture

TABLE I: Performance Metrics and Hyperparameters

Model	**20-Step**	**50-Step**	**100-Step**
Iterative Koopman (N=64)	8.93×10^{-4}	2.74×10^{-3}	2.77×10^{-2}
Iterative Koopman (N=32)	3.43×10^{-3}	2.38×10^{-2}	1.03×10^{-1}
Linear baseline (F,G)	1.08×10^{-2}	6.27×10^{-2}	1.34×10^{-1}
Hyperparameters	$\alpha_1 = 0.5, \quad \alpha_2 = 10^{-7}$ $S_p = 50, \quad \text{Activation} = \tanh$		

of 8.93×10^{-4}, 2.74×10^{-3}, and 2.77×10^{-2} at 20, 50, and 100 steps, respectively. The N=32 model yields moderate accuracy ($3.43{\times}10^{-3}$, $2.38{\times}10^{-2}$, $1.03{\times}10^{-1}$), whereas the linear baseline incurs substantially higher errors ($1.08{\times}10^{-2}$, 6.27×10^{-2}, 1.34×10^{-1}).

To put these results in perspective, Cartocci et al. [18] report a 100-step RMSE of 1.75 (MSE ≈ 3.06) using EDMD with a 9^3 (729-dimensional) RBF lifting basis. In contrast, our $N = 64$ model achieves a 100-step MSE of 2.77×10^{-2} (RMSE ≈ 0.17) using only 64 learned lifting dimensions, over an order of magnitude lower error with far fewer observables, underscoring the efficiency and expressiveness of neural lifting functions.

A. Linear Controller

In this section, we develop a Koopman-based linear controller and a conventional controller for the model linearized about the nominal operating point, and compare their performances under the disturbances described in Section III-A.2. Consider a discrete-time linear system

$$\mathbf{x}_{k+1} \;=\; A\,\mathbf{x}_k + B\,\mathbf{u}_k,$$

where $\mathbf{x}_k \in \mathbb{R}^n$ and $\mathbf{u}_k \in \mathbb{R}^m$. The Linear Quadratic Regulator (LQR) seeks to regulate the system to the equilibrium, $\mathbf{x} = \mathbf{0}$, by minimizing the infinite-horizon cost

$$J \;=\; \sum_{k=0}^{\infty}\Big(\mathbf{x}_k^\top\, Q_x\,\mathbf{x}_k \;+\; \mathbf{u}_k^\top\, R\,\mathbf{u}_k\Big), \tag{21}$$

where $Q_x \succeq 0$ and $R \succ 0$ are weighting matrices. In our experiments, we set $Q_x = 2I$ and $R = I$, where I denotes the identity matrix.

The solution is given by the feedback law $\mathbf{u}_k \;=\; -K\,\mathbf{x}_k$, with K obtained by solving the Discrete Algebraic Riccati Equation (DARE) for (A, B, Q_x, R). Note that in practice, the classical LQR is applied to a linearized model about the equilibrium $\mathbf{x}_{eq} = [1, 0, 0, 0, 0, 0, 0]^\top$; hence, the state $\mathbf{x}$ here represents the deviation from that equilibrium.

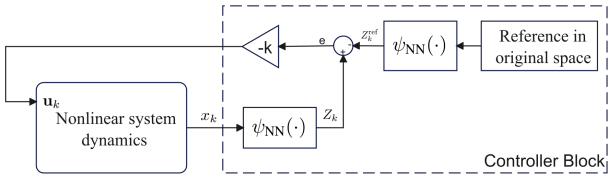

Fig. 5: LQR Controller

In our Koopman framework, the nonlinear dynamics are lifted into a higher-dimensional space via a neural network $\psi_{\mathrm{NN}}(\cdot)$. In this lifted space, the desired equilibrium becomes

$$\mathbf{z}^{\mathrm{ref}} = \psi_{\mathrm{NN}}(\mathbf{x}_{eq})$$

We first lift the state cost matrix using $Q_z = C^\top Q_x\, C$. Then the cost in eq (21) is expressed in the lifted space as

$$J \;=\; \sum_{k=0}^{\infty}\Big(\left(\mathbf{z}_k - \mathbf{z}^{\mathrm{ref}}\right)^\top Q_z\left(\mathbf{z}_k - \mathbf{z}^{\mathrm{ref}}\right) \;+\; \mathbf{u}_k^\top\, R\,\mathbf{u}_k\Big) \tag{22}$$

Solving the DARE for (A, B, Q_z, R) yields a feedback gain K. The resulting Koopman-based LQR control law is

$$\mathbf{u}_k \;=\; -\,K\left(\mathbf{z}_k - \mathbf{z}^{\mathrm{ref}}\right). \tag{23}$$

Figure 5 shows the controller block diagram.

Figure 6 shows the reference tracking performance for CubeSat missions using both the linearized model around the operating point and the proposed Koopman model under sensor noise and disturbances. The linearized model lacks the corrective nonlinear dynamics required to enforce quaternion normalization. As a result, if sensor noise perturbs the quaternion (particularly q_0) away from unit norm, the linearized system is unable to drive the state back onto the unit sphere because the scalar component q_0 is implicitly fixed by the linear approximation. Even small deviations from unit norm are significant in quaternion representation. Any drift off the unit sphere directly translates into a non-physical rotation and growing attitude error. This deficiency leads to degraded tracking performance, whereas the Koopman model preserves the unit-norm property and maintains robust performance.

V. Conclusion & Future Work

In this work, we have introduced a data-driven framework for CubeSat attitude determination and control that integrates Koopman theory with deep learning. By lifting the complex, nonlinear dynamics of low-inertia CubeSats into a higher-dimensional latent space via a deep neural network, we achieve a linear representation that accurately reflects the underlying nonlinear behavior. This Koopman-based model facilitates the use of classical linear control methods, as evidenced by our implementation of a discrete-time LQR controller in the latent space, and exhibits superior robustness by preserving key properties (e.g., quaternion unit

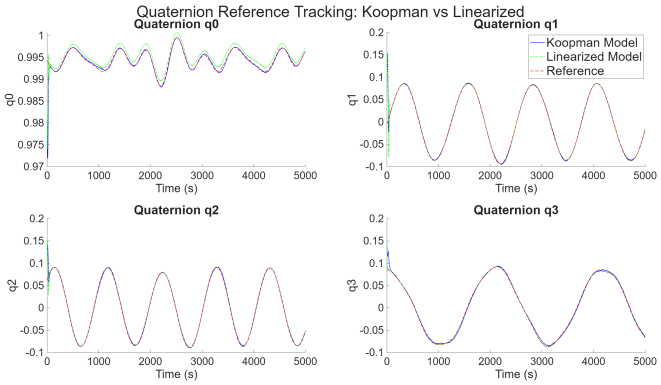

Fig. 6: Reference tracking: Koopman Vs Linearized model

norm) under sensor noise and disturbances. Additionally, by incorporating control inputs into the training loss, the latent dynamics are directly aligned with controller design, providing a robust and computationally efficient approach.

Future research may integrate advanced control schemes like Model Predictive Control directly in the lifted space. Furthermore, practical implementation through Hardware-In-the-Loop (HIL) testing will be essential to bridge the gap between simulation and real-world performance.

VI. ACKNOWLEDGMENT

The work of O.Shouman was supported by Qatar University Graduate Assistance-ship No. 450. The work of T. Khattab is supported by QRDI QNRF grant number AICC03-0530-200033.

REFERENCES

[1] A. Poghosyan and A. Golkar, "Cubesat evolution: Analyzing cubesat capabilities for conducting science missions," *Progress in Aerospace Sciences*, vol. 88, pp. 59–83, 2017.

[2] M. L. Orozco and B. S. Giraldo, "Attitude determination and control in small satellites: A review," *IEEE Journal on Miniaturization for Air and Space Systems*, 2024.

[3] T.-Y. Lin, J.-C. Juang, *et al.*, "Design and verification of the operating procedure of attitude determination and control subsystem of a nanosatellite," in *2014 CACS International Automatic Control Conference (CACS 2014)*, pp. 51–56, IEEE, 2014.

[4] J. Mamarasulov, J. Holaza, and M. Gulan, "Design of an mpc based attitude control system for a cubesat nanosatellite," in *2021 23rd International Conference on Process Control (PC)*, pp. 266–271, IEEE, 2021.

[5] H. Bano and B. Sajid, "Attitude estimation & control of a cubesat using linear quadratic gaussian approach," in *IEEE Aerospace Conference*, 2021.

[6] K. Gaber, M. B. E. Mashade, and G. A. A. Aziz, "High-precision attitude determination and control system design and real-time verification for cubesats," *International Journal of Communication Systems*, vol. 33, 2020.

[7] E. D. C. G., E. Rosero, and G. W. R. P., "Non-linear control strategies for attitude maneuvers of a leo cubesat based on modified rodrigues parameters," in *IEEE 6th Colombian Conference on Automatic Control (CCAC)*, IEEE, 2023.

[8] C. Wei, Q. Chen, J. Liu, Z. Yin, and J. Luo, "An overview of prescribed performance control and its application to spacecraft attitude system," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 234, no. 8, pp. 1–13, 2020.

[9] Q. Yao, "Neural adaptive attitude tracking control for uncertain spacecraft with preassigned performance guarantees," *Advances in Space Research*, vol. 71, pp. 3552–3564, 2023.

[10] A. Li, A. Astolfi, and M. Liu, "Attitude regulation with bounded control in the presence of large disturbances with bounded moving average," *IEEE/CAA Journal of Automatica Sinica*, vol. 9, no. 5, pp. 834–846, 2022.

[11] R. Rodrigues, A. Murilo, R. V. Lopes, and L. C. G. de Souza, "Hardware in the loop simulation for model predictive control applied to satellite attitude control," *IEEE Access*, vol. 7, pp. 157401–157417, 2019.

[12] N. Sugimura, T. Kuwahara, and K. Yoshida, "Attitude determination and control system for nadir pointing using magnetorquer and magnetometer," in *IEEE Aerospace Conference*, 2016.

[13] J. Yao, Q. Hu, and J. Zheng, "Koopman-operator-based safe learning control for spacecraft attitude reorientation with angular velocity constraints," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 59, no. 5, pp. 7072–7086, 2023.

[14] A. Chen, S. Bu, L. Cui, W. Zhang, J. Li, and Z. Wang, "Robust data-driven attitude control for satellite using koopman operator theory," in *IEEE 13th Data Driven Control and Learning Systems Conference (DDCLS)*, pp. 1417–1424, IEEE, 2024.

[15] J. Yao, X. Wang, Q. Hu, and J. Zheng, "Koopman-operator-based model predictive control for constrained spacecraft attitude reorientation," in *China Automation Congress (CAC)*, IEEE, 2023.

[16] K. Zheng, P. Huang, and G. P. Fettweis, "Optimal control of quadrotor attitude system using data-driven approximation of koopman operator," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 834–840, 2023.

[17] W. A. Manzoor, S. Rawashdeh, and A. Mohammadi, "Koopman operator-based data-driven identification of tethered subsatellite deployment dynamics," *Journal of Aerospace Engineering*, vol. 36, no. 4, p. 04023021, 2023.

[18] N. Cartocci, A. Monarca, G. Costante, M. L. Fravolini, K. M. Dogan, and T. Yucelen, "Linear control of a nonlinear aerospace system via extended dynamic mode decomposition," in *AIAA Scitech 2022 Forum*, p. 2046, 2022.

[19] H. Schaub and J. L. Junkins, *Analytical mechanics of space systems*. Aiaa, 2003.

[20] J. R. Wertz, *Spacecraft Attitude Determination and Control*. D. Reidel Publishing Company, 1978.

[21] D. A. Vallado, *Fundamentals of Astrodynamics and Applications*. Kluwer Academic Publishers, 2001.

[22] H. Choi, S. Kim, and Y. Lee, "Cubesat aerodynamics: Center-of-pressure estimation for low earth orbit," in *Proceedings of the 67th International Astronautical Congress*, (Jeju, Korea), pp. 345–350, International Astronautical Federation, 2016.

[23] M. J. Sidi, *Spacecraft Dynamics and Control: A Practical Engineering Approach*. Cambridge University Press, 1997.

[24] J. R. Wertz, D. F. Everett, and J. J. Puschell, *Space Mission Engineering: The New SMAD*. Microcosm Press, 4th ed., 2011.

[25] F. Dell'Olio, T. Natale, Y.-C. Wang, and Y.-J. Hung, "Miniaturization of interferometric optical gyroscopes: A review," *IEEE Sensors Journal*, vol. 23, no. 24, pp. 29948–29959, 2023.

[26] AAC Clyde Space, *ST200 CubeSat Star Tracker Datasheet*. AAC Clyde Space, Glasgow, UK, 2021. Document No. ST200-1, Rev. 1.

[27] K. Hara, M. Inoue, and N. Sebe, "Learning koopman operator under dissipativity constraints," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 1169–1174, 2020.

[28] 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)*, pp. 1344–1350, IEEE, 2022.

[29] W. Zhang and J.-S. Li, "Koopman bilinearization of nonlinear control systems," *arXiv preprint arXiv:2211.07112*, 2022.

[30] V. Zinage and E. Bakolas, "Koopman operator based modeling and control of rigid body motion represented by dual quaternions," in *2022 American Control Conference (ACC)*, pp. 3997–4004, AACC, 2022.

[31] M. Korda and I. Mezić, "Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control," *Automatica*, vol. 93, pp. 149–160, 2018.

[32] K. J. Keesman, *System identification: an introduction*. Springer Science & Business Media, 2011.

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