Application of Koopman Operator Theory to Legged Locomotion

by

Jasmine G. Terrones

B.S.

California Institute of Technology (2022)

Submitted to the Department of Mechanical in partial
fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

SEPTEMBER 2024

Authored by: Jasmine G. Terrones

Department of Mechanical Engineering

August 12, 2024

Certified by: Harry H. Asada

Ford Professor, Department of Mechanical Engineering

Thesis Supervisor

Accepted by: Nicolas Hadjiconstantinou

Chairman, Department Committee of Graduate Theses

Application of Koopman Operator Theory to Legged Locomotion

by

Jasmine G. Terrones

Submitted to the Department of Mechanical on
August 12, 2024, in partial fulfillment of the
requirements for the degree of
Master of Science in Mechanical Engineering

Abstract

Nonlinearities from complicated robot systems and harsh contact dynamics have long impeded the effectiveness of optimal control strategies for legged robots. In this work, we present a linearized simple walking model using Koopman Operator Theory, and its usage in Linear Model Predictive Control (L-MPC). Various walking and contact models were evaluated, but ultimately the rimless wheel was selected due to its inherent stability and low dimensionality, and a nonlinear viscoelastic model was used to accurately capture floor contact and impact dynamics. Koopman models were developed using both Radial Basis Functions (RBFs) and neural network-generated observables for the passive rimless wheel. A novel actuation method with linear actuators, combined with the Control Coherent Koopman methodology, resulted in accurate linear models that effectively enabled L-MPC to control the wheel on flat ground. This model outperformed those created using the more traditional Dynamic Mode Decomposition with Control method.

This work demonstrates the power of Koopman linearization to produce a unified set of linear dynamical equations that encompass various contact and non-contact configurations and demonstrates the effectiveness of the Control Coherent Koopman methodology in generating an accurate input matrix across these different contact modes.

Thesis Supervisor: Haruhiko Harry Asada
Title: Ford Professor of Engineering, Department of Mechanical Engineering

Acknowledgments

I want to express my gratitude to Professor Harry Asada for being an amazing guide throughout this project and Master's thesis journey. His guidance has made a big difference in shaping this work. To my labmates, family, and friends, your support has been invaluable. A special thanks to Diego for listening to my Koopman ramblings.

This material is based upon work supported by the National Science Foundation under Grant No. NSF-CMMI 2021625. I appreciate the support by NSF.

Table of Contents

Table of Figures

List of Tables

Chapter 1

Introduction

1.1 Legged Locomotion

The field of legged locomotion has experienced significant growth over the last few years, largely due to advancements in optimization-based control strategies. In a world designed for humans, the ability to robustly navigate unstructured terrain enables the application of legged robots to factories, homes, urban environments, and even natural settings. Recent improvements to online optimization strategies have enabled the synthesis and execution of complex maneuvers, such as barrel rolls on the MIT Mini-Cheetah and energy-efficient bipedal walking on Cassie [1], [2] . From advances in reinforcement learning to improved Model Predictive Control (MPC) algorithms, real-time optimal control may soon be within reach.

Most of the legged locomotion community has converged on generating motions based on the formulation of an Optimal Control Policy (OCP). An OCP generates trajectories of state, control, and contact forces while minimizing a user-defined cost function. The trajectories must satisfy several constraints: the states must evolve according to the nonlinear whole-body robot dynamics, joint angles must stay within an allowable range, and additional constraints can be encoded to ensure stability and robustness. However, increases in the number or complexity of constraints, the dimensionality of the OCP, and the nonlinearities and stiffness of the dynamics can greatly increase the computational load of solving the OCP [3]. Furthermore, nonconvex dynamics lead to optimizations that are sensitive to initial conditions and locally optimal

solutions [3]. Nevertheless, the OCP framework enables the planning and execution of impressive trajectories for legged robots.

The general idea of Model Predictive Control (MPC) is to run the OCP at every iteration of the control loop and apply only the control from the first part of the trajectory. However, due to computational challenges, there are delays between sensor measurements, OCP calculations, and control implementation [4]. Templates are low-dimensional, simplified walking models that capture the essential elements of walking while eliminating unnecessary complexity, greatly increasing the speed of MPC. Common template models, such as the Linear Inverted Pendulum Model, the Spring-Loaded Inverted Pendulum Model, the Single Rigid-Body Model, and the centroidal model, can be readily solved online [3]. While template MPC is fast due to the low-order dynamics and, in the case of the linear inverted pendulum, convexity, it must rely on a low-level controller to translate the optimized trajectory into whole-body commands, typically via quadratic programming, inverse kinematics, or inverse dynamics [3], [4], [5]. Furthermore, these optimized trajectories are only as accurate as the template model; simplifications in the low-level dynamics, contact, and constraints yield infeasible or inefficient motions [3]. This is represented in Figure 1, where most template models yield feasible motions that do not perfectly overlap with those generated using the full-body dynamics.

In contrast, Whole-Body MPC always produces feasible trajectories but faces significant computational challenges when optimizing with high-dimensional, nonlinear, and non-convex dynamics [3]. Whole-body MPC is often used to generate optimal trajectories offline to create a motion library, where the motion is then executed via closed-loop feedback control. However, following offline trajectories requires the robot to start sufficiently close to the trajectory's initial

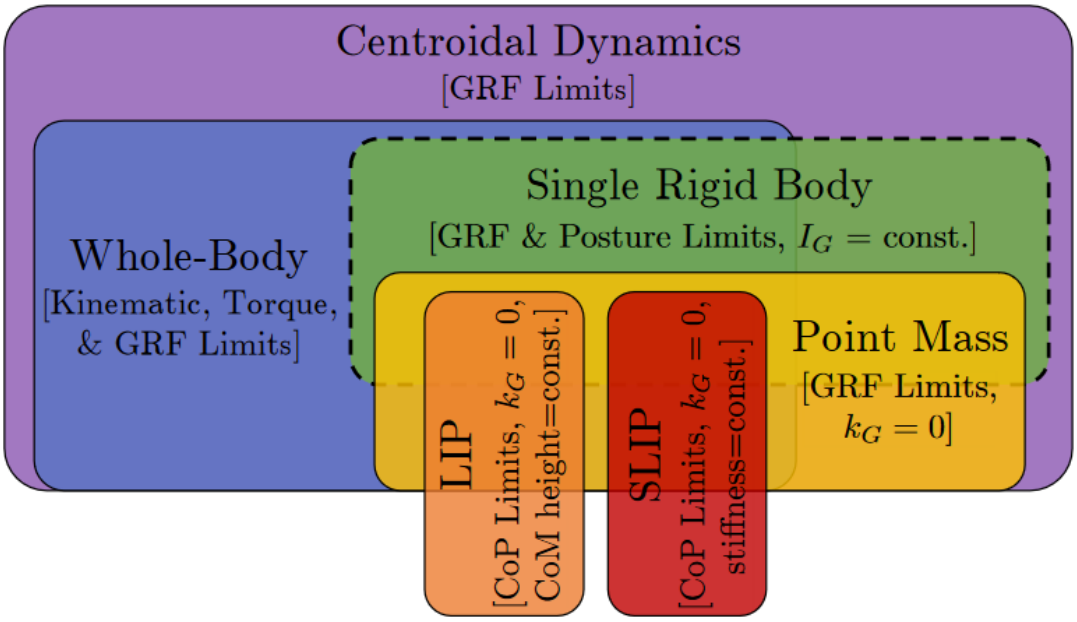


Figure 1. Feasible motions according to different models. The whole-body model is fully contained within the centroidal dynamics model, meaning any feasible motion for the whole-body model is also feasible for the centroidal model. However, due to a lack of torque and kinematic constraints, the centroidal dynamics produce unphysical, unachievable motions. The point mass model enforces ground reaction forces (GRF) limits as well as restricts angular momentum, making its feasible motions a subset of those produced by the centroidal dynamics model. From Figure 5 of [3].

condition and a carefully tuned tracking controller, not to mention the limitation of operating within a predefined set of motions [3].

A difficult aspect of legged locomotion is repeated high-impact ground contact, which introduce segmented dynamics, poor conditions for gradient-based optimization, and inhibit the study of global behaviors. These contacts are typically modeled as rigid or viscoelastic. Viscoelastic contact models allow contacting bodies to intersect and generate spring-damper-like repulsive forces based on material properties and the position, velocity, and penetration depth of the contacting bodies. With the proper framework, a viscoelastic contact model can yield continuous, differentiable dynamics [3].

Rigid contact models (or infinitely stiff viscoelastic models) yield hybrid dynamics, or dynamics that alternate between continuous and discrete regimes. When two bodies make contact, impulsive forces are immediately applied, resulting in an instantaneous jump in velocity. Hybrid dynamical OCPs can be represented as a Linear Complementarity Program or a Mixed Integer Program, but both methods are non-differentiable and less efficient than smooth optimization strategies. Fixing a contact mode sequence beforehand yields time-varying system dynamics, enabling the use of smooth optimization strategies, but limiting the ability to find new contact modes [3].

Overall, optimal control problems for complicated robot systems undergoing contact have slow computation speeds due to the nonlinear dynamics, high dimensionality, and inability of gradient-based optimization to handle contact. The use of templates is viable for online use but suffers from infeasible or suboptimal solutions.

1.2 Lifting Linearization

Lifting linearization, underpinned by Koopman Operator Theory, transforms nonlinear dynamical systems into higher-dimensional linear systems, enabling the application of linear control techniques to complex, nonlinear problems. Koopman Operator Theory states that a nonlinear autonomous system can be represented, without approximation, as a linear dynamical system in an infinite dimensional state. Consider a system with discrete-time, nonlinear dynamics $x_{t+1} = F(x_t)$. By augmenting the state space with additional observable functions $\boldsymbol{g}(x_t) = [g_1(x_t), g_2(x_t), \dots]$ that lie in a Hilbert space, a linear model may be constructed that evolves linearly with the Koopman operator, K.

$$K\boldsymbol{g} \triangleq \boldsymbol{g} \circ F \rightarrow K\boldsymbol{g}(x_k) = \boldsymbol{g}(x_{k+1})$$

The state evolution matrix A can either be obtained through data-driven methods (e.g., Dynamic Mode Decomposition) or by leveraging the underlying dynamics, as seen in Koopman Direct Encoding [6], [7]. The observables oftentimes consist of time series data and function families (RBF, Fourier series, etc.) or can be generated using neural networks. This approach offers significant advantages by simplifying analysis and control design, but it also presents challenges in terms of model construction and the choice of observables.

1.3 Prior Works

Traditionally, the Koopman Operator only applied to autonomous systems without control. However, the Koopman community has developed numerous methods to incorporate

control into the linear models. One popular method to extend Koopman Operator Theory to control is Dynamic Mode Decomposition with Control (DMDc), which approximates the control input term as a linear term with constant coefficients via least squares estimation. While DMDc has been useful, it assumes a constant B matrix, which rarely reflects reality [8]. Nevertheless, DMDc MPC has been used on Koopman models ranging from cable suspension systems to soft robotics [9], [10]. Alternatively, bilinear control matrices more accurately capture control that is affected by both state variables and control variables, but a bilinear formulation is more complex and restrictive in terms of available control strategies [11].

The proposed method by Asada et al. in 2024 presents a solution by constructing a Koopman operator for a class of control systems without approximating the input matrix B. The Control-Coherent Koopman (CCK) methodology ensures that the control matrix retains its correct structure by leveraging the causality of physical system modeling applied to actuator dynamics. This approach bridges the theoretical and technical gap between Koopman Operator Theory and practical control engineering needs [11].

Prior work has had success in applying Koopman Operator theory to hybrid systems, but despite promising results, there is not yet rigorous theoretical support for the applicability of the theory to systems with discontinuous jumps in state. However, with the use of a viscoelastic contact model, a Koopman contact model may be constructed [12].

Finally, various methods have been developed to create controllers with control objectives or augmented dynamics that are impact invariant. These enhancements provide a more robust control strategy that remain effective despite the nonlinearities associated with impact events [13], [14].

1.4 Project Overview

Koopman Operator Theory emerges as a potential methodology to solve many of the problems faced by the legged locomotion field. An accurate linear Koopman model would have the same fast computation as linear template MPC without the shortcomings of inefficient or infeasible trajectories. Furthermore, a globally linear unified representation of walking dynamics, one that subsumes the contact/non-contact dynamics, would enable linear model predictive control with the ability to plan through contact.

The goal of this work is to apply Koopman Operator theory to enable Linear MPC for complex hybrid dynamics found in legged locomotion, potentially leading to online trajectory optimization in the future. The work is organized as follows…

Chapter 2 explores the key challenges and design decisions related to the project.

Chapter 3 covers the Rimless Wheel modeling and MATLAB simulation

Chapter 4 discusses the creation and performance of the passive Koopman models.

Chapter 5 focuses on the actuation method and the creation and performance of the actuated Control Coherent Koopman model.

Chapter 6 focuses on MPC using both Control Coherent Koopman models and a comparison Dynamic Mode Decomposition with Control Koopman model.

Lastly, Chapter 7 concludes the work and discusses future directions for the project.

Chapter 2

Approach

2.1 Key Problems

In the application of Koopman Operatory Theory to legged locomotion, there are a number of design decisions to be made regarding model type, contact model, and actuation. The first attempt utilized the simple compass walker due to its simplicity and ability to exhibit passive dynamic walking. Passive walkers do not require motors to move; they are powered by inertia and gravity. Thus, they are a good starting point for creating a linear Koopman model for both autonomous and dynamic walking. Given the novelty of the application, it was important to select as simple a walking model as possible. Furthermore, a current limitation of common Koopman Operator methodologies is the curse of dimensionality, so a low-dimensional model was especially desirable. However, through this first attempt, three important hurdles were identified for the project:

1. Discrete Dynamics from Impacts

   Capturing energy loss from collisions is especially important for passive walkers, where energy is finite. However, hybrid dynamics and Koopman Operator Theory are incompatible. Koopman Operator Theory assumes that the lifted system's dynamics, and

by extension the composition of the observables and the state transition must lie in the Hilbert space.

$$(g \circ F)(x) = g(F(x)) \in H.$$

This assumption is not guaranteed for hybrid systems with discontinuous states [12].

2. Unstable System Identification

There are very few configurations where a passive walker will walk down a ramp indefinitely. A limit cycle is a closed trajectory that represents a periodic solution. Points near a stable limit cycle that converge onto the limit cycle are said to be within the envelope of convergence. With a complex, high DOF walking model, the envelope of convergence is oftentimes found via numerical simulation, and the limit cycle is found with optimization methods. For our Koopman model to be useful for MPC, it must be accurate on and around the limit cycle while still capturing unstable modes. However, unstable modes tend to dominate, resulting in inaccurate predictions. In fact, the majority of the work for unstable modes in Koopman models involves suppressing or projecting them onto the stable unit circle [15], [16]. Previous attempts were made to generate individual orthogonal Koopman models for each subspace associated with the simple compass walker. In this way, a point on the limit cycle would only stimulate the marginally stable subspace, thereby preventing the unstable modes from dominating. However, it was unclear how to enforce orthogonality between the subspace-specific Koopman models.

3. Koopman Operators and Control

Previously, it was thought that Koopman Operator Theory was only applicable to passive systems. However, a Koopman model of walking dynamics used only for characterization/analysis is redundant due to the added effort needed to capture the stable, unstable, and marginally stable modes of the system. An initial idea was to use impulses to actuate the walker, which would be applied instantaneously, without the need for a control B matrix. With the Control-Coherent Koopman methodology, utilizing the actuator dynamics can enable linear actuation by introducing a delay between the actuation and the main system [11]. However, not all systems are suitable for the Control-Coherent actuation formulation.

2.2 Simple Walking Model Summary

A number of common simple walking models and templates were evaluated for use in the project. Based on the problems identified above, walking models with no reset map or some method by which to absorb the impact from foot strike were of special interest.

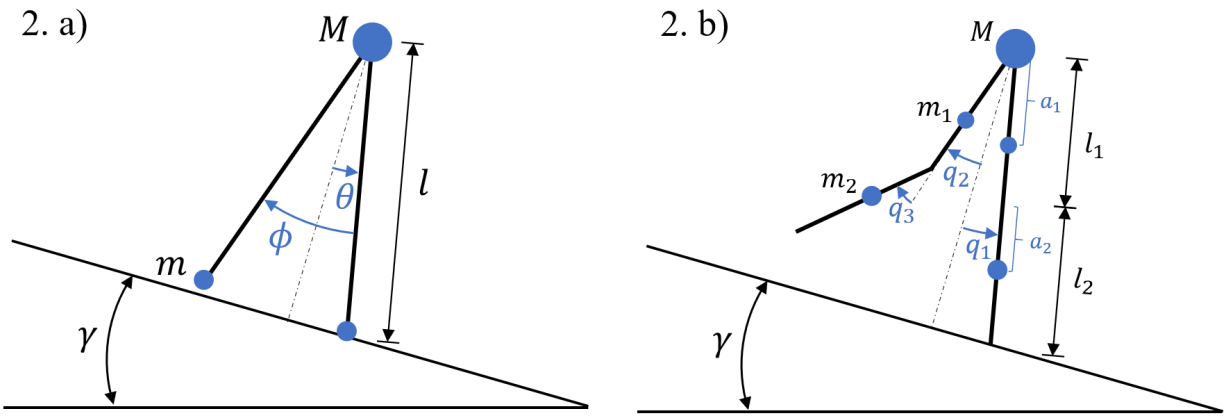


Figure 2. a) Rigid Compass Walker and 2. b) Kneed Walker Models as defined in [17] and [18].

The rigid compass walker was one of the first examples of passive dynamic walking, where stable walking is achieved with no actuation or control. It consists of two legs, a hip mass, and two leg masses [19]. It is commonly assumed that the hip mass is much larger than the leg masses to further simplify dynamics. The rigid compass walker assumes that the stance foot is attached to the ground via a pin joint. When the swing leg hits the ground, the collision is inelastic and instantaneous, and the stance and swing legs are also switched. Thus, the reset map encodes both the collision and coordinate change. Furthermore, the compass walker model ignores foot scuffing, which is when the swing leg briefly makes contact with the ground. The

continuous dynamics are derived using Euler-Lagrange equations, and the discrete dynamics are obtained via the conservation of angular momentum.

Rigid Compass Walker [17]

Parameters

Ramp angle	$0 \leq \gamma \leq \pi/2$
Leg length	$l > 0$
Hip mass	M
Leg masses	m

States $\mathbf{x} = [\theta, \phi, \dot{\theta}, \dot{\phi}]$

θ: angle of the stance leg relative to the slope normal

ϕ: angle between the stance and swing legs

Continuous Dynamics (m << M)

$$\ddot{\theta}(t) - \sin(\theta(t) - \gamma) = 0$$

$$\ddot{\theta}(t) - \ddot{\phi}(t) + \dot{\theta}(t)^2 \sin(\phi(t)) - \cos(\theta(t) - \gamma) \sin(\phi(t)) = 0$$

Guard Function

$$\phi(t) - 2\theta(t) = 0$$

Discrete Dynamics

$$\theta(t^+) = -\theta(t^-)$$

$$\phi(t^+) = -2\theta(t^-)$$

$$\dot{\theta}(t^+) = \cos(2\theta) * \dot{\theta}(t^-)$$

$$\dot{\phi}(t^+) = \cos(2\theta)\,(1 - \cos(2\theta)) * \dot{\theta}(t^-)$$

Key Assumptions

Collisions with ground are inelastic/impulsive

Stance foot is attached to the ground via pin-joint

Weight transfer between legs occurs instantaneously

Foot scuffing is ignored

Similar to the rigid compass walker, the kneed walker is a more complicated passive walking model with the ability to clear the floor. Given the correct initial conditions, the swing leg will bend at the hinge during the forward swing and straighten out before touching the ground. The kneed walker consists of two legs and a hip mass, only now the legs consist of two links attached with a hinge. Each link has an associated mass, yielding a total of five masses. When the links are aligned, knee strike occurs and the knee is locked. This is modeled as a discrete inelastic collision. Thus, the kneed walker circumvents the foot scuffing issue but has more complex dynamics and two instances of discrete dynamics per walking cycle [19]. The full equations of motion for this model can be found in [18].

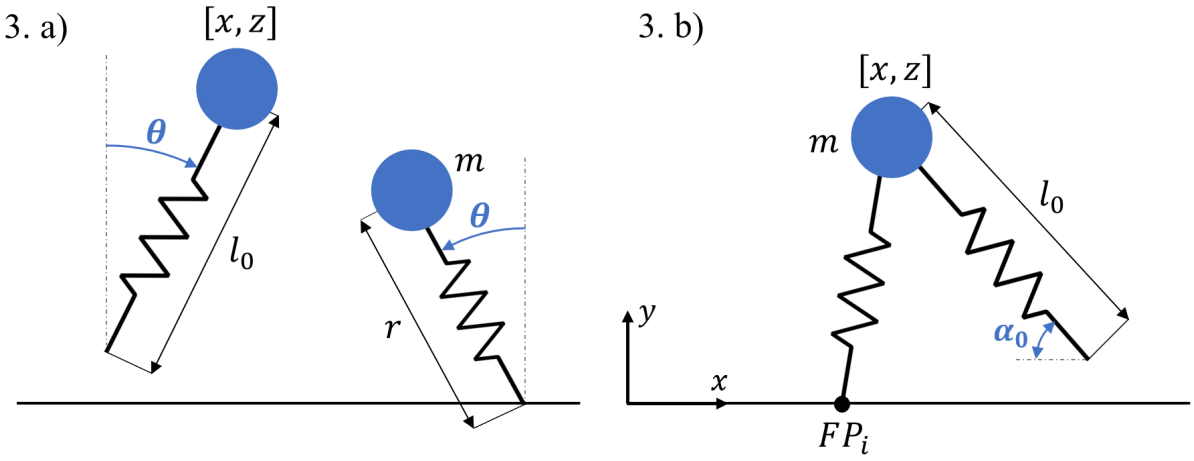


Figure 3. a) The Spring-Loaded Inverted Pendulum (SLIP) model in both flight and stance [19]. 3. b) The bipedal spring mass model as defined in [20].

The spring-loaded inverted pendulum (SLIP) model is a commonly used template for running. The SLIP model consists of a point mass and a massless, spring leg. Unlike the previous two models, the SLIP model uses different dynamics to govern the flight phase and the stance

phase. Furthermore, the coordinates used for each phase are different, yielding a piecewise holonomic dynamical system. Finally, the massless leg allows for the instantaneous repositioning of the leg during flight such that the control input u directly defines leg angle θ. During stance, the leg end point is fixed to the ground via a pin joint. Despite its simplicity, the SLIP model can accurately describe experimental data from humans, to cockroaches and crabs [21].

The bipedal spring-mass model is an extension of the SLIP model that includes two legs. In this model, there are two massless spring-like legs and a central mass. The bipedal spring-mass model is capable of different periodic walking patterns including both walking and running. Unlike the SLIP model, the bipedal spring-mass model has a single set of states, $\mathbf{x} = [x, z, \dot{x}, \dot{z}]$. Furthermore, the equation of motion is given simply by $m\ddot{\boldsymbol{r}} = \boldsymbol{F_1} + \boldsymbol{F_2} - m\boldsymbol{g}$, where forces $\boldsymbol{F_1}$ and $\boldsymbol{F_2}$ are the forces of legs 1 and 2 respectively when they are in stance. These forces become zero when the leg is not in contact with the ground. The force is a function of the leg tip and center of mass (COM),

$$\boldsymbol{F_1} = k\left(\frac{L_0}{|\boldsymbol{r} - \boldsymbol{r_{FP1}}|} - 1\right)(\boldsymbol{r} - \boldsymbol{r_{FP1}})$$

Note that this force becomes zero when the spring becomes fully uncompressed. The transition between swing to stance occurs when a leg fulfills the landing condition $z_{TD} = L_0 \sin(\alpha_0)$ and the vertical velocity is negative [20]. The model assumes a constant angle of attack, α_0. Thus, there are no swing dynamics and the equations of motion require knowledge of the foot point position $\boldsymbol{r_{FP1}}$ and $\boldsymbol{r_{FP2}}$.

Spring Loaded Inverted Pendulum [19]

Parameters

Leg rest length	l_0
Hip mass	m
Spring constant	k

States $\boldsymbol{x_{flight}} = [x, z, \dot{x}, \dot{z}]$, $\boldsymbol{x_{stance}} = [r, \theta, \dot{r}, \dot{\theta}]$

x: horizontal position of the hip mass
z: vertical position of the hip mass
r: radial distance between stance foot and hip mass
θ: angle of leg relative to z-axis

Flight Dynamics

$$\ddot{x} = 0$$

$$\ddot{z} = -g$$

$$\theta = u$$

Flight to Stance Transition

$$z - l_0 \cos(\theta) \leq 0$$

Stance Dynamics

$$m\ddot{r} - mr\dot{\theta}^2 + mgcos(\theta) - k(l_0 - r) = 0$$

$$mr^2\ddot{\theta} + 2mr\dot{r}\dot{\theta} - mgrsin(\theta) = 0$$

Stance to Flight Transition

$$r \geq l_0$$

Key Assumptions

Piecewise dynamics

Massless leg, no leg dynamics

Stance foot is attached to the ground via pin-joint

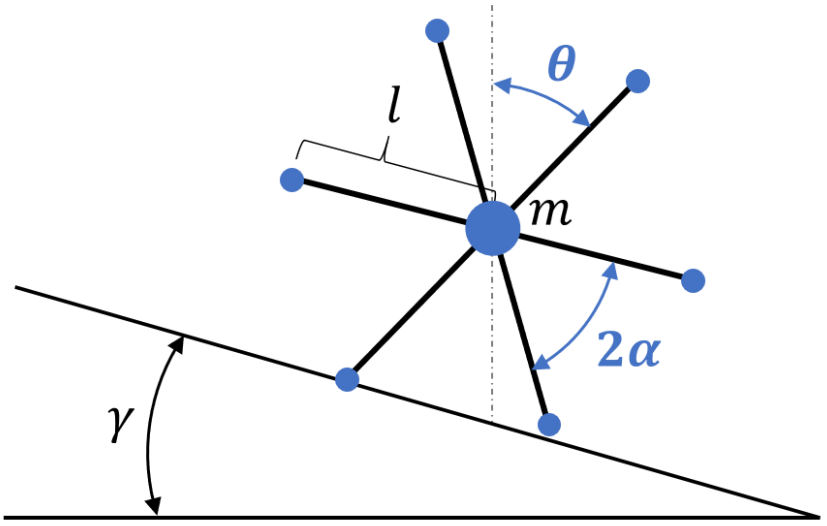


Figure 4. Rimless Wheel Model as defined in [19].

Finally, the rimless wheel was first introduced in Passive Dynamic Walking by Tad McGeer [22]. The rimless wheel emulates the heel strike and associated energy loss without the instability of the previous hybrid models. It consists of rigid legs attached to a central mass and assumes inelastic, impulsive collisions with the ground. Furthermore, the stance foot is attached to the ground via a pin joint, and the transfer of support between legs is instantaneous. Thus, the mode has hybrid dynamics, where the reset map encodes both the impulse from the collision as well as the coordinate change when the stance leg is redefined. The wheel exhibits a steady rolling cycle when the system has sufficient kinetic energy to vault the mass over the stance leg following a collision event [19].

Rimless Wheel [19]

Parameters	
Ramp angle	$0 \leq \gamma \leq \pi/2$
Spoke angle	$0 \leq \alpha \leq \pi/2$
Central mass	m
Leg length	l

States $\mathbf{x} = [\theta, \dot{\theta}]$

θ: angle of the stance leg relative to the vertical axis

Continuous Dynamics

$$\ddot{\theta} = \frac{g}{l} sin(\theta)$$

Guard Function

$$\theta = \gamma \pm \alpha$$

Discrete Dynamics

$$\theta(t^+) = \theta(t^-) - 2\alpha$$

$$\dot{\theta}(t^+) = \dot{\theta}(t^-)\cos(2\alpha)$$

Key Assumptions

Collisions with ground are inelastic/impulsive

Stance foot is attached to the ground via pin-joint

Weight transfer between legs occurs instantaneously

Below is a table describing the key features of each walking model when determining suitability for Koopman modeling (Table 1). From the previous section, key problems include the curse of dimensionality and difficulty capturing stable and unstable modes. Furthermore, walking models that do not have discrete dynamics are preferred, since Koopman Operator Theory lacks guarantees for accurate modeling for systems with discrete jumps in state. Finally, all dynamics must be a function of state. Both the rigid walker and the kneed walker have unstable behaviors and discrete dynamics. SLIP and Bipedal SLIP have no discrete jumps in

state, but the massless leg assumption, the switching between states, and the fixed angle of attacks (Bipedal SLIP) make these models difficult or undesirable for linearization. The rimless wheel has low dimensionality and no unstable behaviors, but the original formulation has discrete dynamics.

	Rigid Walker	**Kneed Walker**	**SLIP**	**Bipedal SLIP**	**Rimless Wheel**
Num. States	4	6	4	4	2
Discrete Impacts	1	2	-	-	1
Instability	Yes	Yes	Yes	Yes	No
Additional Notes	Ignores foot scuffing Instantaneous swap of stance and swing legs No slip	Knee locks into place Instantaneous swap of stance and swing legs No slip	Switches between cartesian and polar No leg dynamics No slip	Dynamics a function of foot location No leg dynamics No slip	Instantaneous swap of stance spoke No slip

Table 1. Summary of walking model suitability for use in Koopman lifted linearization. Key features include low dimensionality, stability, non-discrete dynamics and state-dependent dynamics

With the initial choice of the rigid compass walker, significant effort was made to create a viable model with the rigid walker with springs and dampers attached to the feet. However, there was no clear way to differentiate foot scuffing from heel strike without several assumptions. Furthermore, even with the presence of a linear spring, the walker would still experience a discrete jump in state when the leg impacts the ground if the foot's velocity before landing included a component normal to the stance/spring axis [23]. Thus, a single linear spring oriented along the axis of each leg was insufficient in preventing discrete dynamics. A multi-directional spring or shock absorbing element could be employed, but this would necessitate

additional position and velocity variables to monitor spring deflection in both the parallel and perpendicular directions, which could lead to a significant increase in complexity. Finally, it is not possible to artificially extend the duration of the discrete contact because the dynamics no longer become a function of state.

2.3 Collision Model

Another important choice is that of the collision model. In the rigid contact model, it is assumed there is no penetration between the feet and the ground surface. When two bodies make contact, their relative velocities immediately become zero, resulting in hybrid dynamics. The original rimless wheel formulation falls under this category.

In the viscoelastic model, objects are allowed to intersect, and the ground reaction forces are computed at each contact point as a spring damper force. These forces are a function of the objects' penetration depth and relative velocity. Oftentimes, these forces are modified to ensure unilaterality. Viscoelastic models can apply to either point contact or hydroelastic contact, where the intersection of bodies creates a repelling force based on the objects' associated strain deformation [23]. To model friction, two common solutions are to model friction as a spring damper force in the horizontal direction or to utilize a modified Coulomb model to better relate friction forces to normal force [24].

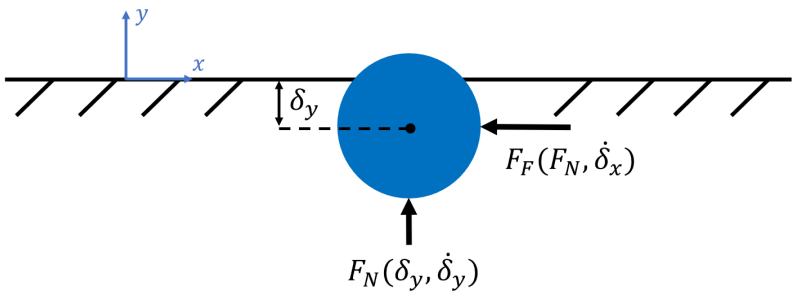


Figure 5.

Figure 5. Normal and Friction Forces for a Point-Mass. The ground reaction forces, based on the floor contact model from Khadiv et al., are nonlinear functions of the point mass's floor penetration (δ_y), and its velocity in the floor's reference frame ($\dot{\delta_x}$, $\dot{\delta_y}$) [24].

In *Compliant Floor Contact from Rigid vs compliant contact: An experimental study on biped walking* by Khadiv et al., a viscoelastic floor model is presented:

$$F_N = -k_y \tan\left(\frac{\pi}{2l_{max}}\delta_y\right) - b_y \left|\delta_y\right| \dot{\delta_y}$$

$$F_F = -\frac{2}{\pi}\tan^{-1}\left(\frac{\dot{\delta_x}}{\lambda}\right)\mu F_N$$

The normal force is composed of a spring and damper term, with k_y and b_y representing floor stiffness and floor damping respectively. Benefits of this contact model include normal force unilaterality, the ability to specify a maximum penetration depth (l_{max}), and zero ground reaction forces when leaving or coming into contact with the ground (i.e. $\delta_y = 0 \rightarrow F_N, F_F = 0$) [24].

The friction force depends on the normal force. When λ approaches zero, the friction force becomes equal to that of the Coulumb model. A larger λ yields a smoother friction model to help generate non-oscillatory friction forces (Figure 6).

This contact model is shown to properly model the impact of a bouncing ball and the empirical measurements of the SURENA II bipedal robot [24].

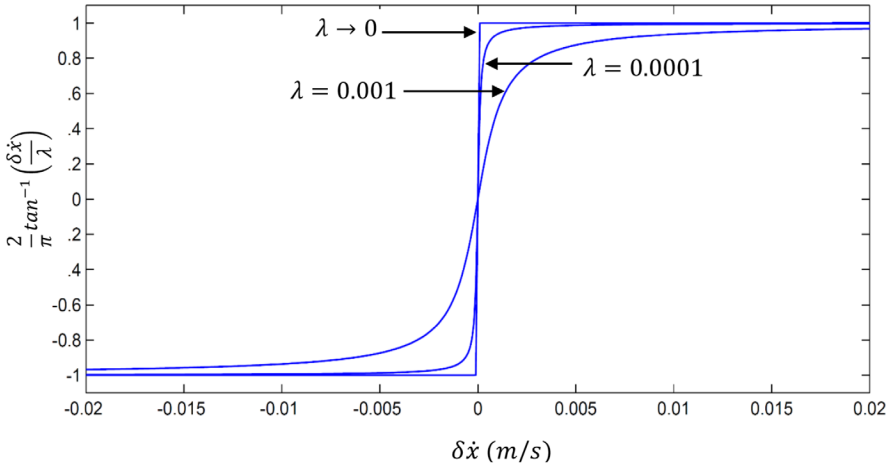


Figure 6. Variation of the Pseudo Coulomb Friction Model with respect to λ. A higher λ value smooths the discontinuity at $\delta\dot{x} = 0$, yielding better numerical results. At $\lambda = 0$, the Coulomb Friction Model is obtained. From Figure 3 of [24].

Chapter 3

Dynamics Modeling and Simulation

3.1 Rimless Wheel Modeling

For the passive Rimless Wheel model, there are only three generalized coordinates, **q** = [x, y, θ], and a total of nine states, **z** = [q, $\dot{q}$]. Coordinates x and y give the location of the central mass in the ramp reference frame. Model parameters include the central mass (M), spoke length (l), wheel inertia (I), ramp angle (γ), floor height (y_C) and additional floor parameters such as ground stiffness (k_y), ground damping (b_y), maximum penetration depth (l_{max}), pseudo-Coulomb parameter (λ), and coefficient of friction (μ). The spoke tip locations are functions of the central mass location and wheel rotation angle. There are six spokes in total, with the angle between spokes $2\alpha = \pi/3$.

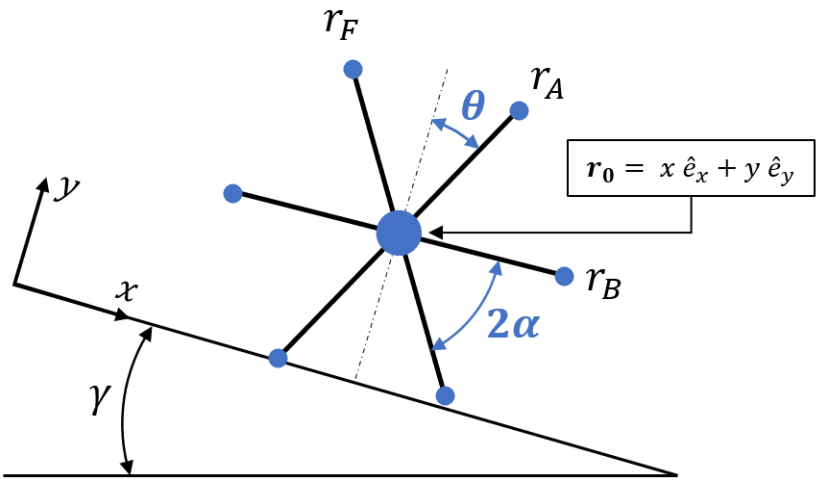


Figure 7. Rimless wheel model. The model has a ramp angle (γ), central mass (M), spoke length (l), and six evenly spaced spokes separated by angle 2α. The central mass position, $\boldsymbol{r_0}$, is measured relative to the ramp's reference frame. The wheel orientation angle θ is the angle between spoke A and the ramp's vertical axis. In this figure, only spoke tip r_D is in contact with the ramp.

The spoke tip locations ($\boldsymbol{r_A}$, $\boldsymbol{r_B}$, $\boldsymbol{r_C}$, $\boldsymbol{r_D}$, $\boldsymbol{r_E}$, $\boldsymbol{r_F}$) are a function of the central mass location, $\boldsymbol{r_0}$, and the rotation angle θ. Then, the spoke tip velocities ($\boldsymbol{v_A}$, $\boldsymbol{v_B}$, $\boldsymbol{v_C}$, $\boldsymbol{v_D}$, $\boldsymbol{v_E}$, $\boldsymbol{v_F}$) are obtained by taking their derivatives.

Positions:

$$\boldsymbol{r_0} = x\,\hat{e}_x + y\,\hat{e}_y$$
$$\boldsymbol{r_A} = \boldsymbol{r_0} + (l + l_0)\,\hat{e}_A = \boldsymbol{r_0} + (l + l_0)\,(\cos(\theta)\,\hat{e}_x + \sin(\theta)\,\hat{e}_y)$$
$$\boldsymbol{r_B} = \boldsymbol{r_0} + (l + l_0)\,\hat{e}_B = \boldsymbol{r_0} + (l + l_0)\,(\cos(\theta + \tfrac{\pi}{6})\,\hat{e}_x + \sin(\theta + \tfrac{\pi}{6})\,\hat{e}_y)$$
$$\vdots$$
$$\boldsymbol{r_F} = \boldsymbol{r_0} + (l + l_0)\,\hat{e}_F = \boldsymbol{r_0} + (l + l_0)\,(\cos(\theta)\,\hat{e}_x + \sin(\theta + \tfrac{5\pi}{6})\,\hat{e}_y)$$

Velocities:

$$\boldsymbol{v_0} = \frac{d}{dt}\boldsymbol{r_0} = \dot{x}\,\hat{e}_x + \dot{y}\,\hat{e}_y$$
$$\boldsymbol{v_A} = \frac{d}{dt}\boldsymbol{r_A} = \frac{d}{dt}\boldsymbol{r_0} + (l + l_0)\frac{d}{dt}\hat{e}_A = \frac{d}{dt}\boldsymbol{r_0} + (l + l_0)\,(-\dot{\theta}\sin(\theta)\,\hat{e}_x + \dot{\theta}\cos(\theta)\,\hat{e}_y)$$

$$\boldsymbol{v_B} = \frac{d}{dt}\boldsymbol{r_B} = \frac{d}{dt}\boldsymbol{r_0} + (\mathrm{l} + l_0)\frac{d}{dt}\hat{e}_B = \frac{d}{dt}\boldsymbol{r_0} + (\mathrm{l} + l_0)\left(-\dot{\theta}\sin(\theta + \frac{\pi}{6})\,\hat{e}_x + \dot{\theta}\cos(\theta + \frac{\pi}{6})\,\hat{e}_y\right)$$

$$\vdots$$

$$\boldsymbol{v_F} = \frac{d}{dt}\boldsymbol{r_F} = \frac{d}{dt}\boldsymbol{r_0} + (\mathrm{l} + l_0)\frac{d}{dt}\hat{e}_F = \frac{d}{dt}\boldsymbol{r_0} + (\mathrm{l} + l_0)\left(-\dot{\theta}\sin(\theta + \frac{5\pi}{6})\,\hat{e}_x + \dot{\theta}\cos(\theta + \frac{5\pi}{6})\,\hat{e}_y\right)$$

This model does not have spoke masses, so the kinetic energy term consists of the central mass and the rotational inertia of the wheel. However, the addition of spoke masses could be subsumed within a larger rotational inertia parameter value using the same formulation:

$$\mathrm{T} = \frac{1}{2}\mathrm{M}\,(\boldsymbol{v_0}\cdot\boldsymbol{v_0}) + \frac{1}{2}\mathrm{I}\,\dot{\theta}^2$$

$$\mathrm{V} = \mathrm{M\ g}\,(\boldsymbol{r_0}\cdot\hat{e}_{\mathrm{ramp}}) = \mathrm{M\ g}\,(\boldsymbol{r_0}\cdot(-\sin(\gamma)\,\hat{e}_x + \cos(\gamma)\,\hat{e}_y))$$

We derive the equations of motion using the Euler-Lagrange equations. Note that the resulting terms are all functions of $\mathbf{q}$ or $\mathbf{z} = [\boldsymbol{q}, \dot{\boldsymbol{q}}]$.

$$\mathrm{L(z)} = \mathrm{T\text{-}V}$$

$$\mathbf{g(z)} = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\boldsymbol{q}}}\right) - \left(\frac{\partial L}{\partial \boldsymbol{q}}\right) - \sum_i \left(\frac{\partial \boldsymbol{r_i}}{\partial \boldsymbol{q}}\right)^T \cdot \boldsymbol{F}_i = \boldsymbol{M(q)}\ddot{\boldsymbol{q}} + \boldsymbol{V(q,\dot{q})} + \boldsymbol{G(q)} - \boldsymbol{Q} = 0$$

$$\mathbf{A(q)} = \left(\frac{\partial \mathbf{g}}{\partial \ddot{\boldsymbol{q}}}\right) = \boldsymbol{M(q)}$$

$$\mathbf{b(z)} = \mathbf{A}\ddot{\boldsymbol{q}} - \boldsymbol{g} = -(\boldsymbol{V(q,\dot{q})} + \boldsymbol{G(q)} - \boldsymbol{Q})$$

$$\therefore \ddot{\boldsymbol{q}} = \mathbf{A^{-1}b}$$

The ground contact is incorporated into the equations of motion via the generalized forces term $\boldsymbol{Q} = \sum_i \boldsymbol{Q_i} = \sum_i \left(\frac{\partial \boldsymbol{r_i}}{\partial \boldsymbol{q}}\right)^T \cdot \boldsymbol{F}_i$. Recall the following floor contact parameters: ground stiffness (k_y), ground damping (b_y), maximum penetration depth (l_{max}), pseudo-Coulomb parameter (λ), and coefficient of friction (μ).

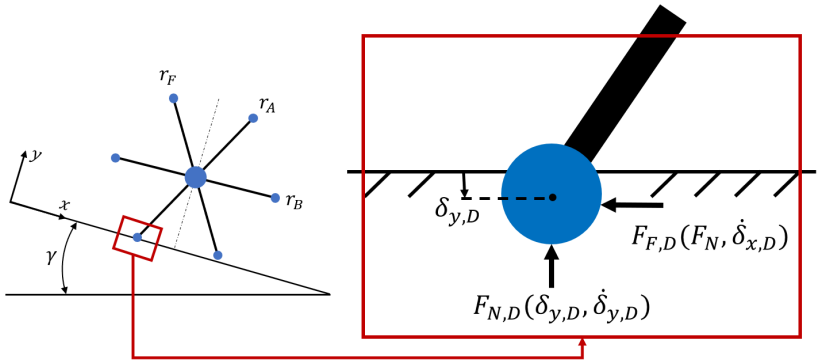


Figure 8. Ground Reaction Forces for Spoke D. Ground reaction forces only exist for spoke tips below the ground

For a generic spoke i, the generalized force contributions for ground reaction forces at the spoke tip $\boldsymbol{r_i}$ is given below:

$$\delta_{y,i} = \boldsymbol{r_i} \cdot \hat{e}_y - y_C$$

if $\delta_{y,i} < 0$

$$\dot{\delta}_{x,i} = \boldsymbol{v_i} \cdot \hat{e}_x$$

$$\dot{\delta}_{y,i} = \boldsymbol{v_i} \cdot \hat{e}_y$$

$$F_{N,i} = -k_y \tan\left(\frac{\pi}{2l_{max}} \delta_{y,i}\right) - b_y \left|\delta_{y,i}\right| \dot{\delta}_{y,i}$$

$$F_{F,i} = -\frac{2}{\pi} \tan^{-1}\left(\frac{\dot{\delta}_{x,i}}{\lambda}\right) \mu F_{N,i}$$

$$\boldsymbol{Q_i} = \left(\frac{\partial \mathbf{r_i}}{\partial \boldsymbol{q}}\right)^{\mathrm{T}} * [F_{F,i}; F_{N,i}]$$

3.2 MATLAB Simulation

The Rimless Wheel model with a viscoelastic contact model is implemented in MATLAB. For each time step t in our simulation, we obtain the accelerations at time t using the manipulator equations and calculating the generalized forces. Then, we advance using the semi-implicit Euler method for integration. The simulation is run with time steps of $\Delta t = 0.001$ seconds.

for $t = 1{:}1{:}t_f$

$\boldsymbol{z_t} = [\boldsymbol{q_t}, \dot{\boldsymbol{q}}_t]$

$\boldsymbol{Q} = \sum_i \boldsymbol{Q_i}$

$\ddot{\boldsymbol{q}}_{\boldsymbol{t}} = (\mathbf{b}(\boldsymbol{z_t}) + \boldsymbol{Q}\,)/\mathbf{A}(\boldsymbol{q_t})$

$\dot{\boldsymbol{q}}_{\boldsymbol{t+1}} = \dot{\boldsymbol{q}}_{\boldsymbol{t}} + \Delta t\, \ddot{\boldsymbol{q}}_t$

$\boldsymbol{q_{t+1}} = \boldsymbol{q_t} + \Delta t\, \dot{\boldsymbol{q}}_{\boldsymbol{t+1}}$

A variety of different floor stiffnesses and ground damping constants were tested for the rimless wheel, but the parameters chosen in Table 2 led to numerical stability while still having a reasonably stiff floor. Despite the high maximum ground penetration depth ($l_{max} = 0.2$ m), the largest depths for the two trajectories shown in Figure 9 and Figure 11 were 0.0166 m and 0.0089 m respectively.

Parameter	Symbol	Value
Central Mass [kg]	M	1.0
Spoke Length [m]	l	1.0
Wheel Inertia [kg · m²]	I	0.1
Ramp Angle [°]	γ	20.0
Floor Height [m]	y_C	-1.0
Ground Stiffness Coefficient	k_y	1.0×10^3
Ground Damping Coefficient	b_y	1.0×10^4
Maximum Penetration Depth [m]	l_{max}	0.2
Pseudo-Coulomb Parameter	λ	1.0×10^{-3}
Coefficient of Friction	μ	0.5

Table 2. Parameter Values for Rimless Wheel MATLAB Simulation.

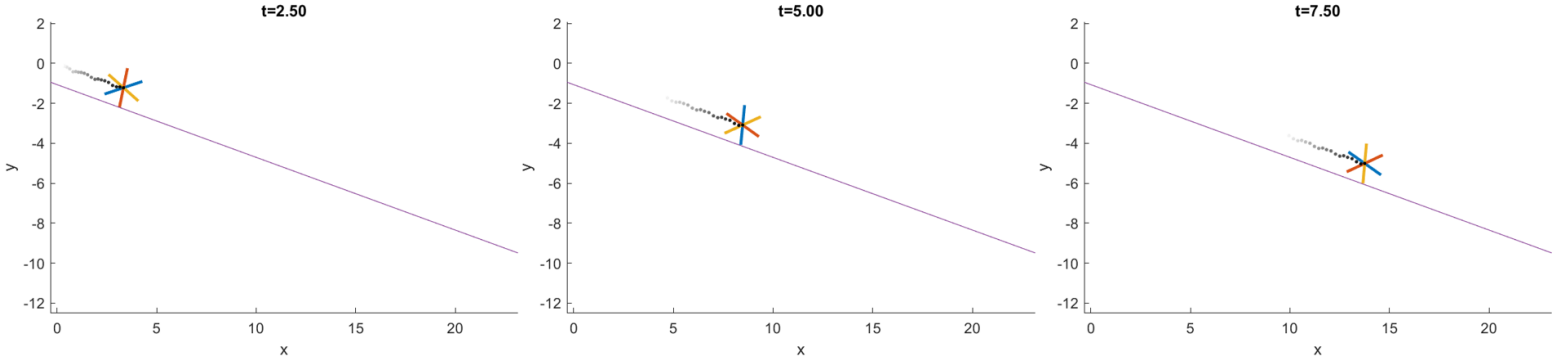


Figure 9. Rimless wheel MATLAB model passive rolling trajectory. Model shown at t = 2.50, 5.00, and 7.50 seconds. Dots represent past center of mass locations, each separated by 0.1 seconds. The initial condition for this trajectory was $z_0 = [0, -0.01, 3, 0, 0, -5]$.

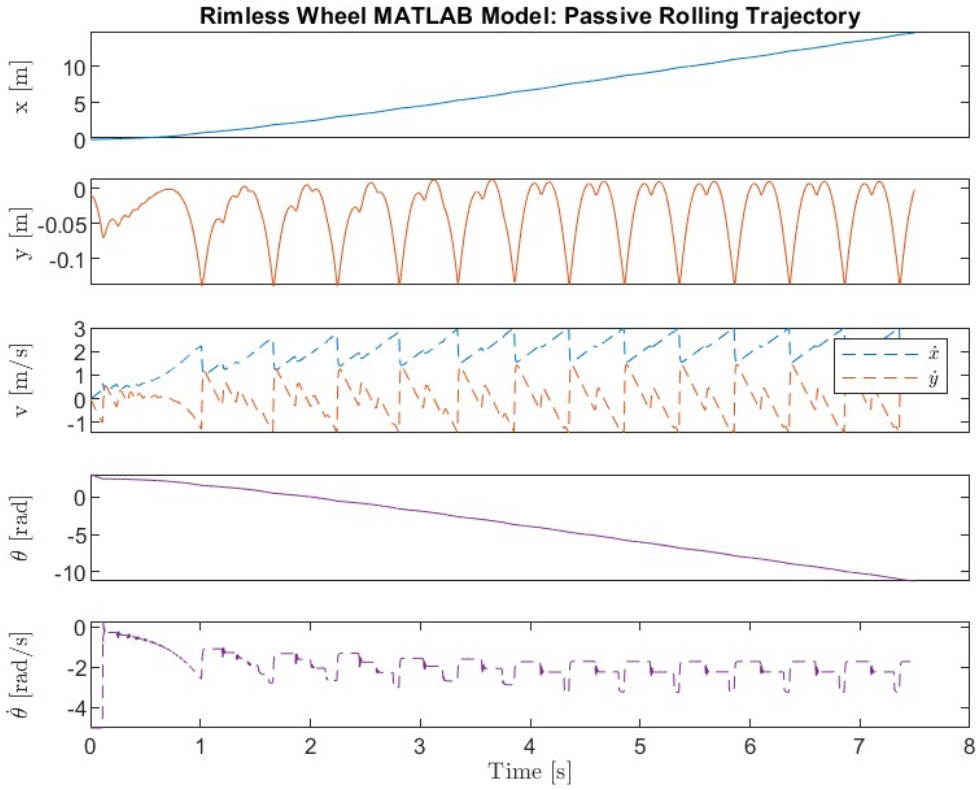


Figure 10. Rimless wheel MATLAB model passive rolling trajectory states vs time. Wheel converges to a stable periodic solution after ~1.5 seconds. Collisions with ground produce sharp but continuous spikes in velocity.

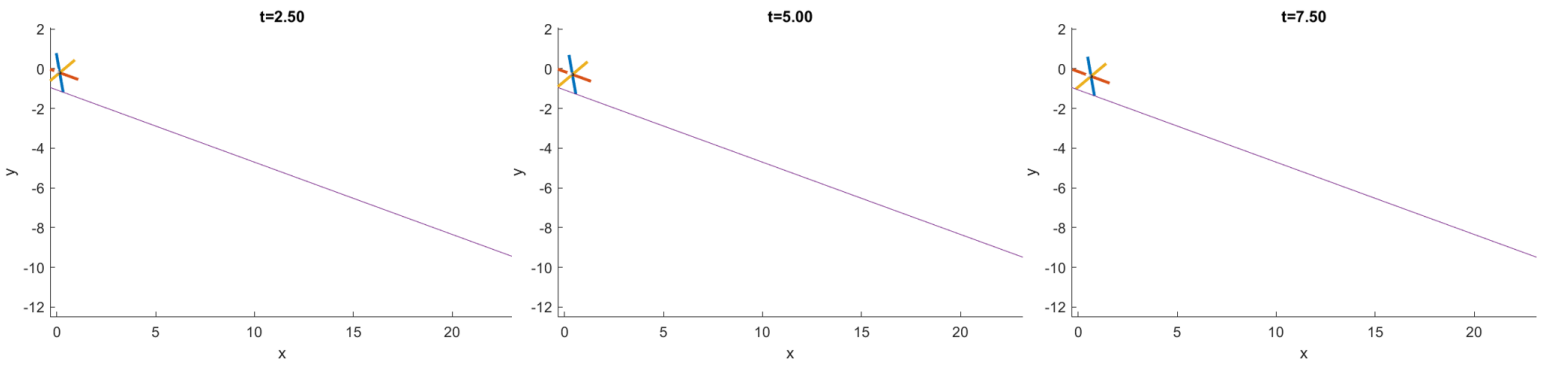


Figure 11. Rimless wheel MATLAB model stance trajectory. Model shown at t = 2.50, 5.00, and 7.50 seconds. Dots represent past center of mass locations, each separated by 0.1 seconds. The small white dot on the model represents the center of mass location from 2 seconds ago, indicating that the model is gradually slipping down the ramp. The initial condition for this trajectory was $z_0 = [0, -0.01, \frac{\pi}{8}, 0, 0, 0]$.

Figure 12. Rimless wheel MATLAB model stance trajectory states vs time. Small oscillations in velocity occur due to repeated ground contact. The x-position plot shows the model slipping down the ramp.

Chapter 4

Passive Koopman Models

4.1 Passive Koopman Models Using Radial Basis Functions

Following the construction of the passive MATLAB model, data was collected for use in the passive Koopman model Specifically, a region of space centered around the origin was randomly sampled. To ensure that floor contact was properly modeled, at least 20% of every trajectory started with ground contact. The trajectories were capped with maximum initial energy (including the potential energy stored in the viscoelastic springs due to ground penetration), and trajectories that deviated too far from the origin were removed. Each trajectory lasted a total of two seconds with data sampled at a rate of dt = 0.05 seconds. Data sets were compiled from 100 and 1000 trajectories. A third validation data set was created of 50 trajectories.

The first Koopman models were obtained with RBF observables. The Gaussian RBFs were placed in the state space of the 100 and 1000-trajectory data sets, with the k-means++ clustering algorithm determining center placement. The equation for the Gaussian radial basis function is given below:

$$\phi_i(x) = \exp\left(-\left(\frac{\left|\left|x - c_i\right|\right|_2}{\epsilon}\right)^2\right)$$

Important hyperparameters include the dilation factor (ϵ), the center locations (c_i), and the number of radial basis functions (N). Too many observables can lead to overfitting and poor estimation performance, while too few will fail to capture the dynamics of the system. All six passive RBF Koopman models in this section, used ϵ =0.4.

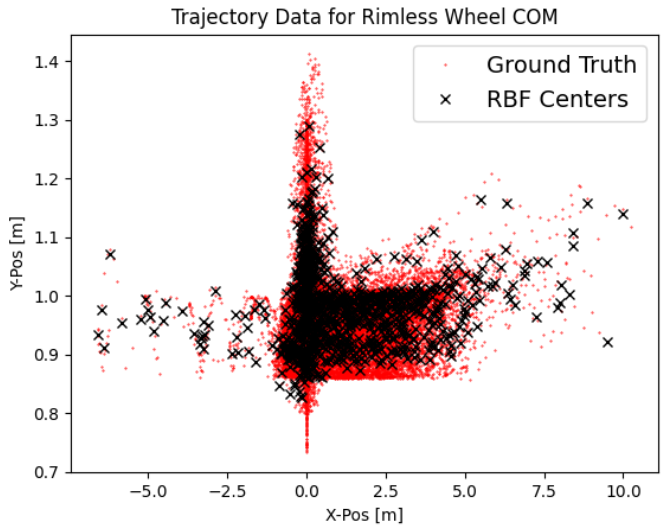


Figure 13. Placement of RBF Centers Using kmeans++. Depicted are 1000 RBF centers based on the 1000-trajectory data set. The Rimless Wheel COM positions are shown in red while the RBF centers are shown in black. The RBF centers are also functions of wheel angle θ and velocities, but these dimensions are not plotted for simplicity.

Once the radial basis functions were obtained, the data set was lifted such that for each data point z_t, we obtain $\chi_t = [z_t, \phi_1(z_t), \phi_2(z_t), \dots, \phi_N(z_t)]$. We included the original states z_t in the lifted space for ease of use when implementing MPC. In this formulation, the A matrix was obtained using a least squares regression of the χ_t and χ_{t+1} lifted data sets: $A = \chi_{t+1}\,\chi_t^{\#}$. Larger data sets should yield better results, assuming that the data is uniformly distributed, as they reduce the effects of randomness.

It is beneficial to have the x-position as a state when implementing MPC. However, since the dynamics of the rimless wheel are invariant with respect to the x-position, the system behaves identically whether at x = 0 m or x = 100 m. Consequently, when constructing the Koopman Operator model, it is advantageous to remove the x-position from the training data to enhance model accuracy. The passive Koopman models presented below were initially developed with a dependency on x, but we found that performance issues only emerge when the x-position is far from the origin. This issue is explored in further detail in Section 685.5.

Mode power is a method to determine the relative importance of a given mode within a data set. Mode powers are evaluated for each data point within a training set and then averaged. Eigenvalues with low mode power are typically associated with noise and can be removed with little consequence [25]. To calculate the mode power, the eigenvalues and eigenvectors of the linearized model are obtained. For each element in the dataset n the lifted state χ_n is projected onto the eigenvectors such that $Z_n = \chi_n' V$. The element $Z_{n,i}$ shows the contribution of the eigenvector V_i for the input data z_n. Finally, the mode power of the eigenvector V_i is given by $\frac{1}{N}\sum_{n=1}^{N} Z_{n,i}$. Figure 14 shows the Mode Power plots for Koopman Models with 10, 100, and 1000 RBFs. We see the emergence of oscillatory behaviors with greater number of observables. However, the more "important" poles belong to lower frequencies or are oriented along the x-axis. It is difficult to read the mode power plots when the number of observables is high, as the many poles overlap.

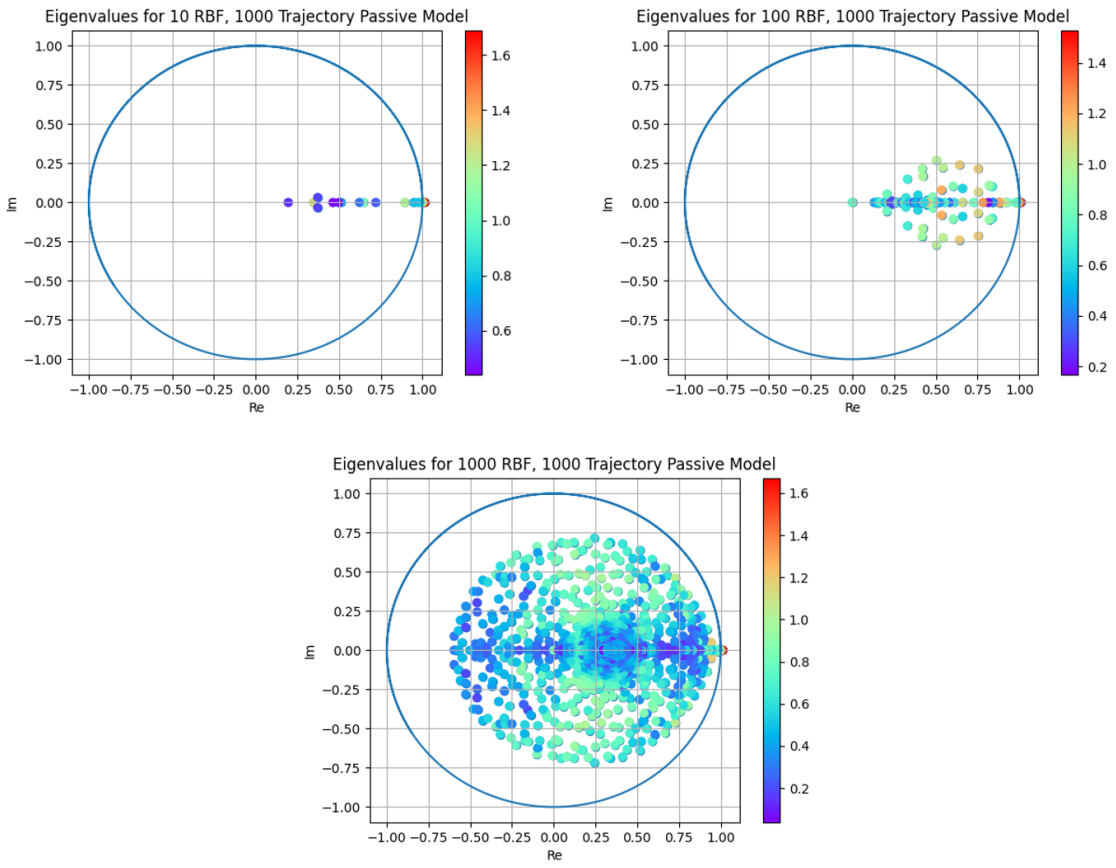


Figure 14. Mode Power Plots for 10, 100, and 1000 RBF Passive Koopman Models. For 10 RBF, the poles are mostly aligned along the x-axis. More imaginary poles/oscillatory behaviors appear with more observables. All three models use the 1000-trajectory training set.

Mean Squared Error (MSE) is a common metric used to evaluate Koopman prediction accuracy. For a test data set with N total data points, the MSE is given by

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(z_i - \hat{z}_i)^2$$

Figure 15 depicts the MSE vs. time of a Koopman Model for predictions of different time horizons. The training data is centered around the origin where the trajectories are initialized. As the trajectories evolve, they disperse, resulting in more varied data over time. This dispersion

suggests that the Koopman model might be more accurate at the beginning of a trajectory compared to the end. However, the figure indicates that higher errors actually occur at the start of a trajectory, possibly due to the increased likelihood of ground contact, resulting in more unpredictable movement.

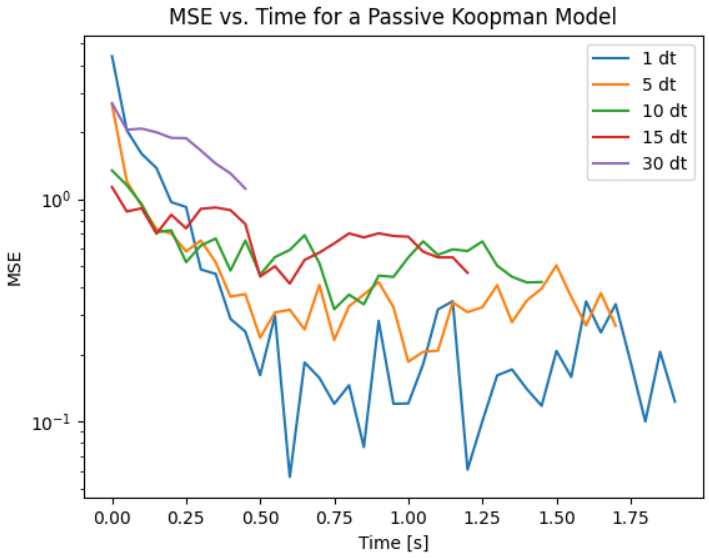


Figure 15. Mean Squared Error vs. Time for a Passive Koopman Model. MSE calculated for 50 trajectories not including in training/model creation There is seemingly a downwards trend between trajectory time and prediction accuracy. The Koopman model was created with the 1000-trajectory training set and 1000 RBFs.

Finally, Figure 16, Figure 17, and Figure 18 depict the tracking performance for different time horizons. Figure 16 shows the Koopman model's predictions for different time horizons for a passive rolling trajectory. While the model captures the rolling motion accurately initially, its performance degrades for the 10- and 15-time step predictions. Figure 17 examines prediction performance on the same trajectory with ground truth updates every 1, 5, 10, and 15 steps. Unsurprisingly, accuracy is improved with more frequent updates, as the model is corrected often and errors are not allowed to accumulate. Figure 18 tests the Koopman model on a

challenging backward, uphill trajectory, demonstrating worse performance due to the trajectory's rarity in the training data, with accuracy declining markedly over longer time horizons. All figures use the same model, trained on the 1000-trajectory data set with 1000 radial basis functions ($\epsilon = 0.4$).

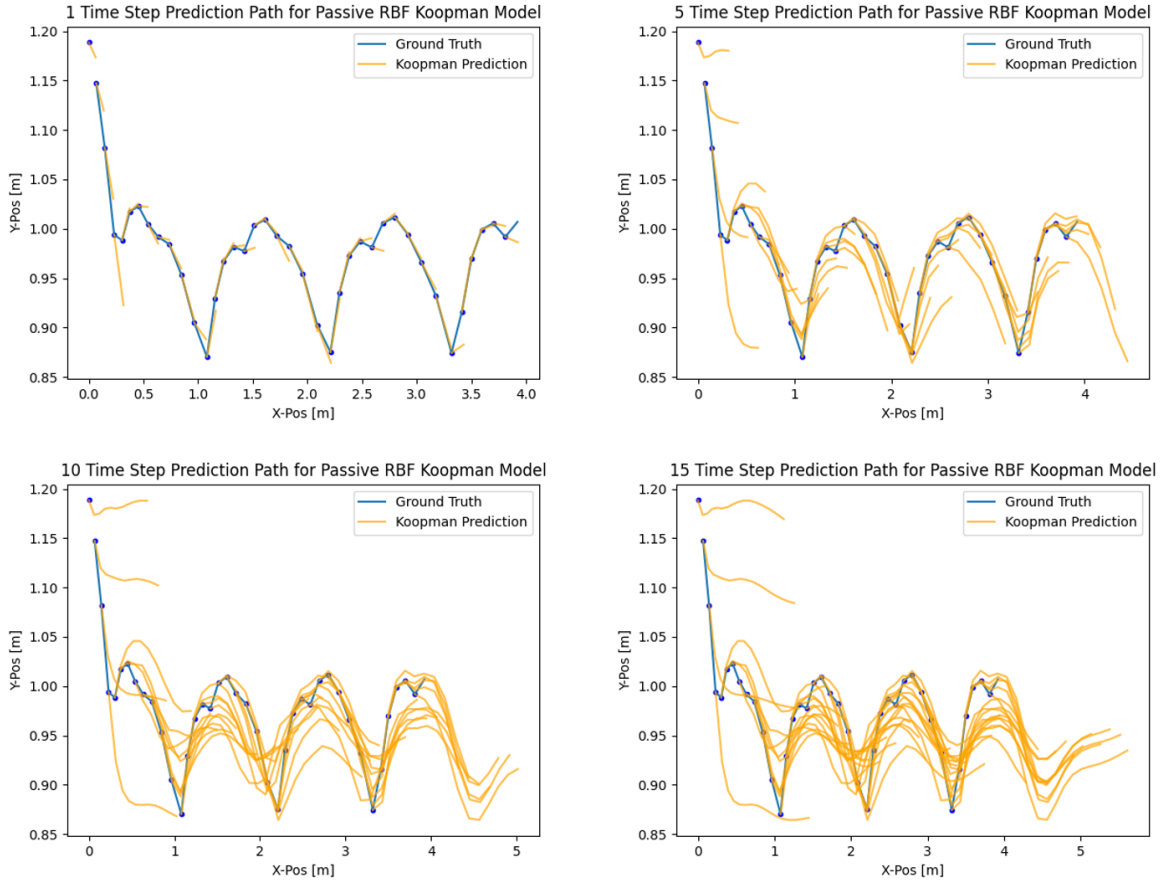


Figure 16. Passive Koopman Model Tracking Performance for Different Time Horizons. At every time step, the model is given the ground truth (depicted by the blue dot) and then blindly predicts the next N time-step (depicted by the orange line going off each blue dot). The Koopman Model is capable of capturing the ground truth rolling motion but with decreasing accuracy over time. This is exemplified by the smoothing out of the peaks and valleys in the 15 Time Step Prediction plot. The Koopman model was created with the 1000-trajectory training set and 1000 RBFs.

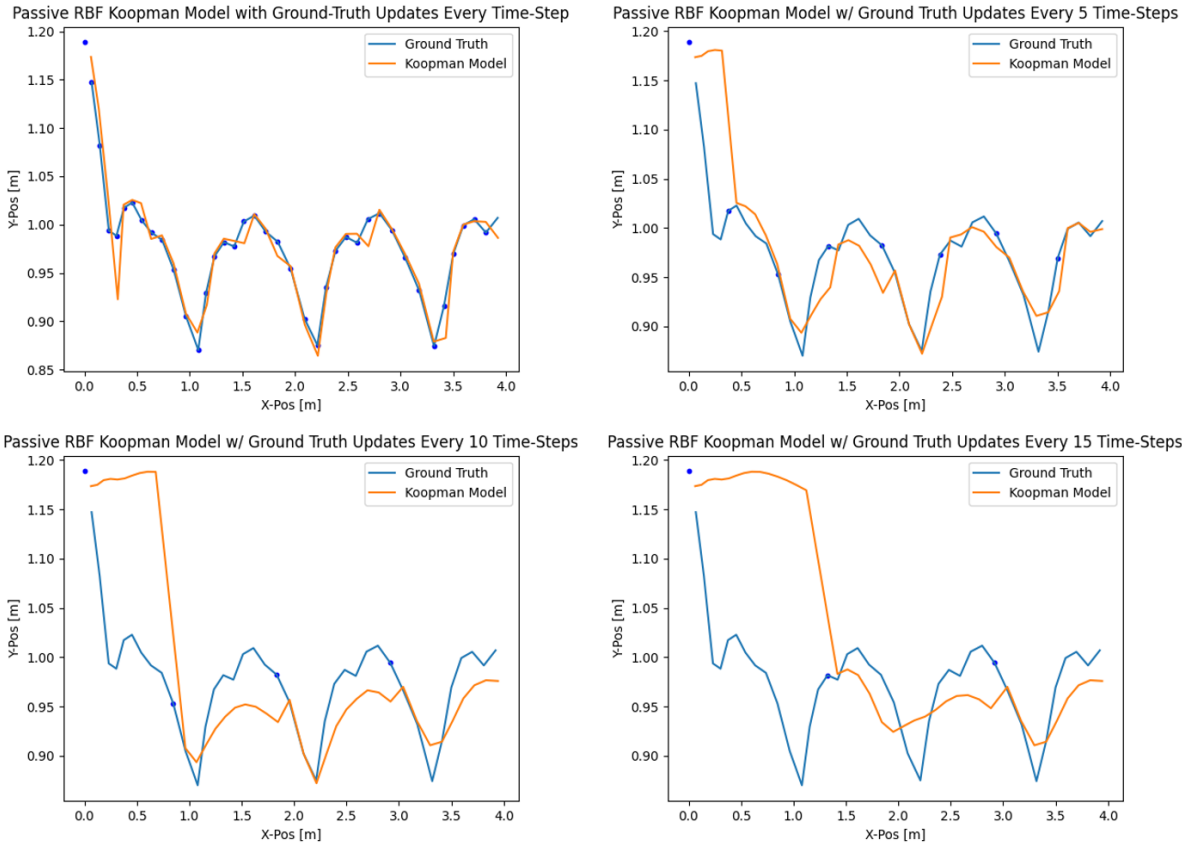


Figure 17. Passive Koopman Model Tracking Performance with Updates Every 1, 5, 10, and 15 Time-Steps. The blue dot depicts data points fed into the model as the ground truth. If the model is very incorrect, these updates will result in a sharp change in state. We see better performance when the update rate is high. The Koopman model was created with the 1000-trajectory training set and 1000 RBFs.

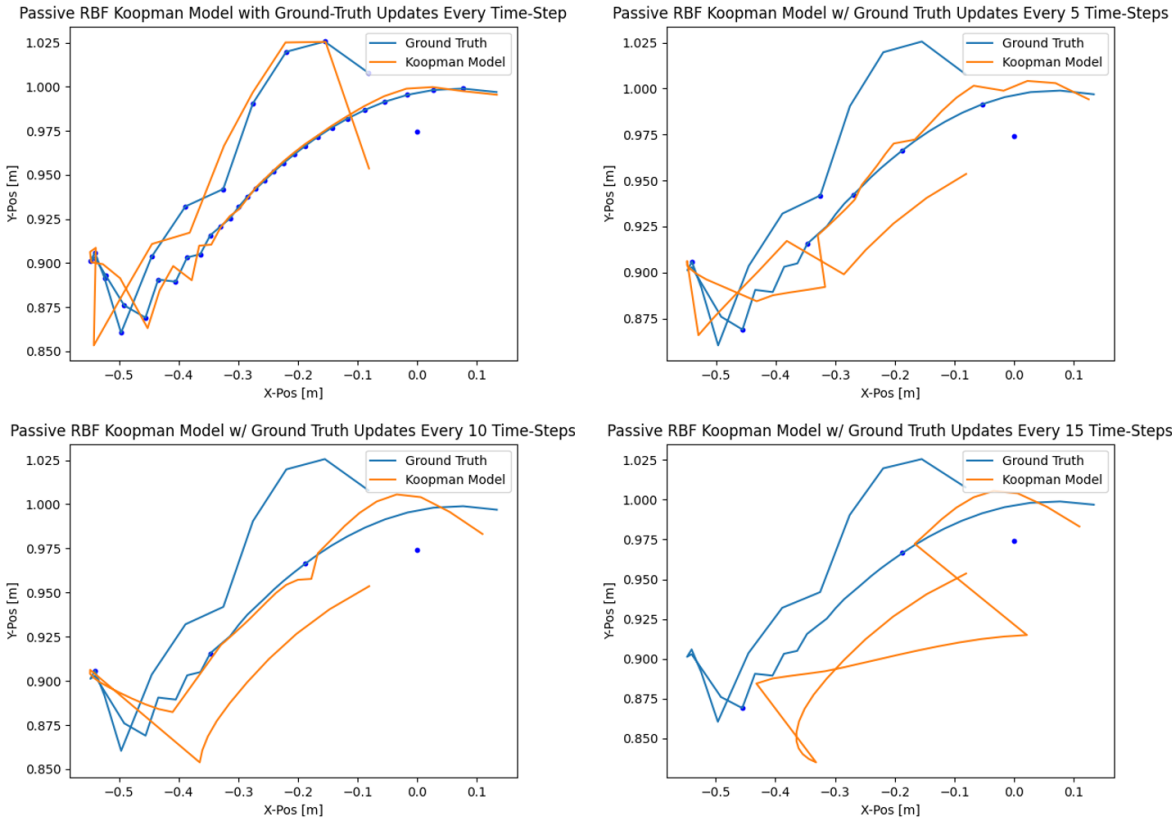


Figure 18. Passive Koopman Model Tracking Performance for a Challenging Trajectory with Updates Every 1, 5, 10, and 15 Time-Steps. Using the same Koopman model as Figure 17, the model is tested for a trajectory that travels backward, up the ramp. The tracking performance is visibly worse, as such trajectories are not common in the training data set.

4.2 Passive Koopman Models Using Neural Net Generated Observables

A second batch of Koopman models was made using Neural Net generated observables. The network structure is shown in Figure 19. The state z_t is fed into the encoder and outputs observables g_t. Then, the states are concatenated to create the lifted state χ_t and fed into the linear layer, which outputs the lifted state χ_{t+1}. The A matrix is obtained by taking the weight matrix from the final linear layer. The encoders have two hidden layers with rectified linear unit (ReLu) activation units. The encoder and linear layer structure is essentially that of [26].

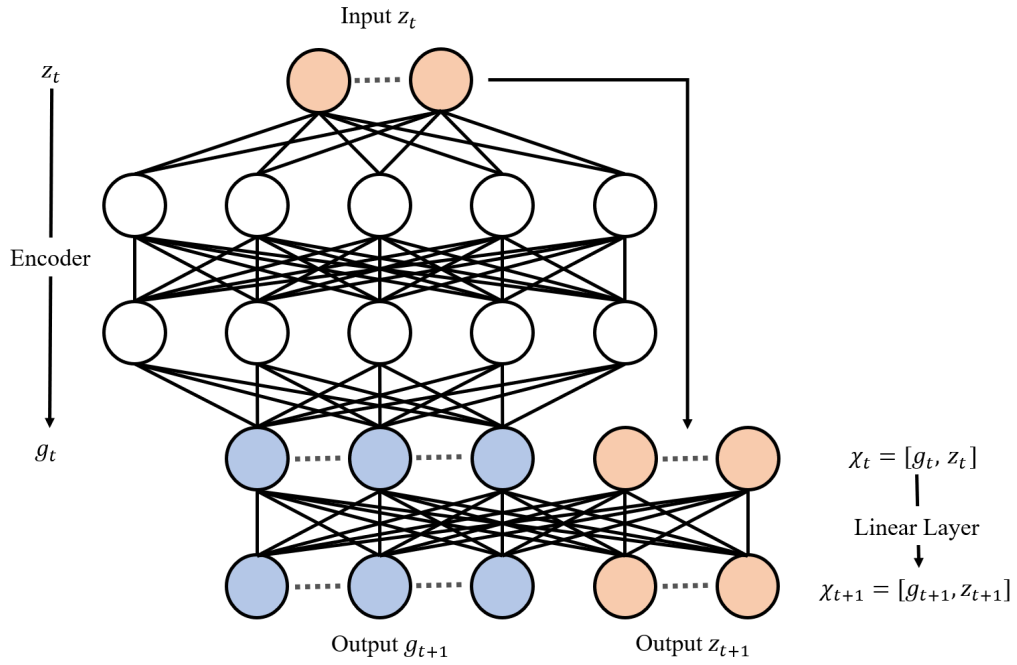


Figure 19. Neural Network Structure. The fully-connected encoder network generates the observable functions g_t based on the input data z_t. Then, the original states are appended to the observables to obtain the lifted state χ_t. The final linear layer advances the lifted state one-time step. In this figure, orange denotes states and blue denotes observables g.

The Neural Network was trained on the 100- and 1000- trajectory data sets defined in Section 3.2. The network was trained on the standard Deep Koopman loss function that takes the sum of the MSE for the state prediction and the MSE for the observable prediction [27].

$$L = MSE(\hat{z}_{t+1}, z_{t+1}) + \gamma MSE\ (\hat{g}_{t+1}, g_{t+1})$$

The parameter γ weights the observable prediction accuracy relative to the state prediction accuracy. For the generated models, γ is kept fixed at 1.

Four models were made using Neural Net generated observables. The Mode Powers for the 30-observable models are shown in Figure 20. Table 3 shows the relative performance between all ten models generated in this chapter. Mean squared error is obtained over a 50-trajectory validation data set. We see the best performer was the 1000 observable RBF model generated with the 1000-trajectory data set. Generally, the Koopman models using RBFs that were trained with more observables and the larger data set performed best. The Neural Net Koopman models had the lowest MSE for 1-time step predictions, but were poor at predicting for longer time horizons. In the future, additional terms could be added to the loss function to help improve long term accuracy by calculating the MSE of future predictions [27]. The Neural Net models with a high number of observables and low amounts of training data performed the worst, perhaps due to overfitting.

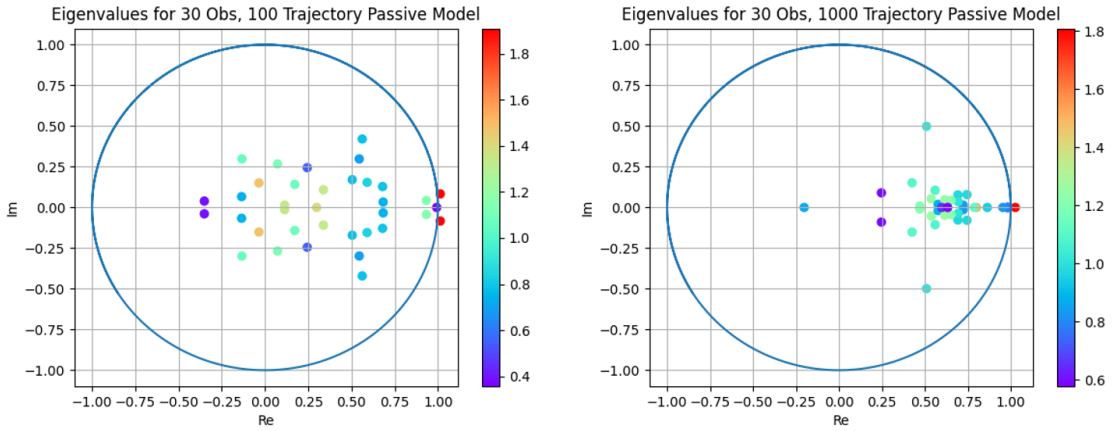


Figure 20. Mode Power Plots for 30 NN Generated Observable Koopman Models. Models were trained with the 100-trajectory data set (left) and the 1000-trajectory data set (right). The 100-trajectory model has more oscillatory poles at higher frequencies.

			1 dt	5 dt	10 dt	15 dt	30 dt
		10	1.55	0.92	0.91	1.08	2.73
		100	1.72	1.37	1.23	1.36	3.68
	100	1000	0.66	0.53	0.59	0.69	1.87
		10	0.66	0.78	0.80	0.91	2.17
		100	0.61	0.72	0.78	0.88	2.16
RBF	1000	1000	0.46	0.48	0.60	0.69	1.81
		30	0.39	1.00	1.27	1.63	4.25
	100	100	0.45	1.26	1.36×10^{1}	7.77×10^{2}	3.55×10^{8}
		30	0.34	0.96	1.81	2.81	7.09
NN	1000	100	0.32	0.78	1.13	1.40	3.51

Table 3. Comparison of Different Passive Koopman Models. The leftmost column describes the type of observables used in the model, followed by the number of trajectories in the data set, and finally the number of observables. The MSE is evaluated for a 1-, 5-, 10-, 15-, and 30-time step prediction over the course of 50 trajectories. The Koopman models trained on the largest dataset performed consistently better than those trained on the smaller one. We see clear overfitting with the 100 trajectory, 100 neural net generated observable model, despite reasonable 1-time step MSE.

Chapter 5

Actuated Koopman Models

5.1 Actuation of the Rimless Wheel

In recent years, the study of the Rimless Wheel has garnered significant attention, particularly in the design of spring-like, impact-reducing rimless wheels. However, despite these impact-absorbing elements, these rimless wheel models typically utilize discrete dynamics. The viscoelastic wheel developed by Kawamoto et al. exhibits discrete impacts both when the spoke contacts the ground and when the spoke leg returns to its free length [28]. Similarly, the models proposed by Sanchez et al. and Hanazawa et al. utilize rimless wheels with telescoping, spring-loaded legs with discrete dynamics upon contact with the floor. Interestingly, these three models do not control the leg length but rather the angle between a torso and the ground [28], [29], [30]. While the integration of a torso into the model presents an interesting avenue for future research, the primary objective of this thesis is to develop a model with interesting ground interaction dynamics.

The model presented in this section is designed with a prismatic actuator at the tip of each spoke (Figure 21). For a six spoke model, we add a total of twelve states for the actuators so $\mathbf{q}$ = [x, y, θ, ϕ_A, $\phi_B, \phi_C, \phi_D, \phi_E, \phi_F$] and $\mathbf{z} = [q, \dot{q}]$. These actuators are allowed to expand or retract a maximum set limit, and have a maximum control input of $|u_n| \leq 1$ m/s. As before, model parameters include the central mass (M), spoke length (l), wheel inertia (I), ramp angle (γ), floor

height (y_C) and now actuator refence length (l_0). Additional floor parameters such as ground stiffness (k_y), ground damping (b_y), maximum penetration depth (l_{max}), pseudo-Coulomb parameter (λ), and coefficient of friction (μ). The model and floor contact parameter values are carried over from the passive model case. Crucially, we assume the actuators are controlled with a high-fidelity control loop, so the actuator dynamics are directly defined by the control input: $\dot{\phi}_n(t) = u_n(t)$.

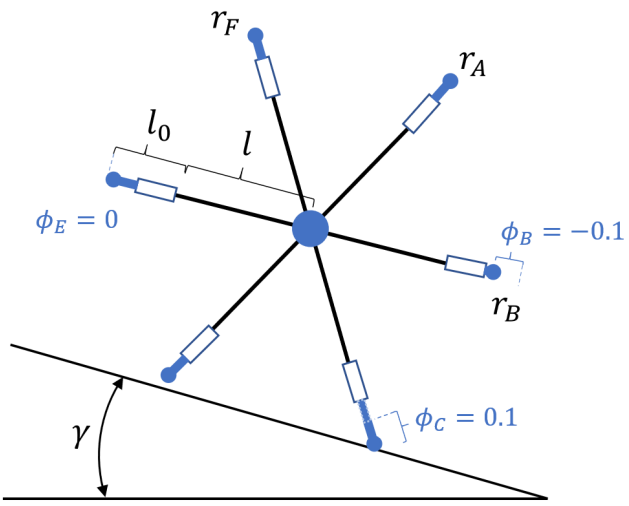


Figure 21. Rimless Wheel with Prismatic Actuators. The spokes have length l as well as the actuator reference length l_0. Actuator states ϕ_n are measured relative to this reference length. In this figure, all actuators are set to $\phi_n = 0$ except for $\phi_B = -0.1$ and $\phi_C = 0.1$.

5.2 Control Coherent Koopman Formulation

The Control Coherent Koopman requires a linear actuation subsystem. In other words, we seek an actuation method that is linear in control and that separates the input from the other state variables. Below, we show how the actuator subsystem is separated for the actuated rimless wheel:

t = 0:
- Obtain states $z_{t=0} = [q_{t=0}, \dot{q}_{t=0}]$
- Calculate GRF and convert to generalized forces
  - $F_N(z_{t=0})$ and $F_F(z_{t=0})$ → $Q_C(z_{t=0})$
- Obtain acceleration
  - $\ddot{q}_{t=0} = (\mathrm{b}(z_{t=0}) + Q_C)/\mathrm{M}(q_{t=0})$
- Implement control
  - $\dot{\varphi}_{N,\,t=0} = u_{N,\,t=0}$

t = 0.05:
- (Integrate) Obtain $z_{t=0.05} = [q_{t=0.05}, \dot{q}_{t=0.05}]$.
- Calculate GRF and convert to generalized forces
  - $F_N(z_{t=0.05})$ and $F_F(z_{t=0.05})$ → $Q_C(z_{t=0.05})$
- Obtain acceleration
  - $\ddot{q}_{t=0.05} = (\mathrm{b}(z_{t=0.05}) + Q_C)/\mathrm{M}(q_{t=0.05})$
- Implement control…

Where q = [x, y, θ, $\varphi_A, \varphi_B, \ldots \varphi_F$] and the text in red denotes variables that have been affected by control input $u_{t=0}$. Depending on when the input is incorporated, we are able to quarantine the effects of the control input $u_{t=0}$ to $\dot{\varphi}_{N,\,t=0}$ and future time steps. However, this method requires implementing the control after the ground reaction forces are calculated, which is an important modeling decision. We leverage the viscoelastic floor to separate the velocity of the prismatic joint and the location/velocity of the central mass [x, y θ]. If the floor was implemented as a hard constraint (i.e. actuator tip $\boldsymbol{r_N}$ is pinned to the floor like in the original rimless wheel hybrid

model), the velocity of the prismatic would directly result in a change in velocity of the central mass in the same time step. With a rigid contact model, a spring and mass on each spoke is necessary to separate the actuator from affecting the other state variables.

5.3 Three-Actuator Model

The three-actuator model is motivated by the curse of dimensionality. With an actuator attached to every spoke, there is a total of 18 states. To keep the dimensions low, the symmetry of the rimless wheel model is leveraged. Koopman models are accurate for relatively short periods of time, so if the constructed Koopman model is expected to be accurate for up to 20-time steps (one second), it should experience about one to two impacts with the ground in that time. Thus, we can create a model where the spokes closest to the ground have actuators ($\boldsymbol{r_A}$(x, y, θ, φ_A), $\boldsymbol{r_B}$ (x, y, θ, φ_B), $\boldsymbol{r_F}$ (x, y, θ, φ_F)), and those near the top are passive ($\boldsymbol{r_C}$ (x, y, θ), $\boldsymbol{r_D}$ (x, y, θ), $\boldsymbol{r_E}$ (x, y, θ)). By removing the actuator states for half of the spokes, the number of states decreases from 18 to 12.

A change of coordinates is required to utilize the three-actuator Koopman model due to the decreased number of spokes and the training data's limited range of initial conditions. For example, to predict the motion of a fully actuated, 18-dimensional rimless wheel we first determine which three spokes are closest to the ground, change the angle to lie within $\theta = [0, \frac{\pi}{3}]$, and change the x-value to start at 0 (Figure 22). By limiting the range of θ, we reduce the state space needed for sampling while still producing a useful model. We assume the spokes not in contact with the ground do not greatly affect the dynamics of the system.

To find the three spokes closest to the ground, we ignore the actuator states φ_i. In other words, the positions we evaluate are not the actuator tips: $r_i = r_0 + (l + l_0 + \varphi_i)\,\hat{e}_i$, but rather the spoke tips: $r_{i\_sp} = r_0 + (l + l_0)\,\hat{e}_i$.

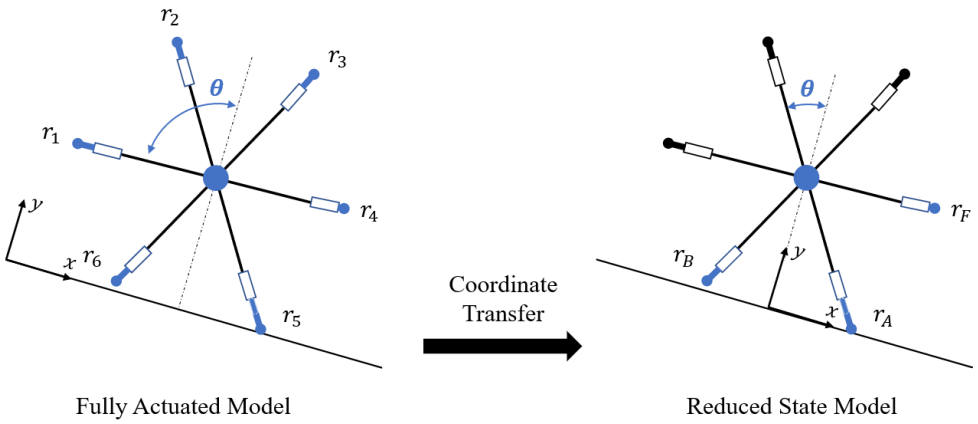


Figure 22. Coordinate Transformation between Fully Actuated and Reduced State Models. Changes include remapping the angle to fit within $\theta = [0, \frac{\pi}{3}]$, setting the x-coordinate of the center of mass to zero, and making the spokes furthest from the ground passive (shown here in black)

Below is the pseudo code for how the coordinate transformation is implemented:

control_law(z_t = [x, y, $\theta, \varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6, \dot{x}, \dot{y}, \dot{\theta}$]):

θ_{new} = mod($\theta, \frac{\pi}{3}$)

if $\theta_{new} == \frac{\pi}{6}$: // breaks ties when the rimless wheel is in stance

if $\dot{\theta} > 0$:

[n1, n2, n3] = return_lowest_three_spokes(z_t =[x, y, $\theta + 0.01, \dot{y}, \dot{\theta}$])

// where [n1, n2, n3] indicates spoke_tip_{n1} is closest to the ground, spoke_tip_{n2} is second closest, etc. Note these are calculated without ϕ_i

else:

[n1, n2, n3] = return_lowest_three_spokes(z_t =[x, y, $\theta - 0.01$, $\dot{y}$, $\dot{\theta}$])

else:

[n1, n2, n3] = return_lowest_three_spokes(z_t =[x, y, θ, $\dot{y}$, $\dot{\theta}$])

$\phi_A = \phi_{n1}$ //the lowest spoke is assigned as spoke A

if n1 == 6: //if the lowest spoke is 6, then the clockwise order of spokes is [5, 6, 1]

$\phi_B = \phi_1$

$\phi_F = \phi_5$

elif n1 == 1: //if the lowest spoke is 1, then the clockwise order of spokes is [6, 1, 2]

$\phi_B = \phi_2$

$\phi_F = \phi_6$

else: //if the lowest spoke is neither 1 or 6, then the clockwise order of spokes is [min(n2, n3), n1, max(n2, n3)]

$\phi_B = \phi_{\max(n2,n3)}$

$\phi_F = \phi_{\min(n2,n3)}$

$z_0 = [0, y, \theta_{new}, \phi_A, \phi_B, \phi_F, \dot{x}, \dot{y}, \dot{\theta}]$ //new state vector with only three actuators

5.4 Actuated Koopman Models (RBF, NN)

A dataset was generated from the three-actuator Rimless Wheel simulation for the actuated Koopman model. Just as before, a region of space centered around the origin was randomly sampled. To ensure that floor contact was properly modeled, at least 20% of every trajectory started with ground contact. The trajectories were capped with maximum initial energy (including the potential energy stored in the viscoelastic springs due to ground penetration), and trajectories that deviated too far from the origin were removed.

The data sets used to generate the Koopman models consisted of *purely passive data*, that is $u_n = 0$ for all time t. The actuator states were randomly initialized with $\phi_n \in [-0.1, 0.1]$ and held constant throughout the trajectory. Each trajectory lasted a total of two seconds with data sampled at a rate of dt = 0.05 seconds. Trajectories were ended prematurely if any spokes other than A, B or F touched the ground. Data sets were compiled from 100 and 1000 trajectories.

A validation data set was created of 100 trajectories created in the same manner as the training sets. A second, actuated validation data set was created of 2,000, trajectories with constant input $u_n = C$ and a duration $t_f = 0.8$ seconds. In this section, the actuators had a maximum extension of $|\phi_n| \leq 0.1\ m$. Due to the randomized actuator states and the constant control input, many trajectories lasted only one- or two-time steps before reaching the actuator state limit. Data collection was ended prematurely before the actuators could exceed this limit.

Four RBF Koopman models were generated in the same manner as in Section 3.2. The training set was truncated to remove actuator velocities, so the states used for these Koopman model were $z = [x, y, \theta, \phi_A, \phi_B, \phi_F, \dot{x}, \dot{y}, \dot{\theta}]$. This was done to prevent the model from estimating

the control inputs ($\dot{\phi}_i = u_i$). Some models were trained on similarly generated datasets but without the x-position data. This practice is further discussed in Section 5.5.

As before, the RBF centers are calculated via kmeans++ and the A matrix obtained via least squares estimation. In addition, two models were made with Neural Net generated observables with the same training data sets. All hyperparameters were kept the same as before. The six models were evaluated for the passive dataset in Table 4. The best performing model was the Koopman Model with 1000 RBF, trained on the 1000-trajectory data set, but the neural net models also had low MSE, especially for the 1-time step prediction.

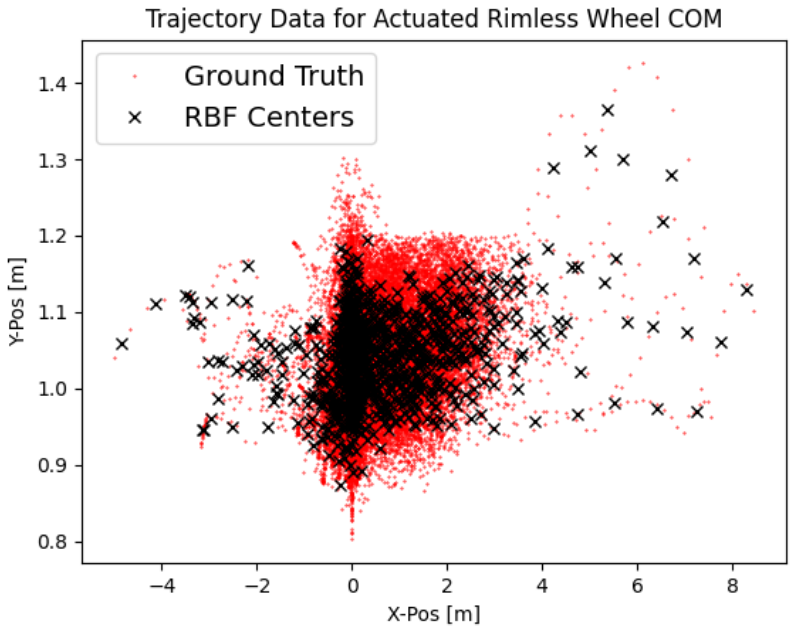


Figure 23 Placement of RBF Centers Using kmeans++ for the Actuated Rimless Wheel Model. Depicted are 1000 RBF centers based on the 1000-trajectory data set. The Rimless Wheel COM positions are shown in red while the RBF centers are shown in black. The RBF centers are also functions of wheel angle (θ), actuator states (ϕ_A, ϕ_B, ϕ_C), and velocities ($\dot{x}, \dot{y}, \dot{\theta}$), but these dimensions are not plotted for simplicity. While this model has actuators, the training data sets are entirely passive ($u_n = 0$).

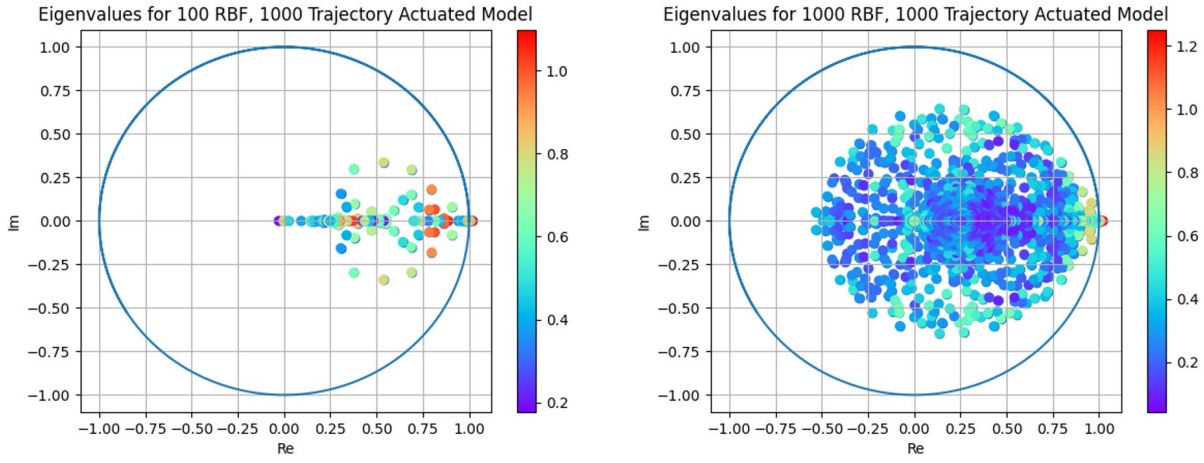


Figure 24. Mode Power Plots for 100, and 1000 RBF Actuated Koopman Models. For 100 RBF, the poles are mostly aligned along the x-axis. Both models use the 1000-trajectory training set.

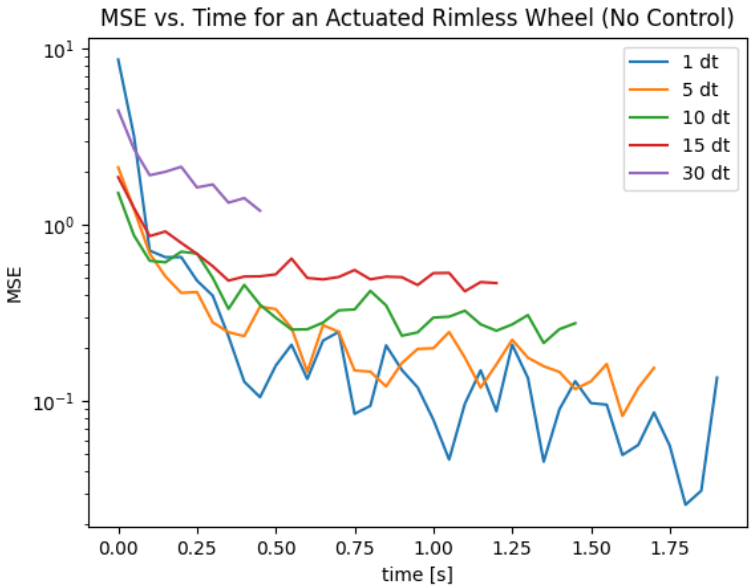


Figure 25. Mean Squared Error vs. Time for an Actuated Koopman Model for Passive Trajectories. MSE calculated for 2,000 trajectories not included in training/model creation. Once again, there is seemingly a downwards trend between trajectory time and prediction accuracy. The Koopman model was created with the 1000-trajectory training set and 1000 RBFs.

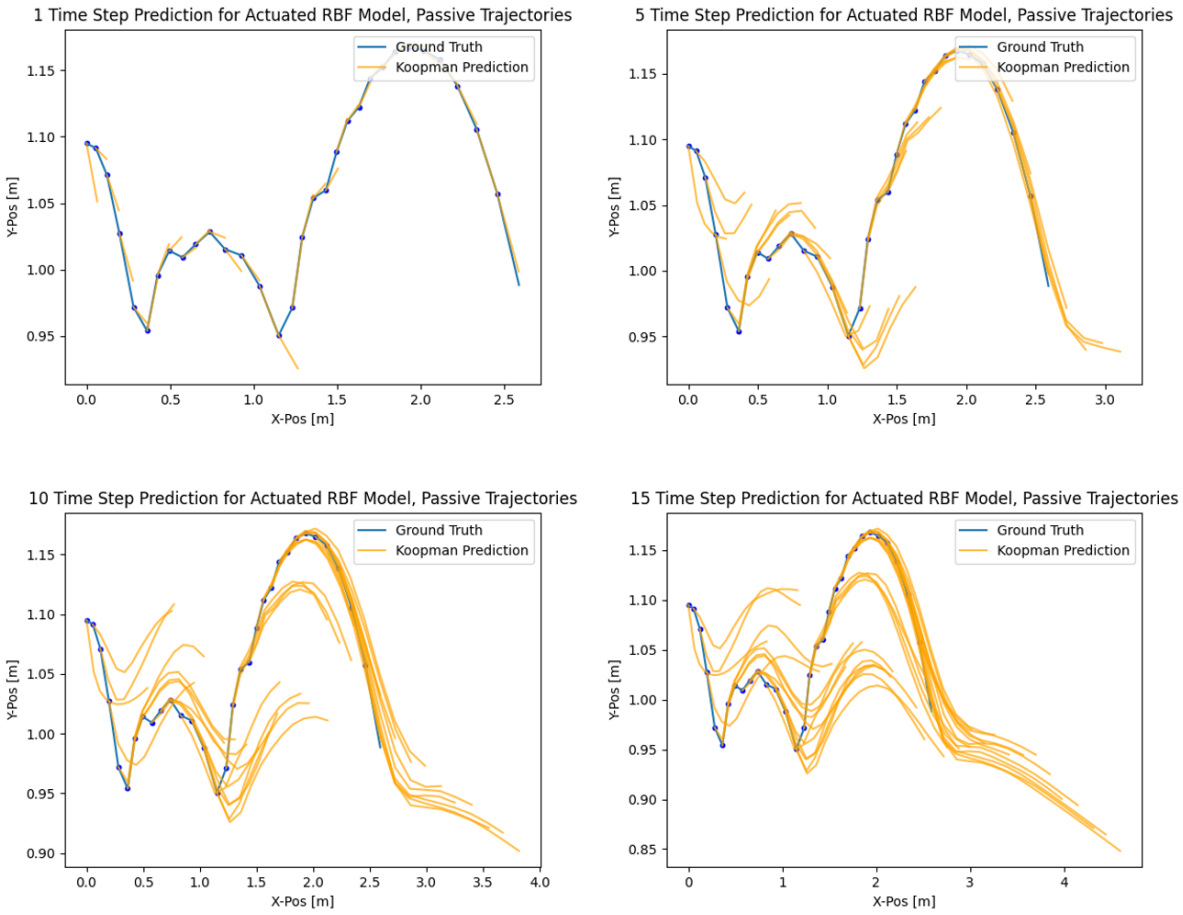


Figure 26. Actuated Koopman Model Tracking Performance for a Passive Trajectory and for Different Time Horizons. At every time step, the model is given the ground truth (depicted by the blue dot) and then blindly predicts the next N time-step (depicted by the orange line going off each blue dot). The Koopman model was created with the 1000-trajectory training set and 1000 RBFs.

			1 dt	5 dt	10 dt	15 dt	30 dt
		100	0.52	0.51	0.72	1.01	2.92
	100	1000	0.62	0.41	0.68	0.82	2.88
		100	0.53	0.50	0.69	0.93	2.52
RBF	1000	1000	0.48	0.31	0.42	0.64	2.05
	100	30	0.49	1.09	3.35	1.68×10^{1}	3.62×10^{3}
NN	1000	30	0.38	0.58	0.92	1.40	5.35

Table 4. Comparison of Different Actuated Koopman Models for Passive Trajectories. The leftmost column describes the type of observables used in the model, followed by the number of trajectories in the data set, and finally the number of observables. The MSE is evaluated for a 1-, 5-, 10-, 15-, and 30-time step prediction over the course of 100 passive trajectories. The RBF Koopman model with the 1000 trajectory data set performed the best. The Neural Net models had low 1-time step predictions but very high MSE for longer time horizons.

Following the evaluation of the Actuated Rimless Wheel Koopman models on the passive dataset, the models were tested on an actuated dataset. As previously mentioned, the actuated dataset was derived from simulations of the actuated rimless wheel with a constant control input ($u_n = C_n$) and time steps of $dt = 0.001$ seconds. In contrast, the Koopman model was generated using data sampled at a rate of $dt = 0.05$ seconds and is therefore only capable of predictions of that scale.

Recall the Koopman models with only have three actuated spokes. With the control coherent Koopman formulation, the control input only affects actuator velocities.

$$\phi_{i,t+1} = \phi_{i,t} + \mathrm{dt}\, u_{i,t}$$

Thus, the B matrix is effectively empty, except for the dt term in the rows corresponding to the actuator states.

When provided with a control input, the Koopman model tends to overestimate the ground reaction forces at the actuator tip, resulting in higher MSE (See Figure 27 and Table 5). This discrepancy arises because the simulation, which updates more frequently, calculates ground reaction forces before the actuator tip penetrates too deeply into the ground, allowing for some degree of repulsion. On the other hand, the Koopman model, updating at a slower rate, pushes the actuator tip further into the ground, resulting in higher estimated ground reaction forces.

Since the Koopman model is trained on passive data, it lacks information about actuators pushing off the ground, and therefore cannot correct the overestimation. An alternative approach could involve updating the Koopman model at the same rate as the simulation. However, considering that Koopman models generally struggle with long-term prediction accuracy, this solution would limit the model's overall utility.

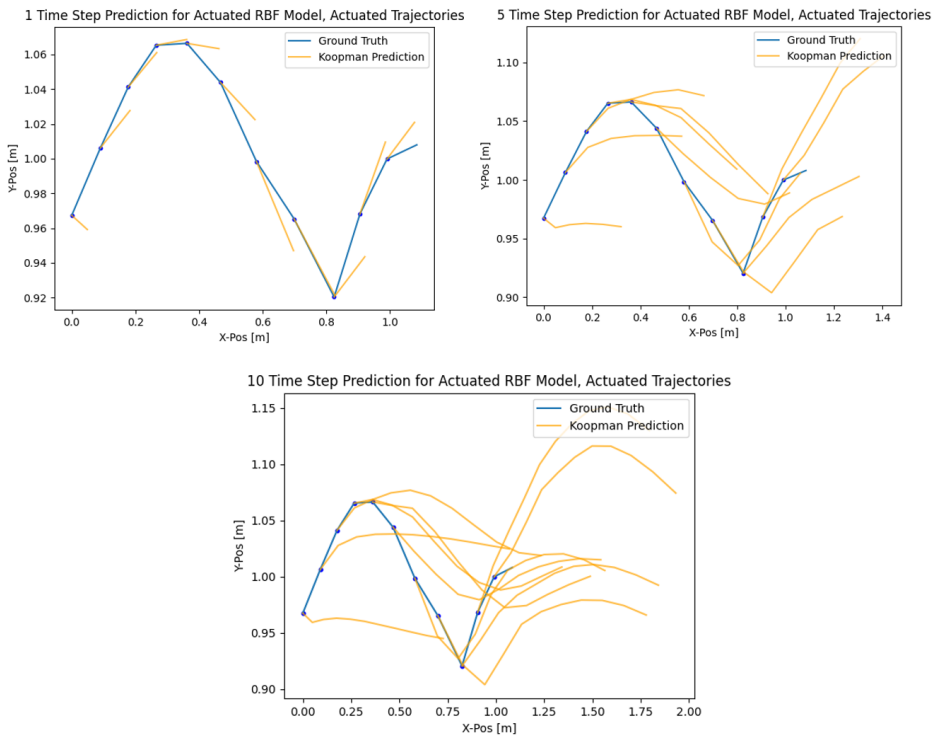


Figure 27. Actuated Koopman Model Tracking Performance for an Actuated Trajectory and for Different Time Horizons. Plots generated using the same model as those in *Figure 23*. The control input is kept constant throughout the trajectory. The tracking is worse for the actuated trajectories, but the model is able to predict the general shape of the trajectory.

			1 dt	**5 dt**	**10 dt**	**15 dt**
		100	1.01	0.71	0.68	0.89
	100	1000	1.35	0.88	2.13	1.43
		100	8.30×10^2	0.67	0.63	0.79
RBF	1000	1000	0.97	0.62	0.56	1.03
	100	30	1.04	0.78	0.83	9.42
NN	1000	30	0.81	0.79	1.10	1.85

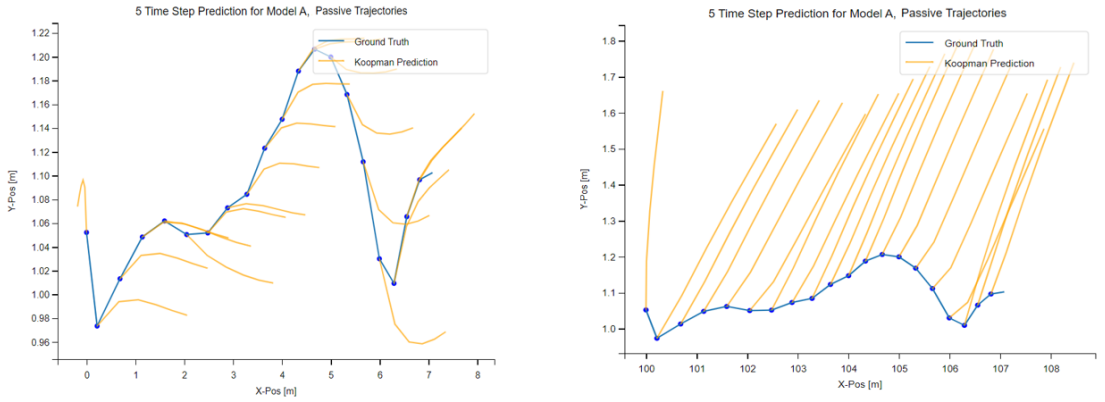
Table 5. Comparison of Different Actuated Koopman Models for Actuated Trajectories. The leftmost column describes the type of observables used in the model, followed by the number of trajectories in the data set, and finally the number of observables. The MSE is evaluated for a 1-, 5-, 10-, 15-time step prediction over the course of 100 passive trajectories. The relative performances are similar to that of Table 4, with the same general trends. The MSE is on average higher for the actuated trajectories, perhaps due to the time-scale discrepancy between the simulation data and Koopman model.

5.5 Impact of X-Position Data on Model Fidelity

In this section, we will investigate how incorporating x-position data in the creation of the Koopman models effects their performance both near and away from the origin. As mentioned in Section 3.3, the rimless wheel dynamics are invariant of the location of the origin on the x-axis. That is, the rimless wheel will behave the same whether it is far up or far down the slope. Furthermore, when ramp angle $\gamma = 0$, the x-position becomes an ignorable coordinate.

Model A was created with RBFs whose centers were placed along all states (including x) while Model B was created with RBFs centers that do not depend on x. The models in this section were created with $\epsilon = 0.4$, $\gamma = 20°$, 100 observables and a 1000 trajectory data set. In addition, the actuator bounds are increased from $-0.1 \text{ m} \leq \phi_N \leq 0.1 \text{ m}$ to $-0.25 \text{ m} \leq \phi_N \leq 0.25 \text{ m}$. All other model, ground contact, and inertial parameters were kept consistent with the previous sections. Figure 28 shows that Models A and B perform similarly when near the origin, but surprisingly, both perform poorly when $x \approx 100$ m. While Model B's RBFs are independent of the x-position, the least squares regression was done with x-position data, resulting in an inaccurate model. An analysis of Model B's A matrix reveals that the x-position state has the greatest influence on the dynamics of the x-velocity and angular velocity (Figure 29).

A.)



B.)

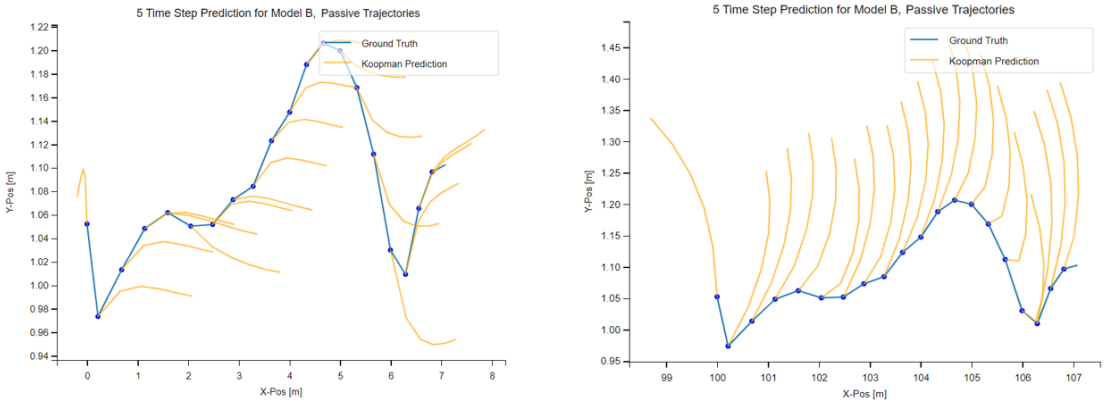


Figure 28. Time Prediction Accuracy Comparison at x = 0 m and x = 100 m. Model A (top row) employs RBF centers that vary with x, while Model B (bottom row) uses RBFs independent of x. The same trajectory was used across all four plots, with only the x-position modified. Both models show poor performance when x = 100 m, suggesting that incorporating x-position data in the least squares regression introduces an x-dependency.

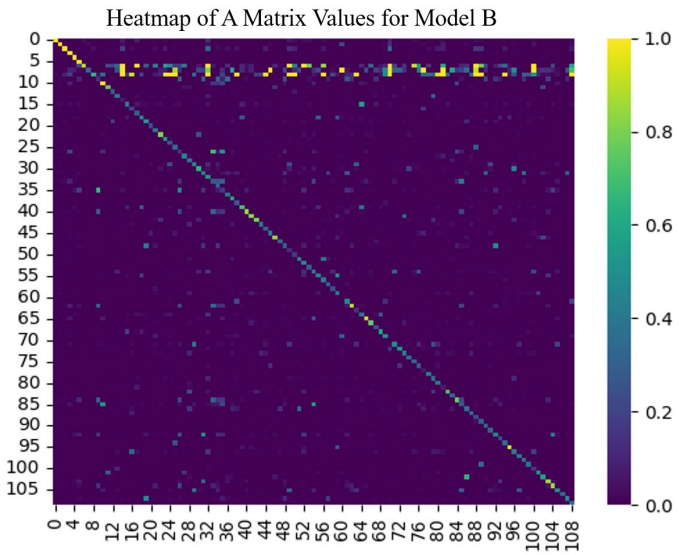


Figure 29. Heatmap of A Matrix Values for Model B. The first column corresponds to the effect of the x-position in the time-evolution of the state variables and observables. The largest value on the first column is at (0,0) = 0.9995, corresponding to the preservation of the x-position. The second largest is at (6, 0) = -0.0195, and the third largest values is at (8,0) = 0.0167 corresponding to the x-velocity and angular-velocity respectively. This indicates that the A matrix incorrectly attributes the x-position as influencing the dynamics of both the x-velocity and angular velocity.

Models B.1 and B.2 were constructed to compare different methodologies for eliminating x-dependency from the A matrix. Model B.1 simply sets all values in the first column of the A matrix to zero, except at (0,0). Model B.2, on the other hand, is generated using the same datasets but modified to exclude the x-position. In this case, $z = [y, \theta, \phi_A, \phi_B, \phi_F, \dot{x}, \dot{y}, \dot{\theta}]$, and $\chi(z) = [z, g_1(z), g_2(z), \ldots, g_N(z)]$. The x-position is obtained by integrating $\dot{x}$ is integrated over the trajectory ($\widehat{x_t} = x_0 + \sum_{i=0}^{t-1} \dot{x}_i * dt$). Table 6 displays the MSE of the Models B, B.1, and B.2 for trajectories at x = 0 m and x = 100 m. The best performing model was Model B.2, the only model fully independent of the x-position. The accuracy of Model B.1 had higher error at x = 100 m then at x = 0 m. This model was initially trained with the x-position included, so simply

zeroing out the first column of the state matrix A, while the model had already learned to compensate for the x-position data, resulted in generally inaccuracies.

	x-pos [m]	1 dt	5 dt	10 dt	15 dt
Model B)	0	0.52	0.54	0.63	0.71
	100	7.62	122.87	313.33	476.32
Model B.1)	0	0.52	0.53	0.62	0.74
	100	0.52	0.59	0.84	1.22
Model B.2)	0	0.52	0.53	0.62	0.72
	100	0.52	0.53	0.62	0.72

Table 6. Model Comparison for Trajectories Centered at x = 0 m and x = 100 m. MSE was calculated using a validation dataset with a shifted x-position. All three models performed similarly near the origin. Model B showed extremely poor tracking performance at x = 100 m. Model B.1's accuracy decreased at x = 100, likely because the model initially learned to estimate dynamics with the x-position included; thus, simply zeroing out the first column still left an inaccurate dynamics model. Furthermore, the x-position dynamics for Model B.1. may also be inaccurate. In contrast, Model B.2's MSE was identical for trajectories centered at x = 0 m and x = 100 m, as it was designed to be fully independent of x-position data, making it the best performer at x = 100 m.

Chapter 6

Control Methods

6.1 L-MPC on a Ramp

Finally, the actuated Koopman models were used for Linear Model Predictive Control on the fully actuated Rimless Wheel simulation. Using the Gurobi Optimizer, a control input was calculated every 0.05 seconds. Since the simulation was running with a 0.001 second time step, the control input was held constant between MPC calculations. The linear control problem was formulated as the following:

$$\min_{\mathbf{u}[.]} \sum_{t=0}^{t_f-1} (\boldsymbol{\chi}[t] - \boldsymbol{\chi}_{\mathbf{ref}})^{\mathbf{T}} \mathbf{Q} (\boldsymbol{\chi}[t] - \boldsymbol{\chi}_{\mathbf{ref}})^{\mathbf{T}} + \mathbf{u}[t]^{\mathbf{T}} \mathbf{R} \mathbf{u}[t])$$

$$\text{Subject to } \boldsymbol{\chi}[t+1] = \mathbf{A}\boldsymbol{\chi}[t] + \mathbf{B}\mathbf{u}[t], \forall\, t \in [0, t_f - 1]$$

$$\boldsymbol{\chi}[0] = \boldsymbol{\chi}_{\mathbf{0}}$$

$$\mathbf{u}_{\mathbf{min}} \leq \mathbf{u}[t] \leq \mathbf{u}_{\mathbf{max}}$$

$$\mathbf{act}_{\mathbf{min}} \leq \boldsymbol{\chi}[t, 3{:}5] \leq \mathbf{act}_{\mathbf{max}}$$

Where both the control input and actuator states $[\phi_A, \phi_B, \phi_C]$ are bounded. We used the lifted states for the positional error and the Koopman A matrix to advance the dynamics between time steps. The time horizon t_f was tuned for the different Koopman models for best results. The Q and R matrices were also varied for each reference point $\boldsymbol{z_{ref}}$. For example, Figure 25 shows the

use of MPC to stop the rimless wheel from rolling down the ramp. To achieve this behavior, we used $\boldsymbol{z_{ref}} = \mathbf{0}$ and diagonal matrix,

$$Q = \begin{bmatrix} Q_x & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & Q_{\phi_n} \end{bmatrix}$$

where diagonal elements associated with the x-position, theta, and actuator states were set to 0, elements associated with y-position and velocities were nonzero, and elements associated with RBF observables were set to 0.1. With this Q matrix, the cost function penalizes nonzero velocities and decreases the relative importance of observable errors compared to state error. This Q matrix also ensures the rimless wheel stays in contact with the ground by penalizing deviations in y-position. Errors in x-position, θ, and actuator states were ignored.

All Q values associated with the observables were given the same small weighting. The RBFs for the reference point are functions of the entire state, but not all these states are relevant. For example, if the goal is to stop the wheel on the ramp, the RBFs are evaluated with reference state $\boldsymbol{z_{ref}} = \mathbf{0}$. Overemphasizing observables errors could inadvertently drive the rimless wheel to $\theta = 0$, when the intended behavior is to decrease velocities $[\dot{x}, \dot{y}, \dot{\theta}] = \mathbf{0}$. However, lowering the weights on observable error limits the usefulness of the observables. A more effective method would be to use trajectory tracking with lifted states, better leveraging the information encoded in the observables (Section 6.2).

Figure 30 and Figure 31 demonstrate cases where MPC is successful in bringing the rimless wheel to a stop on a ramp from a rolling trajectory and initiating motion from a position about to enter stance. However, the MPC was not consistently successful. Even with saturated inputs, there were several initial conditions under which the wheel could not achieve its desired

behavior. For example, when the rimless wheel was fully stopped on the ramp, the actuators could not generate enough force to start a rolling motion. For the ramp angle $\gamma = 20°$, the actuator bounds $-0.1 \leq \phi_N \leq 0.1$ were too restrictive to enable these behaviors. Similarly, the rimless wheel is unable to push itself up the ramp.

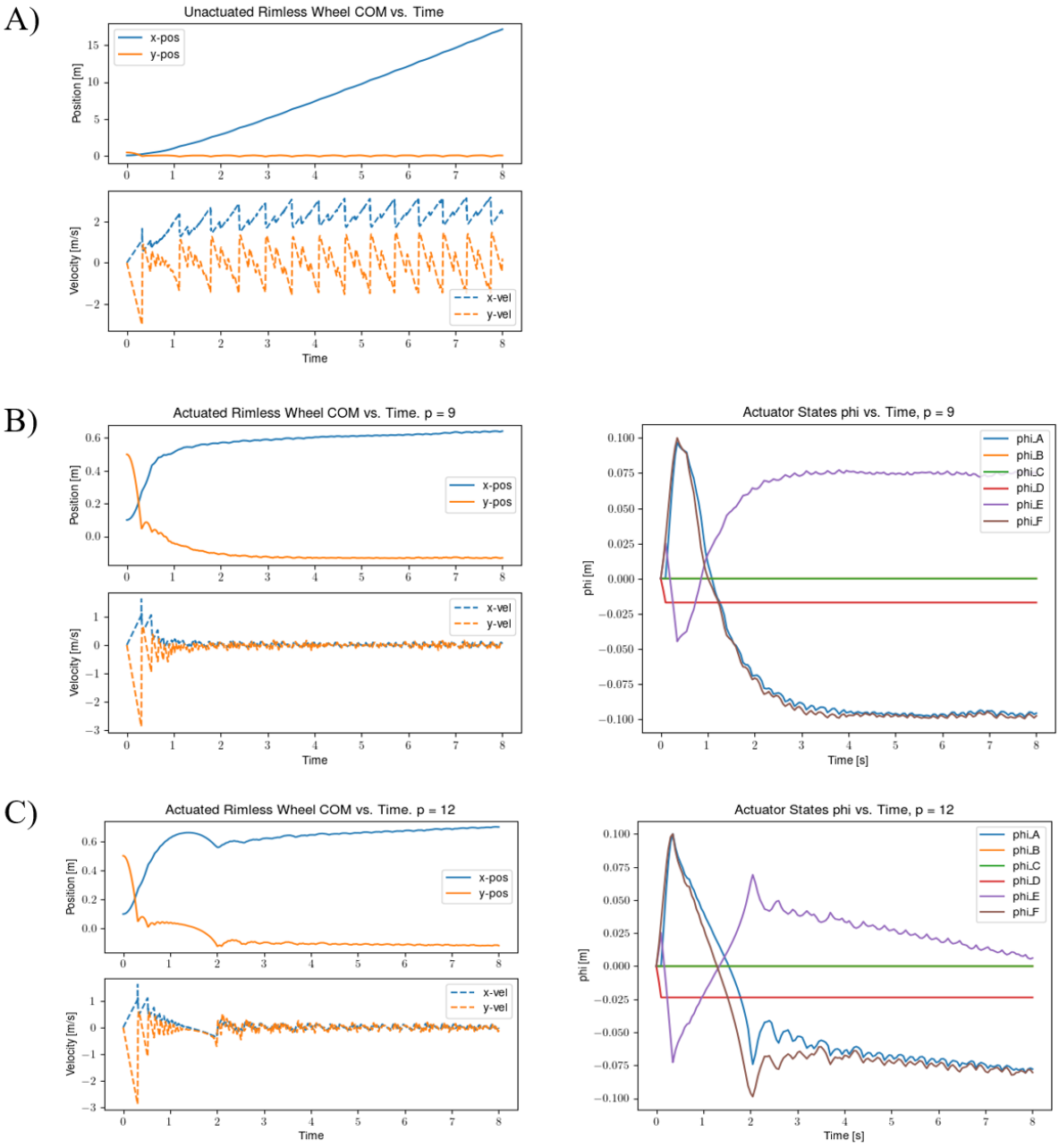
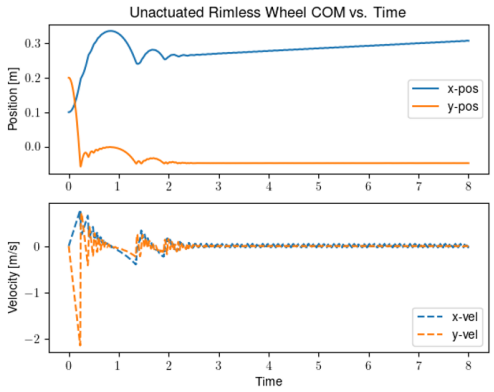
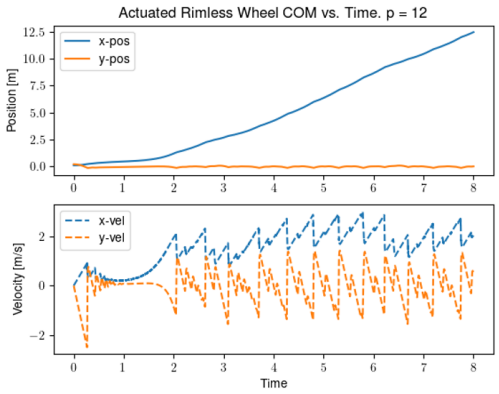


Figure 30. MPC to Stop a Rolling Trajectory. Figure A) shows the trajectory with no control input and initial conditions z_0 = [0.1, 0.5, 2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -5.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]. Figures B) and C) show the wheel is successfully stopped using MPC with the 100 RBF, 1000 trajectory Koopman model, and the 1000 RBF, 1000 trajectory Koopman model. The time horizons (p) for the two models were 9-time steps and 12-time steps, respectively. While both models succeed, the MPC for figure C) took longer to compute due to the larger A matrix requiring significantly more computations.

A)



B)



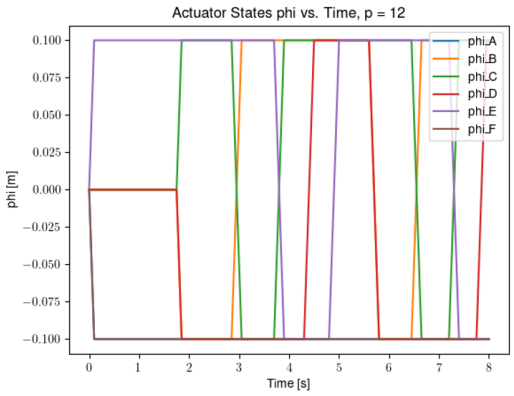


Figure 31 MPC to Start a Stopping Trajectory. Figure A) shows the trajectory with no control input and initial conditions $z_0 = [0.1, 0.2, \frac{\pi}{6} + 0.001, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]$. Figure B) shows the wheel is successfully pushed into a rolling trajectory using MPC with the 100 RBF, 1000 trajectory Koopman model. The positional error was heavily weighted, resulting in fully saturated control inputs.

6.2 L-MPC on Flat Ground

Up to this point, all Koopman models had been created with a ramp angle of $\gamma = 20°$. While constructing Koopman models on ramps was useful for validating prediction accuracy in passive rolling, it is more compelling to study the motion of the rimless wheel when it is entirely driven by the actuators.

There are three behaviors that a rimless wheel can achieve on flat ground: rolling forwards, rolling backwards, and settling into a stance. As before, the stance behavior can be implemented by tracking a zero-velocity reference point. However, when attempting to get the rimless wheel to move forwards or backwards, we found better results when tracking a reference trajectory. A well-designed reference trajectory allows for the incorporation of additional information through the observables, potentially enhancing performance. Trajectory tracking MPC was implemented with the following formulation:

$$\min_{\mathbf{u}[.]} \sum_{t=0}^{t_f-1} (\boldsymbol{\chi}[t] - \boldsymbol{\chi}_{\mathbf{ref}}[t])^{\mathbf{T}} \mathbf{Q} (\boldsymbol{\chi}[t] - \boldsymbol{\chi}_{\mathbf{ref}}[t])^{\mathbf{T}} + \mathbf{u}[t]^{\mathbf{T}} \mathbf{R} \mathbf{u}[t])$$

$$\text{Subject to } \boldsymbol{\chi}[t+1] = \mathbf{A}\boldsymbol{\chi}[t] + \mathbf{B}\mathbf{u}[t], \forall\, t \in [0, t_f - 1]$$

$$\boldsymbol{\chi}[0] = \boldsymbol{\chi}_{\mathbf{0}}$$

$$\mathbf{u}_{\mathbf{min}} \leq \mathbf{u}[t] \leq \mathbf{u}_{\mathbf{max}}$$

$$\mathbf{act}_{\mathbf{min}} \leq \boldsymbol{\chi}[t, 3{:}5] \leq \mathbf{act}_{\mathbf{max}}$$

The reference trajectory was generated using data from passive rolling simulations of the rimless wheel. A key challenge when using a reference trajectory is determining the appropriate segment to track at any given moment, or choosing to follow a set trajectory regardless of the rimless wheel's current state. In this implementation, the point on the reference trajectory closest to the current state, in terms of θ and $\dot{\theta}$, is identified. The next p time steps from this "closest point" form the reference trajectory, where p denotes the time horizon.

The Q matrix for trajectory tracking heavily penalizes deviations in x-velocity and x-position. There are also penalties, though less heavily weighted, for errors in $y, \theta, \dot{y}$, and $\dot{\theta}$. A smaller penalty is assigned to errors in the observables g_i and no penalty is imposed for actuator state errors.

The models presented in the following sections were generated without x-position data (see Section 5.5) and datasets were generated to capture more collisions. Specifically, data was generated with at least 20% of all trajectories starting with ground contact and at least 20% starting near a contact mode change (i.e. initial angles near $\theta = \frac{\pi}{3}, -\frac{\pi}{3}$).

# observables	ϵ	1 dt	5 dt	10 dt	15 dt
100	0.9	0.51	0.37	0.49	0.53
500	1.0	0.49	0.25	0.31	0.40
1000	0.8	0.57	0.20	0.25	0.35

Table 7. Comparison of Flat, DMD + CCK Models for Passive Trajectories. MSE calculated using the same validation dataset for all models. Epsilon was tuned to minimize MSE for each model. Models with a greater number of observables generally produced more accurate predictions, especially for longer time horizons.

Three Koopman models were generated for this section, with their relative performances enumerated in Table 7. However, with the current formulation, the 1000-observable MPC computations regularly take more than 100 times longer than the 100-observable MPC, without a significant improvement in performance. In Korda and Mezic (2018), the method described in the appendix transforms the higher order linear MPC problem into a dense form, thereby eliminating the dependence on the lifting dimension [31]. In the future we aim to implement this formulation and close the gap between the computation loads of the three models, enabling the use of the 1000-observable model in online MPC and reducing computation time across the board. Finally, dimension reduction via SVD is commonly employed in DMD-based Koopman models [8].

Figure 32 depicts the mechanism by which the rimless wheel achieves rolling on flat terrain. The MPC aggressively commands the leg that is about to make contact with the ground to retract (see the orange spoke retracting at t = 4.25 and 4.50 s). Simultaneously, the MPC extends the stance leg (see the blue spoke extend from t = 4.25 to 4.75 s). The extension effort is maximized when the leg is positioned behind the wheel's COM, which pushes the wheel off the back spoke, injecting energy into the system without impeding the rolling motion. This combination of a lengthened back spoke and a shortened front spoke creates a pseudo-ramp effect, encouraging the wheel to roll forward. The rimless wheel's x and y velocities experience sharp changes when the wheel lands on a new spoke. Note that the wheel will continue to extend the back leg beyond contact, indicating potential inaccuracies in the model.

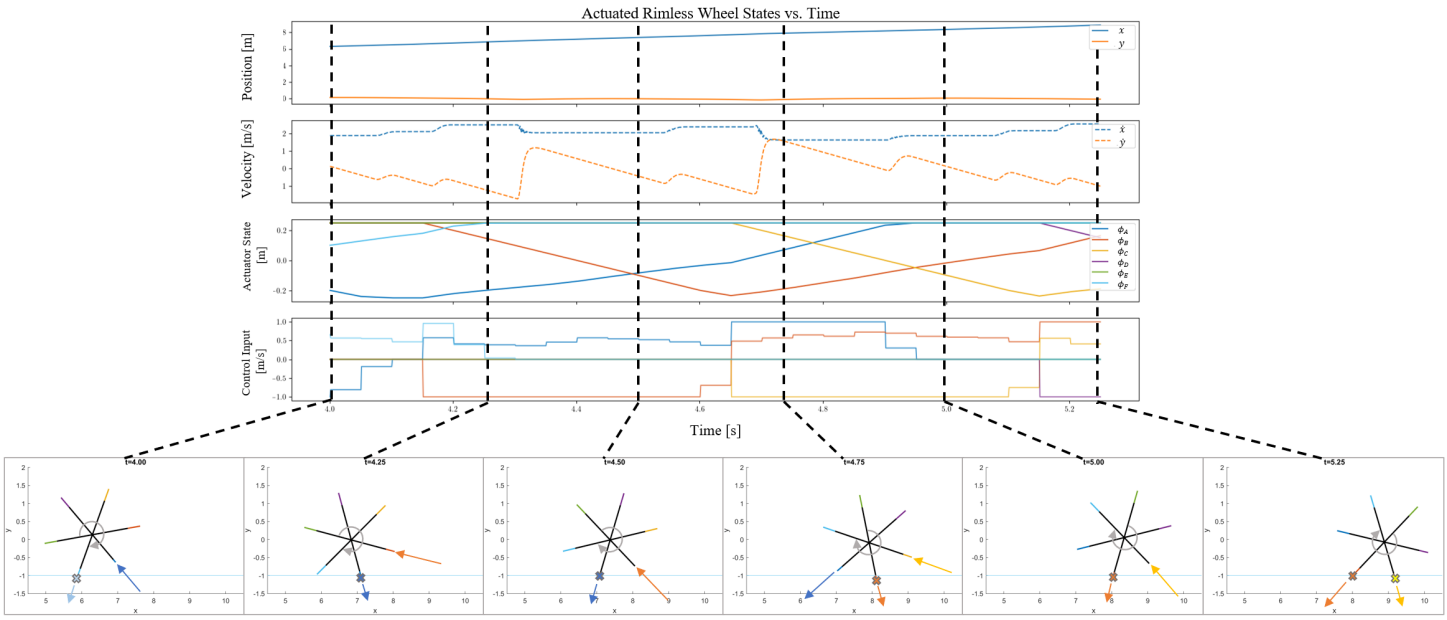


Figure 32. Mechanism of Forward Rolling in the Rimless Wheel. Snapshots are taken every 0.25 seconds, starting at t = 4.0 seconds. In the first snapshot, the wheel maintains ground contact with the light blue spoke (marked by an x), while the control command retracts the dark blue spoke and extends the light blue one (control inputs are represented by colored arrows, with the arrow's length denoting magnitude). The wheel is rolling clockwise, as indicated by the gray circular arrow. The 100 RBF Koopman model was utilized for this figure, tracking the $\gamma = 20°$ passive rolling trajectory.

Figure 33 demonstrates the underlying logic behind the Linear MPC optimization. The MPC rollout shows optimization through multiple instances of the spokes making and breaking contact. This demonstrates the power of the Koopman operator in successfully encoding these changes in contact modes within a linear model, and the control coherent Koopman formulation in generating a B matrix that is accurate through these different contact modes.

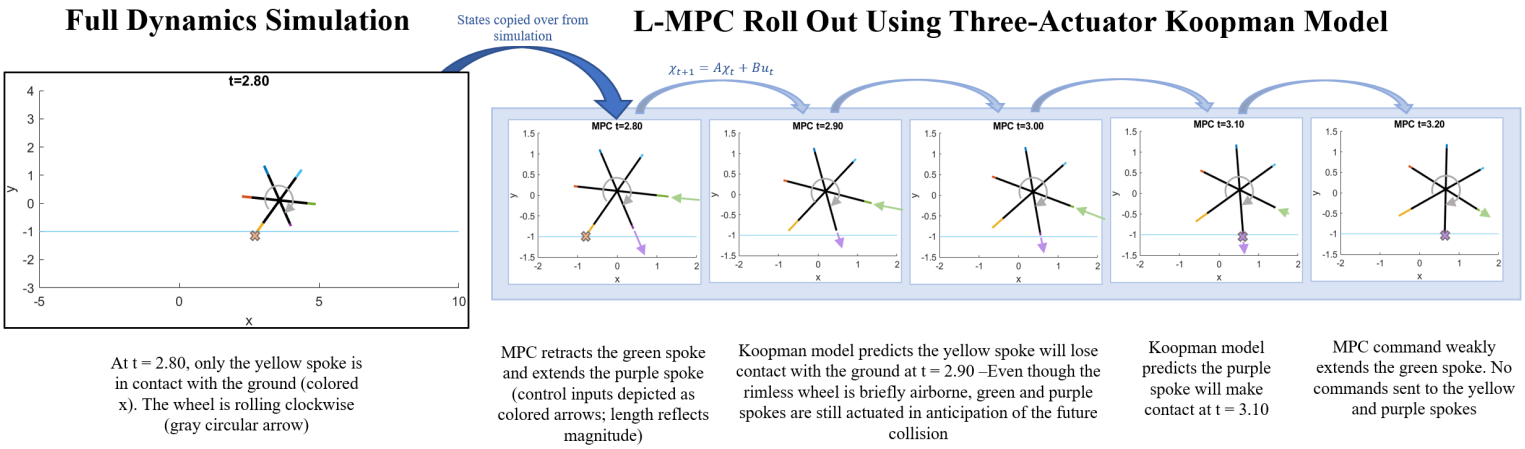


Figure 33. The MPC Rollout using the Three-Actuator Koopman Model. The state at t = 2.80 seconds is copied over from the simulation, but modified to fit the Three Actuator Model (Section 5.3). Future states are predicted using the linear model with control inputs. The Koopman model predicts multiple contact mode changes within the 8 time-step (0.4 second) period. Although the model does not anticipate ground contact with the green spoke, it retracts the green spoke, potentially indicating a learned association between shorter front spokes and forward motion. MPC snapshots are taken two time steps apart, with dt = 0.05 seconds.

To achieve rolling at different speeds, reference trajectories were generated using data from various ramp slopes. Table 8 presents the different rolling speeds achieved with MPC using these reference trajectories. To roll backwards, the same trajectories are used but with the x and the θ values negated.

Reference Ramp Angle [°]	Reference Avg. Velocity [m/s]	Forward Sim Avg. Velocity [m/s]	Backward Sim Avg. Velocity [m/s]
10	1.0	0.96	-0.49
15	1.6	1.42	-0.01
20	2.4	1.49	-0.59
23	3.0	1.44	-0.61

Table 8. Rolling Velocities Achieved in Simulation. Reference trajectories were generated from passive rolling data of the fully actuated rimless wheel at various slope angles. The MPC struggled to increase the average speed beyond 1.5 m/s when rolling forwards, likely due to actuation limitations and the less aggressive nature of contour trajectory tracking. The model was also less successful in achieving high velocities when rolling backwards. Simulation were run for 10 seconds using the 100-observable model with a 12-step time horizon (0.6 seconds) and aggressive inputs (**R** = 0.1 **I**).

The rimless wheel struggled to surpass a forward speed of 1.5 m/s. This limitation is likely due to constraints on the control input ($|u| \leq 1$ m/s) and actuator states ($|\phi_i| \leq 0.25$ m), which restrict the maximum force the rimless wheel can exert to push off the ground and limit the length differential between spokes respectively. This length differential is crucial for simulating the ramp effect, so the limit on actuator states limits the maximum pseudo-ramp angle.

The wheel is less successful in rolling backwards, a surprising result given that there should not be a forwards/backwards bias in the formulation of the model. Figure 34 shows the states and control commands when tracking the $\gamma = 20°$ trajectory both forwards and backward.

In the forward trajectory, we observe that during actuation, adjacent spokes receive opposite inputs—specifically, the middle spoke extends while the front spoke retracts. This action facilitates both a push-off from the ground and creates a pseudo-ramp effect, aiding in forward motion.

In the backward trajectory, however, the front and back spokes are activated. Since the wheel is moving backward, the back spoke retracts while the front spoke extends. Crucially, because the front spoke extends instead of the middle spoke, the overall movement is slower, missing out on the earlier push-off potential that contributes to more efficient forward motion. This behavior remained consistent with different **R** values and time horizon.

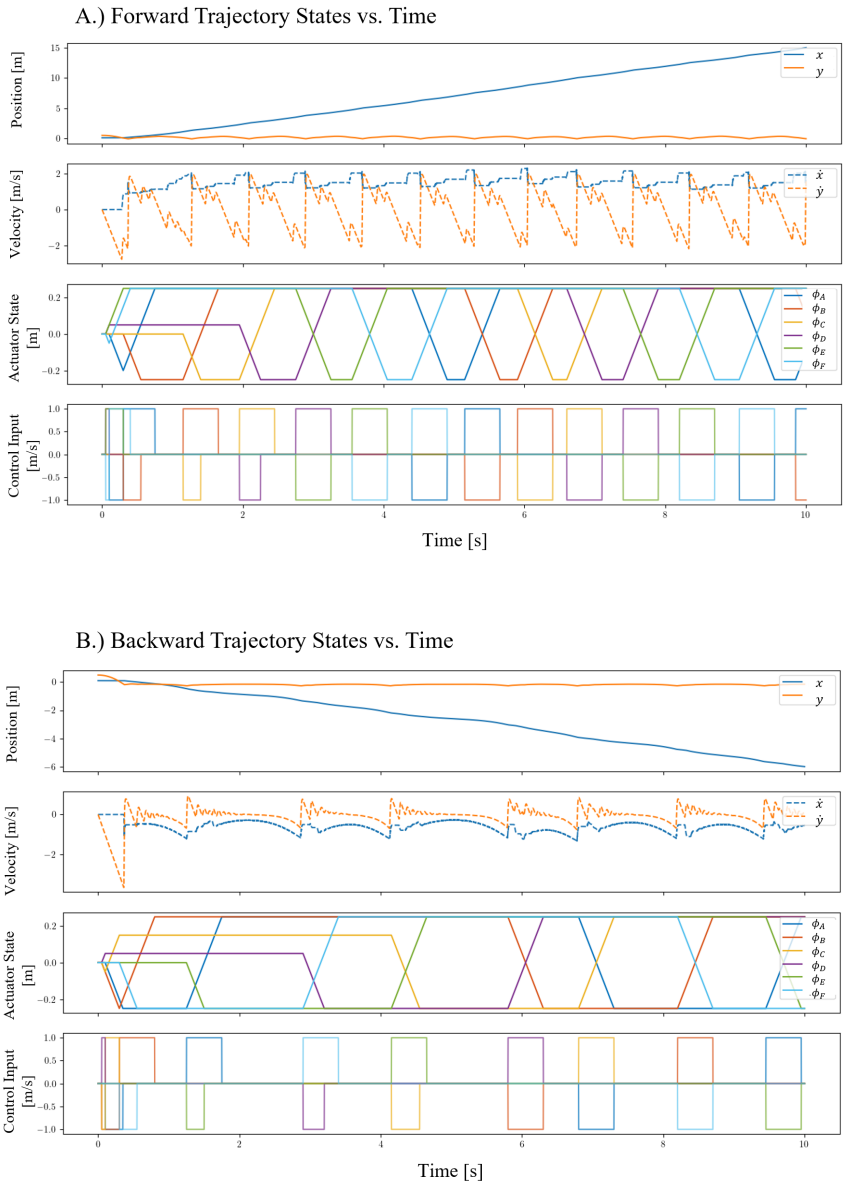


Figure 34. Comparison of Forward and Backward Rolling States and Control. Both figures generated from symmetric initial conditions, and the same weights and reference trajectory.

Finally, there are several ways to fine-tune trajectory tracking. The relative weights of **Q** and **R** can significantly impact performance. Figure 35 illustrates the effects of varying R when tracking a forward rolling trajectory. When **R** is small, the penalty on control inputs is minimal, leading to an aggressive strategy with purely saturated inputs. As **R** increases, control input saturation decreases. In fact, the wheel rolls further with **R** = 0.4 **I**, better tracking the reference trajectory. However, when **R** becomes too large, performance declines, and the wheel takes longer to reach a stable rolling pattern. The total control effort was evaluated over the course of the simulation ($U_{tot} = \sum_{t=0}^{T} ||\mathbf{u_t}||_2 * dt$), with the total effort per rotation for the three simulations as 5.3, 5.7, and 4.8, respectively.

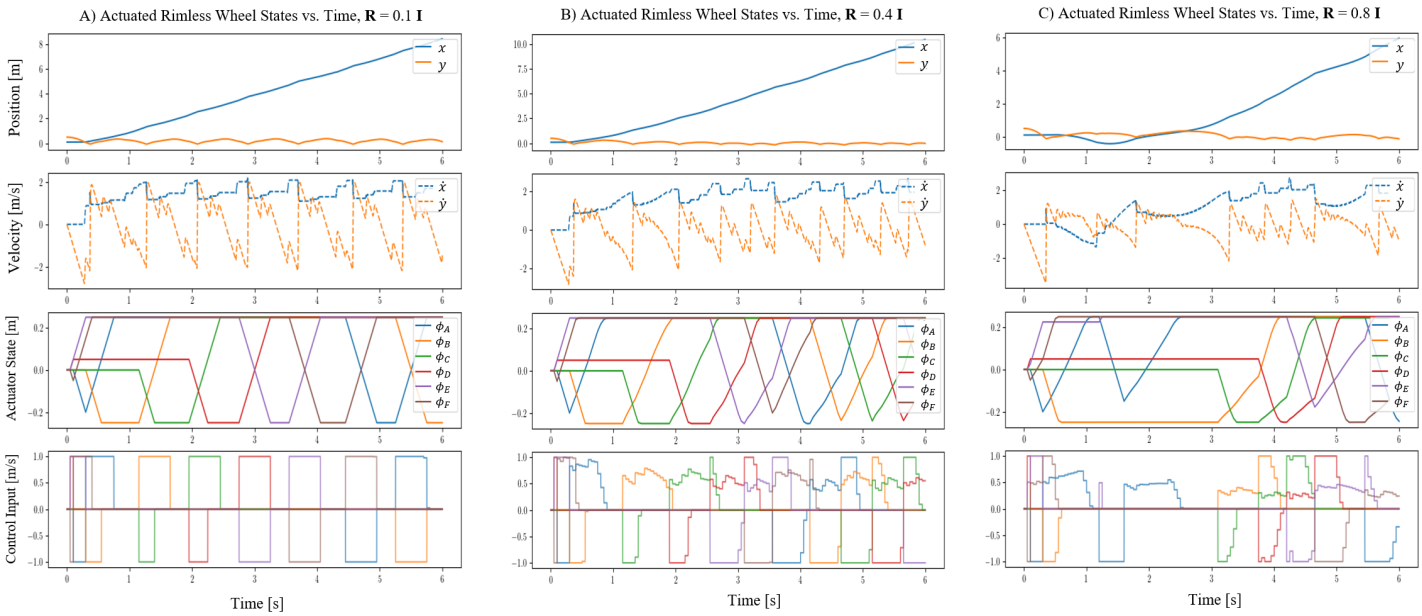


Figure 35. Effect of Input Weight on Rolling Trajectory. Each plot shows the rimless wheel tracking the same 20° trajectory from the same initial condition. In Plot A, where **R** = 0.1**I**, inputs are lightly penalized, leading to fully saturated control inputs. Plot B, with a higher input penalty (**R** = 0.4 **I**), shows less saturated control and more irregularity between steps, yet it traveled the farthest and best tracked the reference trajectory. Plot C, with an input penalty 8 times that of Plot A, traveled the shortest distance. The total command input per rotation for the three simulations was 5.3, 5.7, and 4.8, respectively.

6.3 DMDc Comparison

In the Dynamic Mode Decomposition with control (DMDc) methodology, the B matrix is approximated using least squares regression. Unlike for the Control Coherent Koopman method, generating the B matrix requires actuated datasets. The dataset for this model was generated with the same simulation parameters as those in Section 6.2, but with randomized control commands (within the actuation bounds) that remained constant throughout the trajectory. Due to the constraint that trajectories are terminated once actuator states exceed their limits, many trajectories in this dataset were cut short. To mitigate this, the dataset includes 5,000 trajectories, which contains approximately the same number of data points as the 1,000 trajectory non-actuated dataset.

As before, the radial basis functions are placed in the state space, so the centers are functions of $z = [y, \theta, \phi_A, \phi_B, \phi_F, \dot{x}, \dot{y}, \dot{\theta}]$. Then, the dataset is lifted such that for states z_t, we obtain $\chi_t = [z_t, \phi_1(z_t), \phi_2(z_t), \dots, \phi_N(z_t)]$. We also obtain the lifted dataset for the next time step (χ_{t+1}). Finally, we construct $\Omega_t = [\chi_t, u_t]$. The A and B matrices are obtained using a least squares regression of the Ω_t and χ_{t+1} lifted data sets: $[A, B] = \chi_{t+1}\ {\Omega_t}^{\#}$ [8].

DMDc offers a significant advantage over Control Coherent Koopman by incorporating actuated data into the model formulation, potentially improving the model's understanding of floor interactions. As discussed in Section 5.4, the Control Coherent Koopman model tends to overestimate ground reaction forces at the actuator tip due to its slower update rate compared to the simulation. Since it is trained on passive data, it lacks the ability to correct this overestimation. With DMDc, by including actuation data, the models may provide more accurate predictions of actuation effects. However, there are additional states to fit, which leads to

physically incoherent dynamics, similar to how the inclusion of x-position data led to unphysical dynamics for those Koopman models.

The DMDc models were constructed using 100 RBFs placed with kmeans++. After tuning, the optimal ϵ value for the DMDc Koopman model was $\epsilon = 6.0$, which achieved lower MSE for passive trajectory prediction than the 100 RBF DMD model. However, using this $\epsilon = 6.0$ Koopman model in MPC caused numerical issues due to the model's extreme matrix coefficient range. To avoid these issues, we used a DMDc model with $\epsilon = 2.0$ for MPC. While this model's tracking accuracy was relatively worse, it had a more manageable matrix range. These values are summarized in Table 9.

Model Type	ϵ	1 dt	5 dt	10 dt	15 dt	Max A Value	Avg. A Value
DMDc	2.0	0.49	0.39	0.51	0.58	75	0.020
DMDc	6.0	0.48	0.36	0.47	0.52	29,700	0.057
DMD (+ CCK)	0.9	0.51	0.37	0.49	0.53	1,900	0.10

Table 9. Comparison of Flat DMDc Koopman Models for Passive Trajectories. The MSE values are calculated using the same validation dataset for all models. The table also lists maximum and average value for each model's A matrix. Generally, higher epsilon values will result in larger ranges in the A matrix values, as the RBFs can capture the volatility of floor contact.

Both DMDc models correctly captured the actuator dynamics, represented by $\phi_{i,t+1} = \phi_{i,t} + u_{i,t} * \text{dt}$. As a result, the rows in the B matrix corresponding to the actuator states are nearly zero (around 10^{-16}), except for the dt term. However, when implementing MPC, the DMDc models performed very poorly, which was surprising given their decent passive tracking performance.

An examination of the MPC rollout revealed nonsensical predictions (illustrated in Figure 36). The model incorrectly predicts the wheel's future states, so inefficient and oftentimes unhelpful control commands are implemented, impeding the wheel from rolling trajectory. An examination of multiple MPC rollouts shows the DMDc model generally predicts the wheel will roll forward, despite their being no physical basis for this behavior.

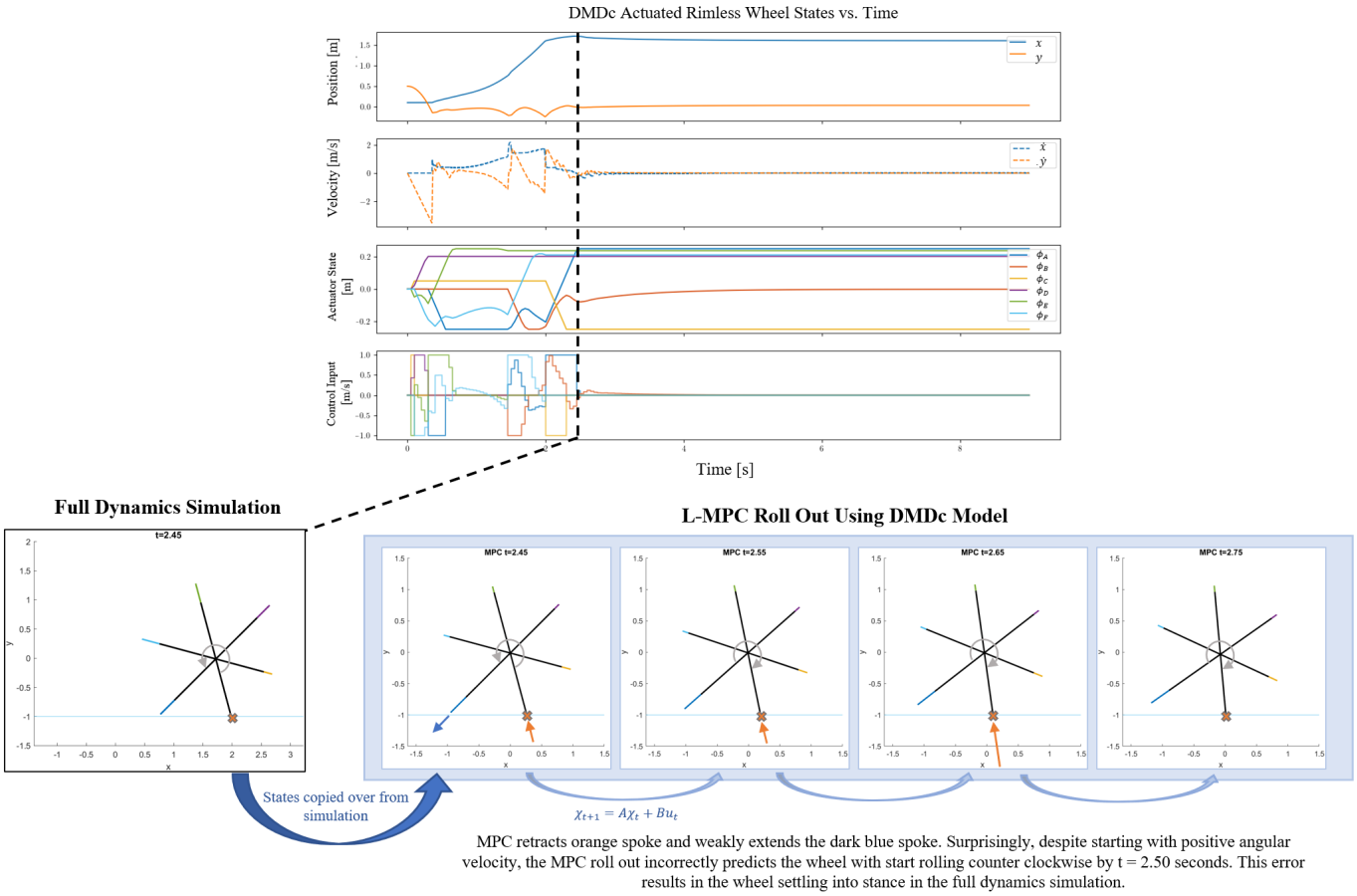


Figure 36. The MPC Rollout using the DMDc, Three Actuator Koopman Model. The state at t = 2.45 seconds is copied over from the simulation, but modified to fit the Three Actuator Model (Section 5.3). Future states are predicted using the linear model with control inputs. Almost immediately, the MPC roll out predicts the rimless wheel will turn clockwise. Due to the incorrect prediction, the wheel settles into a stance for the remainder of the simulation. MPC snapshots are taken two time steps apart, with dt = 0.05 seconds.

Furthermore, replacing the B matrix in the DMDc model with the B matrix from the control-coherent Koopman model also failed to yield success. This is likely because the model had learned to predict trajectories where the actuators influence all the states. This situation is similar to training the model with x-dependency. Even zeroing the values in the A matrix associated with the x-position was insufficient to achieve an accurate model, as the model had been trained to account for the effect of the x-position influencing the dynamics of the other states.

Chapter 7

Conclusion and Future Work

In this thesis, we have detailed the creation of a linear model for an actuated rimless wheel, and evaluated its use for prediction and control. The primary contribution of this work is the linear model's usage in linear model predictive control, and the ability to plan through contact using a single set of linearized dynamics—a powerful technique that enables globally optimal online optimization through impact events. This is an exciting result with the potential to revolutionize control methods for contact-rich dynamical systems.

However, developing a contact-heavy system necessitated careful selection of the model, particularly for this initial implementation. Limitations of the rimless wheel model, as well as the methods discussed in this thesis, are linked to ongoing questions within the Koopman Operator community. To extend this model to legged robots (i.e. higher-order, unstable systems), we must address challenges related to dimensionality and the ability to capture unstable modes within a linear model. Potential future work could involve investigating how to model unstable models without allowing them to dominant, or generating models built solely for specific subspaces.

Currently, we are working on implementing the rimless wheel model in hardware. As we transition to hardware, a key question arises: Will the control-coherent Koopman model and the time-delayed effects of inputs and ground reaction forces be sufficient for estimating real-world dynamics? We anticipate that generating data-driven models based on real hardware data will be necessary. Furthermore, error due to the control time delay effect can be mitigated with shorter

time steps. Images of the initial hardware setups are shown in Figure 37. Challenges in the hardware implementation include improving computation time for both MPC and image processing (for state estimation) to enable 20 Hz control inputs, and the need for actuators that are both fast and capable of generating sufficient force to enable rolling behaviors.

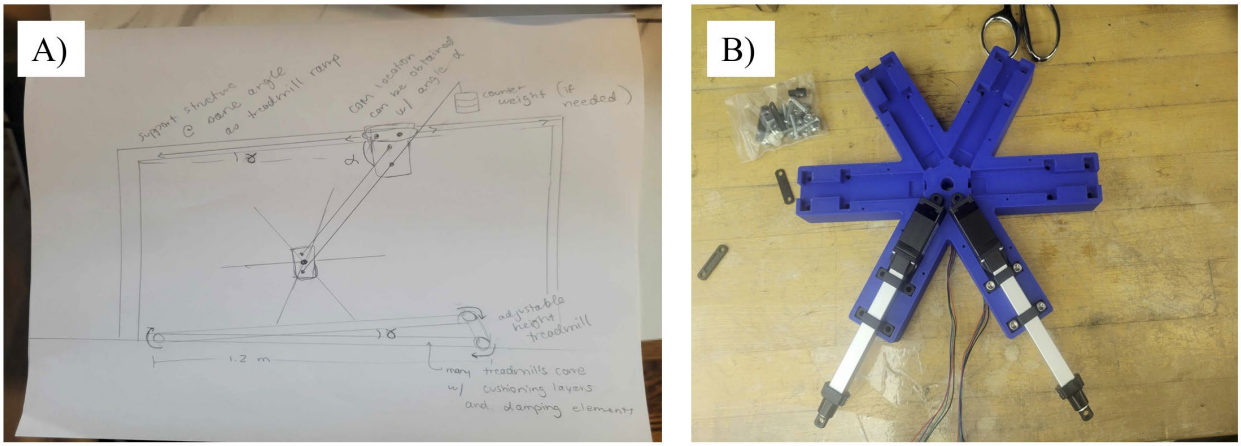


Figure 37. Rimless Wheel Hardware Plans. Figure A) shows the gantry schematic to keep the rimless wheel fixed in-plane. Figure B) shows a prototype of the actuator housing and micro stepper linear actuators

Other areas of interest include utilizing ground reaction forces and other physically meaningful variables as observables, as well as improving Koopman predictions over longer time horizons. Additionally, there is potential to extend the rimless wheel model to more complex motions, such as overcoming uneven terrain, navigating obstacles, or synchronizing motion with another wheel. The application of Koopman operator theory to robot control and contact dynamics is a highly promising area, offering numerous exciting directions for future research.

Bibliography

[1] H. Li and P. M. Wensing, "Cafe-Mpc: A Cascaded-Fidelity Model Predictive Control Framework with Tuning-Free Whole-Body Control," Mar. 14, 2024, *arXiv*: arXiv:2403.03995. Accessed: May 27, 2024. [Online]. Available: http://arxiv.org/abs/2403.03995

[2] Y.-M. Chen, J. Hu, and M. Posa, "Beyond Inverted Pendulums: Task-Optimal Simple Models of Legged Locomotion," *IEEE Trans. Robot.*, vol. 40, pp. 2582–2601, 2024, doi: 10.1109/TRO.2024.3386390.

[3] P. M. Wensing, M. Posa, Y. Hu, A. Escande, N. Mansard, and A. Del Prete, "Optimization-Based Control for Dynamic Legged Robots," Nov. 21, 2022, *arXiv*: arXiv:2211.11644. doi: 10.48550/arXiv.2211.11644.

[4] J. Koenemann *et al.*, "Whole-body model-predictive control applied to the HRP-2 humanoid," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, Germany: IEEE, Sep. 2015, pp. 3346–3351. doi: 10.1109/IROS.2015.7353843.

[5] J. Reher and A. D. Ames, "Dynamic Walking: Toward Agile and Efficient Bipedal Robots," *Annu. Rev. Control Robot. Auton. Syst.*, vol. 4, no. 1, pp. 535–572, May 2021, doi: 10.1146/annurev-control-071020-045021.

[6] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley, "A Data-Driven Approximation of the Koopman Operator: Extending Dynamic Mode Decomposition," *J. Nonlinear Sci.*, vol. 25, no. 6, pp. 1307–1346, Dec. 2015, doi: 10.1007/s00332-015-9258-5.

[7] H. H. Asada, "Global, Unified Representation of Heterogenous Robot Dynamics Using Composition Operators: A Koopman Direct Encoding Method," *IEEEASME Trans. Mechatron.*, vol. 28, no. 5, pp. 2633–2644, Oct. 2023, doi: 10.1109/TMECH.2023.3253599.

[8] J. L. Proctor, S. L. Brunton, and J. N. Kutz, "Dynamic Mode Decomposition with Control," *SIAM J. Appl. Dyn. Syst.*, vol. 15, no. 1, pp. 142–161, Jan. 2016, doi: 10.1137/15M1013857.

[9] J. Ng and H. Asada, "Model Predictive Control and Transfer Learning of Hybrid Systems Using Lifting Linearization Applied to Cable Suspension Systems," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 682–689, Apr. 2022, doi: 10.1109/LRA.2021.3131750.

[10] D. A. Haggerty *et al.*, “Control of soft robots with inertial dynamics,” *Sci. Robot.*, vol. 8, no. 81, p. eadd6864, Aug. 2023, doi: 10.1126/scirobotics.add6864.

[11] H. H. Asada, “Control-Coherent Koopman Modeling: A Physical Modeling Approach”.

[12] C. O’Neill and H. H. Asada, “Koopman Dynamic Modeling for Global and Unified Representations of Rigid Body Systems Making and Breaking Contact”.

[13] W. Yang and M. Posa, “Impact Invariant Control with Applications to Bipedal Locomotion,” in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2021, pp. 5151–5158. doi: 10.1109/IROS51168.2021.9636094.

[14] Y. Gong and J. W. Grizzle, “Zero Dynamics, Pendulum Models, and Angular Momentum in Feedback Control of Bipedal Locomotion,” *J. Dyn. Syst. Meas. Control*, vol. 144, no. 121006, Oct. 2022, doi: 10.1115/1.4055770.

[15] P. Bevanda, M. Beier, S. Kerz, A. Lederer, S. Sosnowski, and S. Hirche, “Diffeomorphically Learning Stable Koopman Operators,” May 30, 2022, *arXiv*: arXiv:2112.04085. Accessed: May 31, 2024. [Online]. Available: http://arxiv.org/abs/2112.04085

[16] G. Mamakoukas, I. Abraham, and T. D. Murphey, “Learning Stable Models for Prediction and Control,” Mar. 24, 2022, *arXiv*: arXiv:2005.04291. Accessed: May 31, 2024. [Online]. Available: http://arxiv.org/abs/2005.04291

[17] M. Garcia, A. Chatterjee, A. Ruina, and M. Coleman, “The Simplest Walking Model: Stability, Complexity, and Scaling,” *J. Biomech. Eng.*, vol. 120, no. 2, pp. 281–288, Apr. 1998, doi: 10.1115/1.2798313.

[18] V. F. H. Chen, “Passive Dynamic Walking with Knees: A Point Foot Model”.

[19] R. Tedrake, *Underactuated Robotics: Algorithms for Walking, Running, Swimming, Flying, and Manipulation*. Course Notes for MIT 6.832, 2023. Accessed: Aug. 10, 2024. [Online]. Available: https://underactuated.csail.mit.edu

[20] H. Geyer, A. Seyfarth, and R. Blickhan, “Compliant leg behaviour explains basic dynamics of walking and running,” *Proc. R. Soc. B Biol. Sci.*, vol. 273, no. 1603, pp. 2861–2867, Nov. 2006, doi: 10.1098/rspb.2006.3637.

[21] P. Holmes, R. J. Full, D. Koditschek, and J. Guckenheimer, “The Dynamics of Legged Locomotion: Models, Analyses, and Challenges,” *SIAM Rev.*, vol. 48, no. 2, pp. 207–304, Jan. 2006, doi: 10.1137/S0036144504445133.

[22] T. McGeer, “Passive Dynamic Walking,” *Int. J. Robot. Res.*, vol. 9, no. 2, pp. 62–82, Apr. 1990, doi: 10.1177/027836499000900206.

[23] Toyota Research Institute, “Rethinking Contact Simulation for Robot Manipulation,” Toyota Research Institute. Accessed: Aug. 13, 2024. [Online]. Available: https://medium.com/toyotaresearch/rethinking-contact-simulation-for-robot-manipulation-434a56b5ec88

[24] M. Khadiv, S. A. A. Moosvian, A. Yousefi-Koma, M. Sadedel, A. Ehsani-Seresht, and S. Mansouri, “Rigid vs compliant contact: An experimental study on biped walking,” Dec. 13, 2018, *arXiv*: arXiv:1709.06314. doi: 10.48550/arXiv.1709.06314.

[25] D. A. Haggerty, M. J. Banks, P. C. Curtis, I. Mezić, and E. W. Hawkes, “Modeling, Reduction, and Control of a Helically Actuated Inertial Soft Robotic Arm via the Koopman Operator,” Nov. 16, 2020, *arXiv*: arXiv:2011.07939. Accessed: Aug. 13, 2024. [Online]. Available: http://arxiv.org/abs/2011.07939

[26] J. Ng and H. H. Asada, “Learned Lifted Linearization Applied to Unstable Dynamic Systems Enabled by Koopman Direct Encoding,” *IEEE Control Syst. Lett.*, vol. 7, pp. 1153–1158, 2023, doi: 10.1109/LCSYS.2022.3231641.

[27] B. Lusch, J. N. Kutz, and S. L. Brunton, “Deep learning for universal linear embeddings of nonlinear dynamics,” *Nat. Commun.*, vol. 9, no. 1, p. 4950, Nov. 2018, doi: 10.1038/s41467-018-07210-0.

[28] J. Kawamoto and F. Asano, “Active viscoelastic-legged rimless wheel with upper body and its adaptability to irregular terrain,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vilamoura-Algarve, Portugal: IEEE, Oct. 2012, pp. 157–162. doi: 10.1109/IROS.2012.6385579.

[29] S. Sanchez and P. A. Bhounsule, “Design, Modeling, and Control of a Differential Drive Rimless Wheel That Can Move Straight and Turn,” *Automation*, vol. 2, no. 3, Art. no. 3, Sep. 2021, doi: 10.3390/automation2030006.

[30] Y. Hanazawa, Y. Uchino, and S. Sagara, “Development of a rimless wheel robot with telescopic legs for step adaptability,” *Artif. Life Robot.*, vol. 29, no. 2, pp. 349–357, May 2024, doi: 10.1007/s10015-024-00943-w.

[31] M. Korda and I. Mezić, “Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control,” *Automatica*, vol. 93, pp. 149–160, Jul. 2018, doi: 10.1016/j.automatica.2018.03.046.