Design and Data-Driven Identification of a Quadruped Robot

Dakota Rufino
drujino@clemson.edu

Follow this and additional works at: https://tigerprints.clemson.edu/all_theses

Part of the Acoustics, Dynamics, and Controls Commons

Recommended Citation
Rufino, Dakota, "Design and Data-Driven Identification of a Quadruped Robot" (2023). All Theses. 3984. https://tigerprints.clemson.edu/all_theses/3984

This Thesis is brought to you for free and open access by the Theses at TigerPrints. It has been accepted for inclusion in All Theses by an authorized administrator of TigerPrints. For more information, please contact kokeefe@clemson.edu.
DESIGN AND DATA-DRIVEN IDENTIFICATION
OF A QUADRUPED ROBOT

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Mechanical Engineering

by
Dakota Leigh Rufino
May 2023

Accepted by:
Dr. Umesh Vaidya, Committee Chair
Dr. Ardalan Vahidi
Dr. Yue Wang
Abstract

The existence of nonlinearities and the lack of precise or sufficient equations are fundamental challenges in modeling, analyzing, and controlling complex dynamical systems. However, recent developments in numerical tools and algorithms from data science are revolutionizing the study of dynamical systems. An emerging method in nonlinear dynamical systems is the Koopman operator theory, which represents nonlinear dynamical systems as a linear system in a lifted space. The Koopman operator provides us with key advantages in performing the modeling, prediction, and control of nonlinear systems. The linear system representation allows us to leverage linear stability analysis. First, this thesis briefly covers the construction of a quadrupedal robot, a sufficiently complex nonlinear dynamical system, for the use of analyzing data-driven modeling techniques. Next, we detail the physics based dynamic model of this quadruped, using linearized single rigid body dynamics and then we provide the data-driven Koopman models using DMDc. We are able to show that the physics based model fails to capture the dynamics of the system, whereas the Koopman models can accurately forecast the nonlinear response of the system over a finite time horizon. Lastly, we propose an experimental framework to obtain a data-driven Koopman model of a quadrupeds leg dynamics over deformable terrains as a switched system. Results show that the Koopman generator has a unique spectrum associated with each terrain, making terrain classification possible, using only proprioceptive sensors. Through the methods presented in this thesis, we are able to show that the data-driven models of dynamical systems using Koopman operator theory can sufficiently approximate the nonlinearities of the system and can accurately predict future trajectories of the system.
Dedication

I dedicate this thesis to my friends, sisters, and brother who have always believed in my. Thank you for always being there to listen to my complaints and pretend to be interested in my research.

To my precious cats. Your constant presence and affection have brought joy and comfort throughout this journey. Thank you for being by my side.

And most importantly, I dedicate this thesis to my parents. Your unwavering support and encouragement have been a constant source of motivation. I am forever grateful for everything you have done for me. I hope I’ve made you proud.
Acknowledgments

I would like to express my deepest appreciation to all those who have helped me in the completion of this thesis. First and foremost, I would like to thank the professors on my committee for their engaging and inspiring teaching. Their teachings have been a highlight of my academic journey and have enriched my understanding of controls and dynamical systems. Their expertise and insights have been instrumental in shaping my research.

I would also like to extend my gratitude to my colleagues and friends for their guidance and encouragement throughout this journey. It has been a pleasure to work alongside such talented and dedicated individuals. I am incredibly grateful for the memories we have shared and the bonds we have formed.
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title Page</td>
<td>i</td>
</tr>
<tr>
<td>Abstract</td>
<td>ii</td>
</tr>
<tr>
<td>Dedication</td>
<td>iii</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>iv</td>
</tr>
<tr>
<td>List of Tables</td>
<td>vi</td>
</tr>
<tr>
<td>List of Figures</td>
<td>vii</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Koopman Operator Theory</td>
<td>3</td>
</tr>
<tr>
<td>1.2 Data-driven Approximation of Koopman Operator</td>
<td>3</td>
</tr>
<tr>
<td>2 The Clemson Tigger</td>
<td>9</td>
</tr>
<tr>
<td>2.1 System Requirements</td>
<td>9</td>
</tr>
<tr>
<td>2.2 Comparisons of Legged Robots</td>
<td>10</td>
</tr>
<tr>
<td>2.3 Quadruped Description</td>
<td>11</td>
</tr>
<tr>
<td>3 System Modeling</td>
<td>18</td>
</tr>
<tr>
<td>3.1 Experimental Methods</td>
<td>18</td>
</tr>
<tr>
<td>3.2 Rigid Body Dynamics Model</td>
<td>20</td>
</tr>
<tr>
<td>3.3 Koopman DMDc Model</td>
<td>24</td>
</tr>
<tr>
<td>4 Data-Driven System Identification for Terrain</td>
<td>35</td>
</tr>
<tr>
<td>4.1 Problem Background</td>
<td>35</td>
</tr>
<tr>
<td>4.2 Switched Systems Model</td>
<td>36</td>
</tr>
<tr>
<td>4.3 Experimental Methods</td>
<td>37</td>
</tr>
<tr>
<td>4.4 Data-Driven System Model</td>
<td>39</td>
</tr>
<tr>
<td>4.5 Results</td>
<td>43</td>
</tr>
<tr>
<td>5 Conclusions and Discussion</td>
<td>46</td>
</tr>
<tr>
<td>5.1 Future Research</td>
<td>47</td>
</tr>
<tr>
<td>Bibliography</td>
<td>48</td>
</tr>
</tbody>
</table>
List of Tables

4.1 Proportional gains $K_p$ for Stance Phase .............................. 39
List of Figures

1.1 The finite dimensional Koopman operator acts on observable functions of a nonlinear system and propagates them linearly forward in time in the lifted dimension $\mathbb{R}^N$. These observables are mapped back to the original state space $\mathbb{R}^n$ to recover $x$ at the next time step. .................. 7

2.1 The Clemson Tigger quadrupedal robot, based on the Stanford Doggo . . 11
2.2 Coaxial drive assembly. The different colors indicate the independently driven leg linkages and their corresponding motors [16] ......................... 12
2.3 Electrical Diagram ...................................... 13
2.4 Piecewise sinusoidal leg trajectories for flight (left) and stance phases (right) of gait motion. Trajectories are ..................................... 14
2.5 Control Architecture for the Clemson Tigger .............................. 15
2.6 Parameters of the 5-bar mechanism for the leg assembly, whose generalized coordinates are $\gamma$ and $\theta$. These parameters are used to define the Jacobian for the inverse kinematics. .......................................... 16
2.7 Generalized coordinates for the wheeled configuration of Clemson Tigger . 17

3.1 Decomposition of a matrix using RPCA [32] ............................. 20
3.2 Single rigid body dynamics of a quadruped in the body and world frames. 21
3.3 System response of the SRBD with discrete inputs. The dynamical model fails to converge to the states of the experimental data with the same inputs 23
3.4 System response of the SRBD with discrete inputs. The dynamical model fails to converge to the states of the experimental data with the same inputs 24
3.5 System response of the Koopman DMDc with SVD algorithm for angular velocity and linear acceleration ........................................ 30
3.6 System response of the Koopman Robust DMDc algorithm for angular velocity and linear acceleration ........................................ 31
3.7 System response of the Koopman Henkle DMDc algorithm for angular velocity and linear acceleration ........................................ 32
3.8 Eigenvalues of the DMDc data driven models for quadruped dynamics. The eigenvalues for each method are shown to be stable. The eigenvalues of the HDMDc (right) are presented in the lifted space of $\mathbb{R}^{n(d+1)}$ ............... 33

4.1 The soil test bed with several terrain types used to collect data during the stance phase experiments [22]. ................................. 39
4.2 (Left) Filtered DMDc predicted state response for the switched dynamical system predicting $N = 5$ time steps (200ms) in the future with a time discretization of $\Delta t = 40\text{ms}$. (Right): Phase portrait of $\theta$ and $\gamma$ obtained for different terrain [22].

4.3 Occupancy maps of the state space on varying terrains[22].

4.4 Eigenvalues of the generator $\mathbf{L}_s$ associated with the each terrain type. (using a $\gamma$ gain of $K_p = 30$).

5.1 Terrabot: A new quadruped for data-driven identification.
Chapter 1

Introduction

During the second half of the twentieth century, engineers shifted away from analyzing linear systems using frequency response, and began using state-space modeling to address issues with more complex linear systems. This modern formulation of linear systems provided representations for both external and internal state of the systems. The spectral decomposition then reveals the true nature of the system, and is used to characterize the internal and external properties of the system. Furthermore, this modern approach to linear systems allows for the application of linear control system strategies. The optimal control solutions are easily obtainable, as opposed to the nonlinear optimal control solutions associated with the Hamilton Jacobi Bellman partial differential equations. The theory of linear systems has been vastly explored and has established a paradigm for explicitly characterizing linear system; however, there does not yet exist a similar analytical framework for the characterization of nonlinear systems.

Nonlinearity is a fundamental challenge in modeling, analyzing, and controlling dynamical systems. It is well known that a variety of unpredictable behaviors that can be exhibited in nonlinear systems, analysis techniques for the linear systems do not translate to nonlinear systems. Moreover, the lack of precise or sufficient equations for systems is a problem prevalent in many real-world applications. Deriving and solving nonlinear dynamical equations often is met with difficulties, leading to the approximation of these
systems using first order approximations. Linearization offers an accurate model for a
given, often restrictive, condition, but accuracy quickly decreases in time. The fundamental
structure of the original system may be lost when this condition fails to be satisfied. On
the other hand, using numerical modeling techniques on nonlinear systems preserves the
system dynamics; however, are computational intensive and cannot be used for real-time
operation.

Recent developments in numerical tools and algorithms from data science are rev-
olutionizing the study of dynamical systems. An emerging method in nonlinear dynamical
systems research is the Koopman operator theory, which represents nonlinear dynamical
systems as a linear system in a lifted space [18, 19, 27, 29]. The Koopman operator pro-
vides us with the ability to analyze, predict, and control nonlinear systems while leveraging
linear stability analysis and machine learning techniques [7]. The Koopman operator theory
has been widely used to establish frameworks for approximating and analyzing a variety of
nonlinear systems [9, 34, 12, 35, 31]. Extensions of the Koopman based modeling approaches
for control have been proposed in [20, 1]. Now, Koopman based methods for learning and
controlling robotics systems have recently become a a topic of interest [39, 5, 2]. These
works demonstrate the benefits of Koopman based models of complex, nonlinear dynamical
systems.

To the best of our knowledge, this is the first documented use of Koopman based
modeling for a physical quadrupedal robot. Further applications of this research will aim
to use the techniques to build controllers.
1.1 Koopman Operator Theory

Consider a continuous-time dynamical system in the form

\[ \frac{d}{dt} x = f(x(t)) \]  

(1.1)

on a finite-dimensional state space \( \mathcal{X} \subseteq \mathbb{R}^n \). The state \( x(t) \) is propagated forward in time by \( \Delta t \) along the flow of the solution \( s_t(x) \) by the nonlinear vector-valued function \( f \). We define real valued-functions \( \psi : \mathcal{X} \to \mathbb{R} \), which are elements of an infinite-dimensional space of functions. These functions are known as observable functions of the system in eq. (1.1). The Koopman operator \( U_f \) is defined as an infinite-dimensional linear operator that acts on the observable functions as follows

\[ U_f \psi(x_t) = \psi(s_t(x_t)) \implies U_f \psi(x_t) = \psi(x_{t+\Delta t}) \]  

(1.2)

The continuous time Koopman operator, known as the Koopman generator, form a ordinary differential equation governing the evolution of the observable given by,

\[ \frac{d}{dt} \psi(x) = K \psi(x). \]  

(1.3)

The eigenfunction, \( \phi \) associated with the eigenvalue \( \lambda \) for the Koopman generator satisfies the following relationship:

\[ \frac{d}{dt} \phi(x) = \lambda \phi(x) \]  

(1.4)

For a comprehensive review of the Koopman Operator theory, we direct the reader to [28].

1.2 Data-driven Approximation of Koopman Operator

The Koopman operator is infinite dimensional, requiring an infinite number of basis functions to span and fully describe the nonlinear dynamics of the system. Therefore, algorithms are used to compute finite dimensional approximations of the Koopman operator.
1.2.1 Dynamic Mode Decomposition

Dynamic mode decomposition (DMD) is a least squares approximation of the Koopman operator which seeks to find the leading Koopman eigenvalues and corresponding eigenfunctions of the system [8]. Consider snapshots of time-series data from the continuous-time dynamical system (1.1). These snapshots are arranged into two data matrices \( X \in \mathbb{R}^{n \times M} \) and \( Y \in \mathbb{R}^{n \times M} \),

\[
X = \begin{bmatrix}
  x_1 & x_2 & \ldots & x_M \\
  \vdots & \vdots & \ddots & \vdots \\
  x_1 & x_2 & \ldots & x_M
\end{bmatrix}, \quad Y = \begin{bmatrix}
  y_1 & y_2 & \ldots & y_M \\
  \vdots & \vdots & \ddots & \vdots \\
  y_1 & y_2 & \ldots & y_M
\end{bmatrix}
\]  

where \( x_i \in \mathbb{R}^n \) and \( y_i \in \mathbb{R}^n \) are related using the flow of the solution \( y_i = s_{\Delta t}(x_i) \), where \( \Delta t \) is the fixed time step between the sampled system measurements. There are \( M \) snapshot pairs making up the dataset \( D = (x_i, y_i)_{i=1}^M \) on which you want to use to learn the Koopman operator using the DMD algorithm. The DMD solves the following minimization problem

\[
K_{dmd} = \min_K \| Y - KX \|_F = YX^T
\]  

where \( \| \cdot \|_F \) denotes the Frobenius norm. The explicit least squares solution is defined as

\[
K_{dmd} = YX^T
\]

where \( ^T \) denotes the pseudo-inverse and \( K \in \mathbb{R}^{n \times n} \) is the finite dimensional approximation of the Koopman operator. This formulation shows that DMD seeks to find the line of best fit between the states and the time shifted states. More explicitly, the DMD algorithm finds a linear coordinate transformation, in the same dimension \( \mathbb{R}^n \), where the original system dynamics appear linear. Because of this, DMD is able to identify a model which accurately captures the dynamics of systems which display linear, periodic, or quasi-periodic trajectories, but is unable to capture systems with fundamental nonlinear features. Despite
this weakness of the approximation, the appeal of utilizing the DMD algorithm is that it
does not require any knowledge of the governing equations, but instead based solely on the
measurement data [8]. This is an important distinction because many complex nonlinear
systems do not have complete information about the underlying dynamical equations, mak-
ing this a favorable algorithm compared to those that are dependent on the physical nature
of the system. This thesis focuses solely on DMD for the approximation of the Koopman
operator. Section 3.14 presents different methods of computing the DMD approximation
for the quadrupedal robot dynamics with control inputs.

1.2.2 Extended Dynamic Mode Decomposition

Extended dynamic mode decomposition (EDMD) extends the DMD algorithm by
introducing nonlinear coordinate transformations with the goal of capturing the nonlinear
dynamics of the system. Let $\Psi = [\psi_1, \ldots, \psi_N]^T$ be the choice of basis functions which
define the nonlinear transformation. For the continuous-time dynamical system (1.1), we
consider an lifted state vector,

$$\Psi(x) = \begin{bmatrix}
\psi_1(x) \\
\psi_2(x) \\
\vdots \\
\psi_N(x)
\end{bmatrix} \tag{1.8}$$

where the new vector of observables $\Psi(x) \in \mathbb{R}^N$ represents a lifted change of coordinates.
Consider snapshots of time-series data from the continuous-time dynamical system (1.1).
These snapshots are lifted to the observable states and arranged into two data matrices

$$\Psi(X) = \begin{bmatrix}
| & | & | \\
\Psi(x_1) & \Psi(x_2) & \ldots & \Psi(x_M)
| & | & |
\end{bmatrix}, \quad \Psi(Y) = \begin{bmatrix}
| & | & | \\
\Psi(y_1) & \Psi(y_2) & \ldots & \Psi(y_M)
| & | & |
\end{bmatrix} \tag{1.9}$$
where $\Psi(X) \in \mathbb{R}^{N \times M}$ and $\Psi(Y) \in \mathbb{R}^{N \times M}$. Note that DMD can be viewed as a particular case of EDMD where the basis functions are chosen as $D = \{e_1^T, e_2^T, ..., e_N^T\}$ each unit basis function $e_i \in \mathbb{R}^n$,

$$
e_1 = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad e_2 = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \ldots, \quad e_n = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}
$$

can be used to form a identity basis transformation $I(x)$. For EDMD, we can use these lifted data snapshot matrices to compute the following matrices,

$$G_1 = \frac{1}{M} \sum_{m=1}^{M} \Psi(x_m)\Psi(x_m)^\top$$
$$G_2 = \frac{1}{M} \sum_{m=1}^{M} \Psi(x_m)\Psi(y_m)^\top$$

where $G_1, G_2 \in \mathbb{R}^{N \times N}$. In a similar method to the DMD algorithm, EDMD finds the solution the following minimization problem,

$$\min_K \| G_2 - G_1K \|_F$$

where now we have a larger system $K \in \mathbb{R}^{N \times N}$. The explicit solution to the least square problem is

$$\mathcal{U}_{\Delta t} \approx K_{edmd} = G_1^\dagger G_2.$$  

The convergence of EDMD towards the true Koopman operator as the number of data points and basis functions go to infinity are provided in [21, 17]. The EDMD-based approximation of the Koopman operator can be used to approximate the Koopman generator as follows,

$$\mathcal{K}_t \approx \frac{K - I}{\Delta t} =: \mathcal{L}.$$  

(1.13)
where $I$ is the identity matrix. This solution, however, is for the observables in the lifted dimension $\mathbb{R}^N$, therefore the states of the original system are not being evaluated. In order to map back to $\mathbb{R}^n$ and obtain the original states, we can solve the following least-squares problem

$$\min_C \|X - C\Psi(X)\|_F$$

(1.14)

where $C \in \mathbb{R}^{n \times N}$ maps the observables back to the original state space $\mathbb{R}^n$. This matrix can be computed explicitly as

$$C = X\Psi(X)^\dagger$$

(1.15)

Finally, this dynamical system can be modeled as

$$\frac{d}{dt} \Psi(x) = L\Psi(x)$$

(1.16)

$$X = C\Psi(X)$$

(1.17)

Figure 1.1: The finite dimensional Koopman operator acts on observable functions of a nonlinear system and propagates them linearly forward in time in the lifted dimension $\mathbb{R}^N$. These observables are mapped back to the original state space $\mathbb{R}^n$ to recover $x$ at the next time step.
As stated previously, EDMD addresses the shortcomings of the DMD algorithm. EDMD allows us to find a nonlinear transformation of coordinates to a lifted state space where the nonlinear dynamics appear linear. This approximation is valid for a larger range of values and is able to capture the nonlinear components of the dynamics. However, for EDMD to be accurate we must assume that the choice of basis functions are rich enough to capture the true nonlinear dynamics of the system. There does not exist analytical approach for defining the optimal choice of basis functions, therefore this remains an open problem in the EDMD method of approximation. There do exist pragmatic choices for these functions which often perform well for a variety of nonlinear systems [37, 23].
Chapter 2

The Clemson Tigger

2.1 System Requirements

In order to utilize the theory of the Koopman Operator, we must determine a suitable dynamical system such that the use of this theory can be justified. The first step in identifying the most suitable system is to establish the requirements. Firstly, the robot must be sufficiently nonlinear. Although there already exists many affordable robots for purchase, the choices mainly include drones and tank-like robots, which have been extensively studied and modeled. These dynamical systems can be modeled fairly easily without restricting assumptions on the system’s motion. Affordability and ease of construction is another critical requirement for this robot. To ensure a feasible timeline for the construction of the robot, the hardware components must be standardized such that these pieces can be manufactured at reasonable prices. Finally, having a configurable robot is required, which would provide a lower degree model and can yield more testing for the operator theory. As a result, this would require customizable software.

Quadrupedal robot have been a central area of research for the past two decades. These systems are complex machines; however, they display unparalleled transversability that can be leveraged for scenarios with difficult locomotion. Due to their emergence in the research field and their complex structure, quadrupedal robots would be the most appro-
appropriate systems for this research. Therefore, only these legged robots would be considered for this thesis.

2.2 Comparisons of Legged Robots

A well-known quadrupedal robot that was made available for public purchase was Spot from Boston Dynamics. This legged-robot was marketed as a powerful machine that would be able to traverse varying obstacles by use of vision sensors and mapping localization capabilities. Along with the state-of-the-art hardware, the software development kit was available to users via GitHub, allowing for the creation of custom actions, sensor inputs, and applications. This state-of-the-art machine is available for purchase at the price of $74,500.

Unitree Robotics released its own quadruped called the Unitree A1, which was designed for research and development purposes. This robot boasts speeds up to 12km/h, long operating times, and high torque, which allows the robot to perform jumps and backflips. The system also comes with computing power suitable for localization and mapping. The price for the A1 starts at $10,000, with upgrades (i.e. sensors, upgraded CPU, etc.) available for additional costs. Comparing to other state-of-the-art robots on the market, this robot is comparable in capabilities and power, made more affordable to the users. As of 2022, this model has been discontinued, and Unitree has since released a new quadrupedal robot called Unitree Go1.

In 2019, a group of Stanford students published an open-source, highly agile quadrupedal robot called the Stanford Doggo [16]. This quadruped is designed with a quasi-direct drive mechanism that is capable of dynamic locomotion comparable to state-of-the-art legged robots. Software of the Stanford Doggo is available on GitHub, which provides access to both the microcontroller and motor controller code, allowing for extensive modifications to be made. The hardware for this robot is also available online and can be manufactured with hand tools, in total costing less than $3,000.
Although the development in robotics has greatly improved over the recent decades, state-of-the-art systems are not widely available to the public as they often rely on expensive and specialized hardware, hindering these systems from being mass-produced. Further development requires these systems to be manufactured more efficiently and more economically. For these reasons, and comparing the available options for a quadrupedal robot, the final decision was to construct a robot based on the open-source project, Stanford Doggo.

2.3 Quadruped Description

2.3.1 Physical Description

![Figure 2.1: The Clemson Tigger quadrupedal robot, based on the Stanford Doggo](image)

The exterior of the body frame and the legs are fabricated from aluminum. Overall, the robot is 20 cm wide and 13 cm in height with a mass of 6 kg. The legs are designed as a symmetric five-bar linkage with a length of 9 cm and 16 cm for the upper and lower links, respectively. The actuation of each leg is driven by coaxial drive assemblies, which involves two motors independently controlling the rotation of concentric leg shafts. Each
drive assembly is connected to a drive belt of ratio 3:1 that transfers the power from the motors to the independent coaxial drive shafts. Figure 2.2 shows this mechanism. This design of the quasdi-direct-drive allows an increase in effective torque while maintaining a low reflected torque, contributing to its vertical jumping agility [16].

Figure 2.2: Coaxial drive assembly. The different colors indicate the independently driven leg linkages and their corresponding motors [16]

2.3.2 Electrical Description

The electrical system of this robot consists of a Teensy 3.5 microcontroller, four Odrive motor controllers, and eight motors. To provide safety to the system, a relay is wired in series with the 24V battery and the power distribution board (PDB). The emergency stop button disengages the circuit in the relay, disconnecting the power to the robot. The PDB distributes power between the four motor controllers. Each motor controller powers and controls the two motors in the coaxial drive assembly for an individual leg. Figure 2.3 displays the electrical system of the Clemson Tigger. A separate 12V battery was added to exclusively power the Teensy to avoid a ground loop in the circuit. The microcontroller is connected serially to the ODrives, along with the inertial measurement unit (IMU) to communicate about information regarding motors and the position of the robot. The mi-
The Clemson Tigger is controlled by open-loop trajectories that enable the robot to walk, trot, bound, pront, and dance. The leg trajectories used on this robot are composed of two halves of sinusoidal curves, each half corresponding to either the stance or flight phase [16], as shown in Figure 2.4. These sinusoidal trajectories of the foot position are governed by the generalized coordinates: the desired leg angle $\theta$ and leg separation angle $\gamma$, where the leg angle is taken with respect to the vertical. There are other unique, predefined parameters including step frequency, stride length, stance height, etc., that describe a specific gait motion.

To enable the open-loop controller, the user inputs a command for the desired gait. This command sends the corresponding leg parameters and the proportional and derivative gains to the microcontroller. Using these values from the given command, the
microcontroller computes the sinusoidal trajectories and sends the leg positions to the motor controllers. The method that the microcontroller sends off and receives data from the motor controllers is through universal asynchronous receiver-transmitter (UART) protocol. UART protocol is a serial communication protocol which sends out data as packets of bits. Each bit corresponds to a particular piece of data. The microcontroller combines the gains and the desired (setpoint) values for $\theta$ and $\gamma$ into a binary packet message and then sends it serially to the motor controllers. The motor controllers then parse and store the data. A PD controller is used to convert the position input into a torque command that will execute the desired leg position. The PD control uses the gait gains to add virtual compliance to the system. The proportional gains add virtual stiffness and the derivative gains add virtual damping, creating a virtually compliant system [16].

In order to obtain the states of the system, the microcontroller requests data from the motor controllers. The values that will be useful for testing are the setpoint angles ($\theta_s, \gamma_s$), the measured angles ($\theta, \gamma$), currents of the motors ($i_0, i_1$), and the motor velocities ($v_0, v_1$). The Stanford Doggo has only allowed the setpoint and measured angles to be reported from the ODrive, so the UART protocol had to be rewritten to allow all of the aforementioned data to be retrieved. In a separate thread, the microcontroller is being given data from the IMU. This data is composed of linear acceleration and rotational velocity of the robot. After the microcontroller receives the states of the system, Robot Operating
System (ROS) messages were created to send the data, through the wireless module, to the off-board computer where they can be stored for data analysis. Figure 2.5 shows this control architecture of the Clemson Tigger and the frequency of the individual control threads.

![Control Architecture for the Clemson Tigger](image)

**Figure 2.5: Control Architecture for the Clemson Tigger**

### 2.3.4 Quadruped leg Kinematics

As discussed in the earlier sections, Clemson Tigger uses PD control to attain the desired leg angle ($\theta$) and leg separation angle ($\gamma$). The output of the controller is the torques for the desired setpoints. Inverse kinematics are derived to perform real time calculations of the motor torques needed to place the foot at a given point on its trajectory. The corresponding Cartesian coordinates for the $\gamma$ and $\theta$ positions are given by

\[ X = (L_1 \cos(\gamma) + h_l) \sin(\theta) \]  \hspace{1cm} (2.1)
\[ Y = (L_1 \cos(\gamma) + h_l) \cos(\theta) \]  \hspace{1cm} (2.2)

where $h_l = \sqrt{L_2^2 - L_1^2 \sin^2(\gamma)}$. Figure 2.6 shows the generalized coordinates in a Cartesian frame, used for inverse kinematics. Using this coordinate transformation, the leg
Figure 2.6: Parameters of the 5-bar mechanism for the leg assembly, whose generalized coordinates are $\gamma$ and $\theta$. These parameters are used to define the Jacobian for the inverse kinematics.

Jacobian $J_l$ for the quadruped robot can be computed as follows

$$\begin{bmatrix}
\dot{X} \\
\dot{Y}
\end{bmatrix} =
\begin{bmatrix}
\cos(\theta)(L_1 \cos(\gamma) + h_l) & \sin(\theta)\left(-\frac{L_2^2 \sin(\gamma) \cos(\gamma)}{h_l} - L_1 \sin(\gamma)\right) \\
-sin(\theta)(L_1 \cos(\gamma) + h_l) & \cos(\theta)\left(-\frac{L_2^2 \sin(\gamma) \cos(\gamma)}{h_l} - L_1 \sin(\gamma)\right)
\end{bmatrix}
\begin{bmatrix}
\dot{\theta} \\
\dot{\gamma}
\end{bmatrix} = J_l
\begin{bmatrix}
\dot{\theta} \\
\dot{\gamma}
\end{bmatrix}$$

(2.3)

The leg Jacobian can be used to convert the applied motor torques to the ground reaction forces on the foot using the leg Jacobian. This relationship is given by

$$\begin{bmatrix}
F_x \\
F_y
\end{bmatrix} =
\begin{bmatrix}
\tau_1 \\
\tau_2
\end{bmatrix} (J_l^T)^{-1}$$

(2.4)

Motor torques are not being directly measured on the quadruped, so the estimated torques are approximated by the following relationship

$$\tau_m = K_t I_m$$

(2.5)

where $K_t$ is the motor torque constant, and $I_m$ is the armature current of the motor. The kinematics for a quadruped with this leg linkage is detailed in [4].
2.3.5  Wheeled Configuration

As mentioned previously in this chapter, one of the desires for this robot was to reduce the degrees, thereby creating a reduced ordered system. To do this, a change in the firmware and software was created to facilitate this motion. The gait control was reduced, removing all parameters used to specify gait motion, and only taking in two parameters. The parameters of this robot were the leg angle with respect to the vertical ($\theta$), and the speed of the robot ($v$). In this design, a legged wheel assembly is attached to the concentric shafts of the QDD mechanism. The inner shaft is directly connected to the leg link and is control of the leg angle $\theta$. The outer shaft is connected to the wheel by a pulley system and controls the speed of the wheel. Figure 2.7 shows a diagram of the wheeled-legged configuration of the Clemson Tigger. After sending the commands for $\theta$ and $v$ from the computer, the microcontroller performs simple calculations to convert the leg angle to motor position and the velocity command to motor velocity (in turns/s). Since these are standard ODrive motor controller commands, these commands can be read from the motor controller without custom firmware.

![Figure 2.7: Generalized coordinates for the wheeled configuration of Clemson Tigger](image)

Although having a configurable mechanism was an initial requirement for this robot, the primary focus of this thesis was testing the legged gait motion as opposed to the wheeled-legged configuration. Therefore, testing on this configuration of the robot will be reserved for future works.
Chapter 3

System Modeling

This chapter introduces different dynamic models to represent the Clemson Tigger, which could be applicable to other quadrupedal robots. First we will detail the experimental methods used for system modeling and validation. Next, we introduce a physics based dynamic model of a quadruped using rigid body dynamics. After, we will introduce data-driven Koopman dynamic models which aim to capture the nonlinearities in the original system. Finally, we will display and discuss the results of the different methods.

3.1 Experimental Methods

For the dynamical models discussed in this section, we are concerned with creating models where the states of the system are the position of the Clemson Tigger robot. For the experiment, we ran gait commands for trotting and dancing and recorded data from each experiment, generating continuous recordings of trajectories with a total of 24,000 data snapshot pairs. As explained section 2.3.3, the state information from the motors that we sampled were $\theta, \gamma$ and current inputs. Gait motor data was sampled at 36Hz and the IMU data was sampled at 400Hz. For modeling discrete time dynamical systems, inputs and states need to be sampled at the same times. To correct this, the motor data was upsampled to the same frequency as the IMU data, at 400Hz.
3.1.1 Data Filtering

The data given out from the robot sensors is often rife with noise, and the data from these experiments proved to be no exception. Discontinuities are present in our measurement data due to jolts in sensors readings from the robot contact with its environment. To generate the most accurate results that represent true dynamics of the system, we must filter the measured data.

For the first method of filtering, raw measurement data acquired from the robot was filtered using a finite impulse response filter. Since our data is periodic, we choose to implement an anti-causal, zero-phase filter which eliminates phase distortion. We first designed a butterworth filter to eliminate any high frequency noise. Then, using the \textit{filtfilt} command in MATLAB, we obtained the filtered data snapshots with no phase distortion.

The second method of filtering utilized was robust principal analysis (RPCA). Using RPCA we can decompose a data matrix \( X \) such that,

\[
X = L + S \tag{3.1}
\]

where \( L \) is a structured low-rank matrix and \( S \) is a sparse matrix containing outliers and noisy data, shown in Figure 3.1. The principal components of \( L \) are robust to the noisy data present in \( S \) \cite{8}. The optimal decomposition of matrix \( X \) can be solved, with high probability, using the following convex problem

\[
\min_{L,S} \|L\|_* + \lambda \|S\|_1 \quad \text{subject to} \quad L + S = X \tag{3.2}
\]

where \( \| \cdot \|_* \) is the nuclear norm, which is defined as the sum of its singular values. For more information about RPCA for modal decomposition and applications, we direct readers to \cite{32}. Because of the upsampling process with the motor data, used specifically for the inputs of the system, this data did not respond well to the RPCA filtering. For this reason, the input data was only filtered using the previous method.
3.2 Rigid Body Dynamics Model

A variety of mathematical models have been used for the purpose of motion planning and control. Each model requires varying levels of approximations and assumptions to be considered a feasible description of the system. One method to model a quadrupedal robot is through rigid body dynamics. Rigid in this formulation implies that the individual bodies of the system do not deform when forces are applied, which is a reasonable assumption for the quadruped. With this dynamical model, the system can be fully defined through generalized coordinates $q = [q_b^T \ q_j^T]^T$, which is comprised of the body and joint states. The equations of motion for a system composed of rigid elements can be described using the Euler-Lagrange equations as follows

$$M(q)\ddot{q} + H(q, \dot{q}) = S^T\tau + J(q)^T,$$

(3.3)

where $M$ is the inertia matrix, $H$ captures the Coriolis, centrifugal, and gravity terms, $S$ is the actuation matrix that applies the torques $\tau$ to the joints and $J$ is the Jacobian that maps the forces $f$ to generalized forces. Furthermore, these equations can be separated into unactuated and actuated equations corresponding to the base and the joints, respectively [38]. Although these equations could be constructed for our quadruped robot, the structure of this formulation is complicated and cannot be used for any type of meaningful control algorithms, like Model Predictive Control.
In order to create a dynamical model that would enable meaningful formulation for modeling and control strategies, single rigid body dynamics are utilized for modeling the quadruped robot. This formulation removes the dependence on the joint angles while having approximately accurate dynamics. The assumptions for single rigid body dynamics require that the inertia of the legs is negligible to the inertia of the body. This could be a reasonable assumption for robots which have negligible leg masses compared to the base, or robots with significant inertia in the legs, but the joints move slowly and don’t move far from nominal positions [38]. For our quadruped, the mass of the legs are a small fraction (approximately 13%) of the base’s mass, and the gaits for testing did not allow for high levels of movement from the legs. Therefore, this is a valid assumption for using this model. The Newton-Euler Equations of the single rigid body are defined as

\[
\begin{align*}
\dot{\mathbf{p}} &= \frac{\sum_{i=1}^{n} f_i}{m} - \mathbf{g} \\
\frac{d}{dz} (I_b \dot{\mathbf{\Theta}}) &= \sum_{i=1}^{n} \mathbf{r}_i \times \mathbf{f}_i \\
\ddot{\mathbf{R}} &= [\mathbf{\Theta}] \times \mathbf{R}
\end{align*}
\]

where \( \mathbf{p} = [x \ y \ z]^T \) is the robot’s position, \( m \) is the mass of the body, and \( \mathbf{g} = [0 \ 0 \ 9.81]^T \)
is the acceleration of gravity in world coordinates, \( \mathbf{I}_b \in \mathbb{R}^{3 \times 3} \) is the robot’s inertia tensor in the body frame, \( \dot{\mathbf{\Theta}} = [\dot{\phi} \; \dot{\theta} \; \dot{\psi}] \) is the robot’s angular velocity and \( \mathbf{R} \in \mathbb{R}^{3 \times 3} \) is the rotation matrix which transforms a vector from the body frame to the world frame. For each foot force \( \mathbf{f}_i \in \mathbb{R}^3 \), the distance from the center of mass (COM) to the location of the foot force is given by \( r_i \in \mathbb{R}^3 \). The derivations for these equations are detailed in [38].

The orientation of the robot in the world coordinate frame is defined as \( \mathbf{\Theta} = [\phi \; \theta \; \psi]^T \). The angular velocity of the robot in the world coordinates can be found using the following relationship with the orientation

\[
\dot{\mathbf{\Theta}} = \begin{bmatrix}
\cos(\theta) \cos(\psi) & -\sin(\psi) & 0 \\
\cos(\theta) \sin(\psi) & \cos(\psi) & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
\dot{\phi} \\
\dot{\theta} \\
\dot{\psi}
\end{bmatrix}
\tag{3.7}
\]

If the robot not pointed vertically such that \( \theta = 90 \text{ deg} \), and for small values of roll and pitch \( \phi, \theta \), the above equation can be inverted to find

\[
\begin{bmatrix}
\dot{\phi} \\
\dot{\theta} \\
\dot{\psi}
\end{bmatrix}
\approx
\begin{bmatrix}
\cos(\psi) & \sin(\psi) & 0 \\
-\sin(\psi) & \cos(\psi) & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
\dot{\phi} \\
\dot{\theta} \\
\dot{\psi}
\end{bmatrix}
\approx
\mathbf{R}_z(\psi) \dot{\mathbf{\Theta}}
\tag{3.8}
\]

The approximated orientation relationship provided above, and the dynamical equations 3.4, 3.5, 3.6 can be combined to form the state-space representation as follows

\[
\frac{d}{dt}
\begin{bmatrix}
\mathbf{\Theta} \\
\mathbf{p} \\
\dot{\mathbf{p}}
\end{bmatrix}
=
\begin{bmatrix}
0_3 & 0_3 & \mathbf{R}_z(\psi) & 0_3 \\
0_3 & 0_3 & 0_3 & 1_3 \\
0_3 & 0_3 & 0_3 & 0_3
\end{bmatrix}
\begin{bmatrix}
\mathbf{\Theta} \\
\mathbf{p} \\
\dot{\mathbf{p}}
\end{bmatrix}
+
\begin{bmatrix}
0_3 & \cdots & 0_3 \\
0_3 & \cdots & 0_3 \\
I_3^{-1}[\mathbf{r}_1] \times \cdots \times I_3^{-1}[\mathbf{r}_4] \times
\end{bmatrix}
\begin{bmatrix}
\mathbf{\Theta} \\
\mathbf{p} \\
\dot{\mathbf{p}}
\end{bmatrix}
-
\begin{bmatrix}
\mathbf{f}_1 \\
\mathbf{f}_i \\
\mathbf{g}
\end{bmatrix}
\tag{3.9}
\]

where \( 1_3 \) is defined as the identity matrix. In the this model, parameter \( \mathbf{R}_z(\psi) \) is nonlinear and varies with time, therefore making this model a nonlinear time varying system. Further
details on constructing this dynamical formulation is provided in [11].

3.2.1 Results

Figure 3.3: System response of the SRBD with discrete inputs. The dynamical model fails to converge to the states of the experimental data with the same inputs.

Using the relationships defined by equations 2.1-2.5, we can use the leg coordinates (θ, γ) and the current inputs to the motors, we can calculate the distance from the foot to the COM, ri, and the corresponding foot force, fi, at a given time step. The only position data that is given from the IMU is the angular velocity ω and the linear acceleration ̇p. In order to obtain the angular position Θ and linear position and velocity (p, ̇p), the initial data was filtered using a low-pass filter and then integrated and derived accordingly. The initial condition of these states were given to the model 3.9, and, using an zero-order-hold on the inputs fi, simulated the SRBD model of the Clemson Tigger as a discrete time system. The response of this system is presented in Figure 3.3 and its eigenvalues in Figure 3.4.
The SRBD system is unable to stabilize its position other than the angular position along the X and Y axis, where it is marginally stable at best. This can be explained by the eigenvalues of the system. Although the system is time-varying, because the $A$ matrix is sparse, all eigenvalues are zero for each step in time. Therefore, if the input forces are not optimal, then the system will not be driven to stability.

### 3.3 Koopman DMDc Model

This section will present different formulations of the DMD algorithm, presented in section 1.2, to develop a data-driven model for the quadruped. However, the Koopman operator is defined only for autonomous dynamical systems and our quadruped is a controlled system. DMD with control (DMDc) is used for controlled systems and helps discover the underlying state dynamics without external control effecting the results [30]. For DMD to be applied to our system, we consider a dynamical system in following form

$$\dot{x} = f(x, u), \quad x \in X \subseteq \mathbb{R}^n, \quad u \in U \subseteq \mathbb{R}^m$$  \hspace{1cm} (3.10)
This augmented dynamical system now has a state vector comprised of the original states $x$ and the inputs $u$. The inputs have a direct impact on the dynamic evolution of the system, therefore, the DMD algorithm needs to be modified to allow control into the formulation. In addition to the snapshot matrices of the data $X = [x_1 \, x_2 \, \ldots \, x_m]$ and $Y = [y_1 \, y_2 \, \ldots \, y_m]$, the control input needs to be in the form

$$
U = \begin{bmatrix}
| & | & | \\
\mathbf{u}_1 & \mathbf{u}_2 & \ldots & \mathbf{u}_m \\
| & | & | 
\end{bmatrix}
$$

(3.11)

where $m$ is the number of snapshots of data and $u$ is the input vector. These matrices are then used to create the augmented state and control regression problem given by

$$
\begin{bmatrix}
\mathbf{Y} \\
\mathbf{U}'
\end{bmatrix} = \begin{bmatrix}
\mathbf{K} & \mathbf{B} \\
\mathbf{K}_{ux} & \mathbf{B}_{uu}
\end{bmatrix}\begin{bmatrix}
\mathbf{X} \\
\mathbf{U}
\end{bmatrix} \rightarrow \mathbf{Y} = \mathbf{KX} + \mathbf{BU}
$$

(3.12)

where $\mathbf{U}'$ is the time shifted input snapshots, and $\mathbf{K}$ and $\mathbf{B}$ are the Koopman operators for the state dynamics due to the states and input, respectively and $\mathbf{K}_{ux}$ and $\mathbf{B}_{uu}$ are the corresponding Koopman operators for the input dynamics. Because we are not concerned with the evolution of the input dynamics, we modify our Koopman matrix such that we only retain the operators for the state dynamics. However, this shows that DMDc allows us to quantify the effects of control inputs on the states of the system [30]. The least-squares solution to the DMDc problem is

$$
[K \quad B] = \mathbf{Y} \begin{bmatrix}
\mathbf{X} \\
\mathbf{U}
\end{bmatrix}^\dagger
$$

(3.13)

The linear operators $K$ and $B$ should satisfy the following linear, time invariant model of
the measured data dynamics

\[ x_{k+1} = Kx_k + Bu_k \]  \hspace{1cm} (3.14)

3.3.1 DMDc with SVD

The leading eigenvalues and eigenvectors of the operator \( K \) are obtained by dimensionality reduction and regression [30]. Firstly, the decomposition of the output matrix \( Y \) is defined as

\[ Y = \hat{U}\hat{\Sigma}\hat{V}^* \]  \hspace{1cm} (3.15)

where \([\cdot]^*\) denotes the conjugate transpose. The SVD of the input matrix is given by

\[
\begin{bmatrix}
X \\
U
\end{bmatrix} = \begin{bmatrix}
\hat{U}_1^* \\
\hat{U}_2^*
\end{bmatrix}\tilde{\Sigma}\hat{V}^* \longrightarrow \begin{bmatrix}
X \\
U
\end{bmatrix}^\dagger = \tilde{V}\hat{\Sigma}^{-1}\begin{bmatrix}
\hat{U}_1^* \\
\hat{U}_2^*
\end{bmatrix}^*
\]  \hspace{1cm} (3.16)

The matrix \( \hat{U} \) is separated into reduced bases for inputs space of both \( X \) and \( U \), given by \( \hat{U}_1^* \) and \( \hat{U}_2^* \), respectively. The SVD of the output matrix \( Y \) is given by

\[ Y = \hat{U}\hat{\Sigma}\hat{V}^* \]  \hspace{1cm} (3.17)

where \( \hat{U} \) defines the reduces basis for the output space [8]. Using this decomposition, we solve the least-squares regression problem in equation 3.13:

\[
[K \ B] = Y\tilde{V}\hat{\Sigma}^{-1}\begin{bmatrix}
\hat{U}_1^* \\
\hat{U}_2^*
\end{bmatrix}^*
\]  \hspace{1cm} (3.18)
these reduced order approximations are used to form the reduced order discrete time model of the system as

\[ \tilde{\mathbf{K}} = \hat{\mathbf{U}}^* \mathbf{Y} \hat{\mathbf{V}} \hat{\mathbf{S}}^{-1} \hat{\mathbf{U}}_1^* \hat{\mathbf{U}} \]
\[ \tilde{\mathbf{B}} = \hat{\mathbf{U}}^* \mathbf{Y} \hat{\mathbf{V}} \hat{\mathbf{S}}^{-1} \hat{\mathbf{U}}_2^* \]

(3.19)

(3.20)

\[ \tilde{x}_{k+1} = \tilde{\mathbf{K}} \tilde{x}_k + \tilde{\mathbf{B}} u_k \]
\[ \tilde{x}_k = \hat{\mathbf{U}} \tilde{x}_k \]

(3.21)

(3.22)

The singular value decomposition method of DMD is numerically stable compared to the pseudo inverse method. A thorough derivation for this formulation of DMDc is provided in [30].

3.3.2 Robust DMDc

If measurement noise is present in the data set, the DMD method does not account for this uncertainty and the approximation often does not perform well. A robust optimization approach to the approximation of the Koopman Operator works to accounts for the uncertainty in the data set. The robust DMDc algorithm seeks to find the solution of the following optimization problem:

\[
\min_{\mathbf{K}, \mathbf{B}} \| \mathbf{Y} - [\mathbf{K} \ \mathbf{B}] \begin{bmatrix} \mathbf{X} \\ \mathbf{U} \end{bmatrix} \|_F + \lambda \| [\mathbf{K} \ \mathbf{B}] \|_1
\]

(3.23)

where \( \| \cdot \|_1 \) is the 1 norm. The 1 norm penalty helps induce sparsity in the structure on the optimal solution of the linear operators \( (\mathbf{K}, \mathbf{B}) \) [33]. This regularization ultimately prevents overfitting on the training data. The model produced by RDMDc is given by the same form of the DMDc model, given by equation 3.14.
3.3.3 Hankel-DMD

Often times only partial observations of a dynamical system are available, therefore information about the hidden states is lost. It is possible to use delayed time measurements from the system to enrich the data measurements [6]. This can be done using the Hankel matrix representation of the data. The Hankel matrix is constructed from the delay embedding of time series measurements of a single trajectory, computed as the following

\[
H = \begin{bmatrix}
  x_1 & x_2 & \cdots & x_{M-d} \\
  x_2 & x_3 & \cdots & x_{M-d+1} \\
  \vdots & \vdots & \ddots & \vdots \\
  x_{d+1} & x_{d+2} & \cdots & x_M \\
\end{bmatrix}
\]

where \(d\) is the number of delay embeddings and \(h \in \mathbb{R}^{n(d+1)}\). The Hankel matrix representation depends on a single state trajectory of the system, therefore the snapshots data matrices are arranged such that

\[
H_1 = \begin{bmatrix}
  h_1 & h_2 & \cdots & h_{M-1} \\
\end{bmatrix}, \quad H_2 = \begin{bmatrix}
  h_2 & h_3 & \cdots & h_M \\
\end{bmatrix}
\]

It is important to note that the inputs to the system will not be lifted, so the matrix for the inputs is still in the form shown in eq. (3.11). The Hankel-DMD (HDMD) formulation of the least squares problem is therefore given by

\[
\begin{bmatrix} K & B \end{bmatrix} = H_2 \begin{bmatrix} H_1 \\ U \end{bmatrix}^\dagger
\]

where \(K \in \mathbb{R}^{n(d+1) \times n(d+1)}\) and \(B \in \mathbb{R}^{n(d+1) \times n}\). Due to the inclusion of the time delay coordinates, the system has been lifted to a \(\mathbb{R}^{n(d+1)}\) dimension. Although lifting to a higher
dimension is a key component of the EDMD algorithm (section 1.2.2), this is classified as a DMD approach because the coordinate transformations are linear. Regardless, a matrix $C$ to map from the lifted dimension to the original state space needs to be found. In order to recover the states $x$ from the observables $h$, we can conclude that $C$ must be

$$
C = \begin{bmatrix}
I_n & 0_n & \ldots & 0
\end{bmatrix}
$$

where $C \in \mathbb{R}^{n \times n(d+1)}$. The goal of using HDMD is to extract the Koopman invariant subspaces, which implies the existence of eigenvalues for the Koopman operator [3]. The information provided by the delayed embeddings is especially useful for systems in which trajectories frequently visit attractors in the state space [8].

3.3.4 Results

For the DMDc method of modeling the quadruped, the state observables were chosen to be the linear accelerations and angular velocities of the center of mass. To ensure that the Koopman model accurately captures the dynamics of the system and that system is controllable, input observables were chosen to best define the system. After testing multiple combinations of inputs, the input observables we found to provide the best results for the model are the foot force magnitudes $f_i$, the combined foot forces $F \in \mathbb{R}^3$, and the torques about the center of mass, $T \in \mathbb{R}^3$. This choice of input observables allows in enough information about the forces and geometry of the dynamical system. The model of system is therefore given by the following discrete-time model:

$$
\begin{bmatrix}
\dot{\Theta}(k+1) \\
\dot{p}(k+1)
\end{bmatrix} = K \begin{bmatrix}
\Theta(k) \\
p(k)
\end{bmatrix} + Bu \begin{bmatrix}
f_1(k) \\
\vdots \\
f_4(k) \\
F(k) \\
T(k)
\end{bmatrix}
$$

(3.28)
Figure 3.5: System response of the Koopman DMDc with SVD algorithm for angular velocity and linear acceleration

where $u \in \mathbb{R}^{10}$ and $B \in \mathbb{R}^{6 \times 10}$. Although the measurement data was filtered, the DMDc with SVD method failed to produce a model that would be controllable. An uncontrollable system would be unusable for future control formulations. Instead, this method utilized the unfiltered data and was able to satisfy the necessary conditions for our dynamical model. The DMDc model was simulated using the same initial conditions of the state measured data, and solved as a discrete-time system with time-steps of $\Delta t = 0.0025$ seconds with the chosen input measurements. Then, RPCA was applied to both the output data and the output of the learned model and the comparison of these responses are shown in Figure
3.5 and the eigenvalues of the data-driven model are shown in Figure 3.8. The eigenvalues of the discrete system are shown to be inside the unit circle, indicating that the model is stable. The root-mean-square error (RMSE) shows that learned model predicts the rigid body dynamics with some significant error over six seconds, approximately 2,400 iterations. Despite the inaccuracies in the predictions, the response still remains stable and the error does not increase significantly over the time horizon. These results are expected for DMDc, since no nonlinear lifting is performed and there still exists noise in the input measurements.

For the robust DMD method, the data was preprocessed using RPCA to filter the
noisy data. Unlike for DMDc with SVD, the RDMDc method was able to find a controllable model with accurate dynamics using this filtered data. This system was simulated in the same way as DMDc with SVD. The response of this system is shown in Figure 3.6 and its eigenvalues are shown in Figure 3.8. It can be shown that this learned model also accurately predicts the rigid body dynamics over six seconds while remaining stable. The RMSE tells us that this robust algorithm is much more accurate in forecasting the response of the true system than the previous Koopman model. The eigenvalues of the discrete system are shown to be inside the unit circle, indicating that the model is stable, and more stable than
those of the DMDc with SVD model. The improved results can be directly attributed to
the filtered data as well as the regularization term $\lambda$, which further accounts for the residual
noise in the state and input data. It is important to note that both RDMDc and DMDc
were able to predict the future trajectories of the system for approximately two seconds
before the next measurement arrives in 0.0025 seconds. This has great implications for the
use of these methods in real world applications where decisions must be performed within
milliseconds.

![Figure 3.8: Eigenvalues of the DMDc data driven models for quadruped dynamics. The
eigenvalues for each method are shown to be stable. The eigenvalues of the HDMDc (right)
are presented in the lifted space of $\mathbb{R}^{n(d+1)}$.](image1)

Similarly to the RDMD approach, the data was preprocessed using RPCA and we
were able to obtain a stable and controllable model using HDMDc. Various numbers of
delay embeddings were tested, and the value that seemed to result in the best performance
was $d = 400$. This is equivalent to storing information for one seconds of measurement data.
Results did not improve significantly when the number of embeddings increased, whereas
computation time was slower. After modeling the lifted states (observables) of the system,
we recovered the original state trajectories using the output matrix shown in eq. (3.27).
Figure 3.7 displays the state response, $x(t)$, of this linear system, and Figure 3.8 shows
the eigenvalues of the lifted dimensional system. Each eigenvalue for the lifted system is
contained inside the unit circle, proving the system to be stable. The error of the HDMD
approach provides the most accurate model from the DMDc methods. This is the expected
result due to the lifting of the states. The delay embeddings allow us to utilize information about the evolutions of the states. Higher frequencies, however, are still more difficult to track than the slower frequencies. This trend can be see for each method presented. Although RDMDc is shown to track these high frequencies with the most accuracy, the predictions becoming noticeably less accurate as time progresses.

As stated in section 1.2.1, these models will only be accurate in capturing the local linear dynamics. But due to the periodic gait trajectories of the quadrupedal robot, DMDc has shown that it is able to identify a linear system that accurately describes the nonlinear system. This, however, will prove to be a challenge for gaits and environmental interactions which do not appear periodic and add strongly nonlinear patterns to the response. The EDMD or DMD with deep learning methods will need to be used to model dynamics beyond the local linear dynamics of the system.
Chapter 4

Data-Driven System Identification for Terrain

4.1 Problem Background

Locomotion of quadrupedal robots has been actively researched due to the rising demand for autonomous machines. The main advantage of a quadrupedal robot, compared to wheeled, bipedal, and other configurations of robotic systems, is its capability to traverse challenging natural environments. This benefit drives the need for quadrupedal robots in industrial surveying, search-and-rescue missions, and other situations tasked with travelling over unknown or dangerous terrain.

Because of the inherent complexities of these dynamical systems, methods for controls rely on simplified models and assumptions to reduce computational complexity. In the recent years, new modeling techniques have emerged that allow for a more robust system, which involve formulating the quadruped dynamics as switched/hybrid systems [26]. This approach allows for stable gait planning using only proprioceptive sensors (no perceived information on the terrain). Further works model the robot as a switched system and design a whole-body MPC with online switching time optimization (STO) [14]; however, the limitation for this research is that it requires a pre-defined contact sequence and contact...
locations. Reinforcement learning techniques have been developed for quadrupedal locomotion, but focused mainly on testing in lab environments with rigid or lightly textured surfaces, not providing much robustness to deformable terrain [36], [24], [13].

These methods developed for the gait control of a quadruped are based on the assumption of rigid contacts. The interaction between the foot contacts and the terrain is essential for the control approach and planning of the robot. The use of exteroceptive sensors, such as LiDAR and cameras, is used for mapping and motion planning in previously unexplored environments, however, these sensors are only useful for determining the profile of the terrain, but are incapable of ascertaining the physical properties of the ground, such as friction coefficient, compliance, or stability of terrain. Therefore, these controllers will not be able to successfully navigate environments that have soft deformable, or slippery and unstable terrain. These issues could also be exacerbated by foliage coverage of the ground, obscuring the sensors and resulting in an inaccurate detection of terrain. Recent breakthroughs in quadrupedal locomotion have proposed robust control designs developed through reinforcement learning which relies only on proprioceptive signals, allowing quadrupedal robots to traverse challenging environments [25], [10]. However, these complex reinforcement learning strategies rely on expensive training data established from generating numerous simulation environments.

This chapter aims to provide a methodology for estimating terrain dynamics in real time using the spectrum of the Koopman operator. This method of terrain detection should not only lead to more accurate motion planning and gait control, but also is a more efficient approach since it does not rely on constructing expensive simulation environments and training expensive reinforcement learning policies.

### 4.2 Switched Systems Model

A switched dynamical system can be modeled as a set of indexed subsystems of the form $\dot{x} = f_j(x, u_j)$. Here, $j = \{1, 2\}$ is the subsystem index and $u_j$ is the corresponding
control input. A switching sequence is a list consisting of time instants $\tau_i$ and corresponding indices $j_i$ which determines a logical transition to the subsystem $j_i$ at the time instant $\tau_i$. A switching sequence can be modelled as $s = (\tau_0, j_0), (\tau_1, j_1), \ldots, (\tau_N, j_N)$, $0 \leq N < \infty$, $t_0 \leq \tau_0 \leq \tau_1 \leq \tau_N \leq t_f$. The states are assumed to be continuous at the switching instants, i.e., $x(\tau^+) = x(\tau^-)$.

In the case of quadruped locomotion, the continuous dynamics of the flight and stance phase can be defined as

$$\begin{cases}
\dot{x} = f_f(x, u_f) \\
\dot{x} = f_s(x, u_s)
\end{cases}$$

(4.1)

where $f_f$ and $f_s$ defines the flight and stance phase dynamics respectively. The time instant $\tau_i$ at which the height of the foot from the ground $h_z(t)$ crosses zero can be used to construct a switching sequence $s$ as follows

$$s(\tau_i, j_i) = \begin{cases}
(\tau, f) & |h_z(\tau^-) < 0, h_z(\tau^+) > 0 \\
(\tau, s) & |h_z(\tau^-) > 0, h_z(\tau^+) < 0
\end{cases}$$

(4.2)

where $f$ and $s$ are the subsystem indices of flight and stance phase respectively.

### 4.3 Experimental Methods

To demonstrate the effectiveness of the operator based modeling of quadruped dynamics, we performed experiments with the Clemson Tigger quadruped. As stated in section 2.3.3, gait commands are sent the Clemson Tigger through an off-board computer and are sent to the microcontroller. Low level control is achieved through a PD controller at 100 Hz to execute the desired gait trajectories. The joint states and control inputs were sampled at 36 Hz through real time ROS messages. Because position of the robot is not required for the leg-terrain dynamics, the IMU data is not sampled. This allows us to use the original motor data without the need to upsample. Gaits were tested first in a flight phase experiment.
where the robot did not come into contact with the ground, and then tested in a stance phase experiment where the robot stayed in contact with the ground.

Similarly to the procedure in Chapter 3, the measurements had significant noise present. Therefore, raw measurement data acquired from the robot was filtered using a finite impulse response filter. Since our data is periodic, we choose to implement an anti-causal, zero-phase filter which eliminates phase distortion. We designed a butterworth filter to eliminate any high frequency noise and obtained the filtered data snapshots with no phase distortion.

4.3.1 Flight Phase Experiment

To study the quadruped leg dynamics at flight phase, the robot was mounted on a test bench for the entire gait sequence so that there is no ground contact. We sampled open-loop gait trajectories from the following gaits: pronk, bound, walk, turn trot, and dance. The only input into the system for the flight phase is the torque output of the motors, also called the feed forward torque \( u_f \), based on the flight phase dynamics modeled as \( \dot{x} = f_f(x, u_f) \).

4.3.2 Stance Phase Experiment

For the stance phase experiments, the robot was set on different terrain such as rigid ground, sand, pebbles on sand, soil, and mulch on soil as shown in Figure 4.1. We then execute a static dance gait, where the robot actuates up and down with the feet in full contact with the ground. This will allow us to identify the stance phase dynamics \( \dot{x} = f_s(x, u_s, w) \). Tests were conducted using three different proportional gains for \( \gamma \), as shown in Table 4.1. The gains were chosen depending on the interaction between the robot and the specific terrain. Gains that were too high for certain terrains results in high amplitudes of response from the robot, causing the feet to lose contact with the ground. On the other hand, gains that were too small did not provide enough actuation for well defined data measurements. This results in a range of contact force responses acting on
the robot foot depending on the terrain properties. The input required for the robot in the stance phase $u_s$, is the sum of the feed forward torque and the feedback torque from the PD controller.

Table 4.1: Proportional gains $K_p$ for Stance Phase

<table>
<thead>
<tr>
<th>Proportional Gain</th>
<th>Rigid Ground</th>
<th>Pebbles on Sand</th>
<th>Mulch on Soil</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_p = 40$</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>$K_p = 30$</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>$K_p = 25$</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
</tr>
</tbody>
</table>

### 4.4 Data-Driven System Model

Consider the two phases of contact for a legged system, depicted in Figure 2.4. The continuous dynamics of this switched system are given in eq. (4.1), but can be rewritten as
the following

\[ \dot{x} = f(x) + g(x)u_f \]  
\[ \dot{x} = f(x) + g(x)(u_f + u_{pd}) + h(x)w \]

(4.3) 
(4.4)

where \(u_f\) is the feedforward current, and \(u_{pd}\) is the current generated by the PD controller. These leg dynamics are almost equivalent between the flight and stance phase, besides the different inputs and the introduction of a disturbance (or adversary) \(w \in \mathbb{R}^p\). In this formulation, the disturbance models how the terrain enters the system, and generally we attribute this term to the interaction of the terrain with the legs. Since our system is a continuous dynamical system, we will utilize the continuous time counterpart to the discrete Koopman operator, namely the Koopman generator defined in eq. (1.13). The goal is to model the switched system in the following form

\[ \dot{z} = L_f z \iff \dot{x} = f(x) + g(x)u_f \] 
\[ \dot{z} = L_s z \iff \dot{x} = f(x) + g(x)(u_f + u_{pd}) + h(x)w \]

(4.5) 
(4.6)

In order to identify the dynamics associated with flight (4.5), we use the DMDc algorithm presented in section 3.3. Although this formulation does not mention a system with disturbance, this term is treated like an input. Therefore, we let \(z\) be the augmented observable vector given by

\[ z = \begin{bmatrix} x \\ u \\ w \end{bmatrix} \]

(4.7)

In the case for the flight stance, the disturbance \(w\) can be dropped and the observable vector is that of DMDc. First we consider the flight dynamics where the generator equation
can be written as,

\[
\begin{bmatrix}
\dot{x} \\
\dot{u}
\end{bmatrix} =
\begin{bmatrix}
L_{2n \times 2n} & B_{2n \times m} \\
L_{\mathfrak{g}} & B_{\mathfrak{g}}
\end{bmatrix}
\begin{bmatrix}
x \\
u
\end{bmatrix}
\]

(4.8)

where \( L \) is the continuous-time, finite dimensional Koopman generator. As stated in the section 3.3, we are not concerned about the evolution of the inputs, only the dynamics of the states of this system. Therefore, we ignore the matrices corresponding to the input dynamics. Now consider the stance dynamics, where now we include the disturbance entering the system through \( H \).

\[
\begin{bmatrix}
\dot{x} \\
\dot{u}
\end{bmatrix} =
\begin{bmatrix}
L_{2n \times 2n} & B_{2n \times m} & H_{2n \times p} \\
L_{\mathfrak{g}} & B_{\mathfrak{g}} & H_{\mathfrak{g}}
\end{bmatrix}
\begin{bmatrix}
x \\
u \\
w
\end{bmatrix}
\]

(4.9)

The solutions to equations (4.8) and (4.9) can be obtained using \( M \) snapshots generated from the system at a fixed sampling frequency, so as to construct the snapshot matrices,

\[
X = \begin{bmatrix}
z_1 & z_2 & \cdots & z_{M-1}
\end{bmatrix}, \quad Y = \begin{bmatrix}
z_2 & z_3 & \cdots & z_M
\end{bmatrix}
\]

Using these snapshots of time series data, we can use the DMDc algorithm to find the Koopman operator \( K_j \) associated with each subsystem \( j \) (corresponding to flight and stance dynamics). We can then obtain the corresponding Koopman generator \( L \) using Equation (1.12). Furthermore, it is known that a bilinear controlled dynamical system exists in the space of functions; however, for the purposes of prediction we can simply assume that the control enters linearly through some \( B \) matrix found using least squares.

For our robot, the motion of the legs can be fully described through the generalized
coordinates $\theta$ and $\gamma$ and their derivatives (angular velocities). The current sent to the motors produces two opposing torques to the joint of the leg linkage. Therefore, we choose to define the states as $x = [\theta \, \gamma \, \dot{\theta} \, \dot{\gamma}]^T$, and the inputs as the motor currents given by $u = [I_0 \, I_1]$. The inputs for the flight phase are the feedforward currents, and the inputs for the stance phase include these and the feedback currents. During the stance phase, the normal force is modeled as a disturbance to the system, with $F_{tot}$ denoting the total estimated normal force as measured through the reflected motor torques. Altogether, these measurements will be used to identify the Koopman operators associated with the flight and stance dynamics.

4.4.1 Terrain Classification: Exploiting Linearity

The flight dynamics are a special case of the disturbance dynamics when $w = 0$ and we are left with only the drift and control vector fields $f$ and $g$ respectively.

$$\dot{x} = L_f x \approx f(x) + g(x)u_f + h(x)0 \quad (4.10)$$

When $w \neq 0$ the leg is in contact with the ground.

$$\dot{x} = L_s x \approx f(x) + g(x)(u_f + u_{pd}) + h(x)w \quad (4.11)$$

An important property of the Koopman operator is that it is linear, meaning the following is true

$$K(af(x) + bg(x)) = aK(f(x)) + bK(g(x)) \quad (4.12)$$

Therefore, it is easy to see that, through the exploitation of linearity of the Koopman generator, we could approximate the vector field $h(x)$.

$$L_s - L_f = L_s g(x)u_{pd} + Lh(x) \approx Lh(x) \quad (4.13)$$
It can be shown that we combine the contributions of both the disturbance and the feedback current into a term that we attribute to the entirety of the leg and terrain interaction dynamics.

### 4.5 Results

The results from these experiments with the Clemson Tigger quadruped show that the switched system model learned through a data-driven Koopman framework is able to predict gait trajectories on unknown terrains. The learned system model is validated on testing data, which was not used to build the system model. The results are presented in Figure 4.3. It can be seen that this data-driven model can accurately predict the future leg dynamics over a finite time horizon. The data-driven Koopman model is stable and predicts the state trajectories with high accuracy for approximately five time steps. The error between the measurement data and the predictions are seemingly minimal, however, it can be seen that the error begins to grow over the time horizon. These trajectories can be further visualized through the phase portrait shown in Figure 4.4. Using the flat ground as a baseline comparison, it can be inferred that the mulch and pebble causes perturbations in $\theta$ due to terrain-leg interaction dynamics. However, mulch has similar separation angle.
γ to the flat ground while pebbles lead to more abrupt terrain disturbances, causing higher variance in θ.

Figure 4.3: Occupancy maps of the state space on varying terrains[22].

Furthermore, different terrain properties show up as different eigenvalues in the spectrum of the Koopman generator shown in Figure 4.3. Once again, we use the eigenvalues of the flat ground as a baseline model for comparison. Soft terrains, such as mulch, shifts the eigenvalues more towards the real axis, resulting in a greater dampening in the

Figure 4.4: Eigenvalues of the generator \( L_s \) associated with the each terrain type. (using a \( \gamma \) gain of \( K_p = 30 \)).
legs response. Conversely, stiffer terrain such as pebbles on sand has less of a dampening effect. These distinctions are seen more clearly through the spectrum of the generator (see Figure 4.4). Ultimately, by analyzing the state-space trajectories and the spectrum of the Koopman generators, we can show distinctions between the varying terrains on which our robot traversed. Developing a motion controller based of this data-driven formulation is beyond the scope of this thesis, however, future works will research the application of these terrain identification framework using koopman. Also, it is important to note that, although the experimental setup requires two set of test to distinguish the flight and stance phases, this could also be accomplished through a single training phase through segmentation of contact states gathered from walking gait data.
Chapter 5

Conclusions and Discussion

In conclusion, accurate modeling of complex dynamical systems still proves to be an essential challenge due to the existence of nonlinearities and the lack of precise or sufficient equations for these systems. Even with reasonably accurate physics based models for legged robots, these models are complex and do not allow for meaningful control designs. We show that dynamic models, formed from linearization and more strict assumptions, do not easily stabilize and fails to capture the dynamics of the true nonlinear system. On the other hand, using the Koopman Operator we can find stable linear data-driven models which accurately forecast the nonlinear response of the system over a finite time horizon. Moreover, this method will enable real time predictions for real-life dynamical system applications due to the computational efficiency of the algorithms. Different methods for computing these data-driven models for a controlled system are presented. Furthermore, we show that linearity properties of the Koopman operator can be exploited to uncover underlying dynamics from external inputs or disturbances.

The modeling of dynamics and deformable terrain locomotion are both problems prevalent in quadruped research, and this thesis aims to address the problems using data-driven techniques. The methods presented can be further expanded upon by using EDMD approximation of the Koopman operator to better capture the complex nonlinear dynamics of the true system. Future works will aim to utilize these derived system models to design
a controller for a quadrupedal robot.

## 5.1 Future Research

![Terrabot Diagram]

Figure 5.1: Terrabot: A new quadruped for data-driven identification

Due to the current limitations with the Clemson Tigger, we have been working towards building a new quadrupedal robot, Terrabot. This quadruped will serve as the next generation of data-driven learning on a quadruped for our lab. Like its predecessor, Clemson Tigger, Terrabot is based on another open-source quadrupedal robot known as the Stanford Pupper v2 [15]; however, more significant changes will be implemented into this robot. Firstly, this robot is designed to be approximately twice the size of the Stanford Pupper, with a length and width of 470mm (18.5in) and 180mm (7in), respectively, and a nominal height of 280mm (11in). The motors are chosen to produce more than twice the amount of torque than the original motors to compensate for the additional load. The body of this robot, including the legs, are fabricated from 3D printing with nylon. This has allowed us to make custom and intricate pieces so that the physical construction of the body is simple, efficient, and will be conducive for testing. The most significant change is this quadruped will be housing an onboard computer and a stereo camera. This will now provide us with the capability of handling localization and mapping algorithms, and, most importantly, future data-driven algorithms and control strategies.
Bibliography


