

**Developing Walking Controllers for a Bipedal Robot Using Analytical
Models and Data Driven Approaches**

BY

ERNESTO HERNANDEZ HINOJOSA
B.S., The University of Texas Rio Grande Valley, 2016

THESIS

Submitted as partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Mechanical Engineering
in the Graduate College of the
University of Illinois at Chicago, 2023

Chicago, Illinois

Defense Committee:

Pranav A. Bhounsule, Chair and Advisor
Myunghee Kim
James Patton, Biomedical Engineering
Milos Zefran, Electrical and Computer Engineering
Hananeh Esmailbeigi, Biomedical Engineering

Copyright by

ERNESTO HERNANDEZ HINOJOSA

2023

*Para mis papas Enrique Hernandez Mendiola y Maria Esperanza Hingosa de Hernandez,
los que sacrificaron sus sueños para que yo pudiera cumplir el mio.*

To my wife Marina Astrid Hernandez,

the one who has been with me every day of this journey. I love you.

*Para mis hermanos Enrique R. Hernandez Hingosa y Erick A. Hernandez Hingosa y
mis abuelos, gracias por apoyarme y alentarme siempre. Los quiero.*

ACKNOWLEDGMENTS

I would like to extend my heartfelt gratitude to everyone who has been a part of my doctoral journey.

Foremost, I owe immense appreciation to my advisor, Pranav A. Bhounsule, for the pivotal moment when he entrusted me with this project. His wisdom, unwavering belief, and invaluable guidance have left an indelible mark on me as a researcher. I hope to follow in his footsteps, aspiring to be a great mentor and educator as he is.

To my esteemed committee members, Prof. Myunghee Kim, Prof. James Patton, Prof. Hananeh Esmailbeigi, and Prof. Milos Zefran, I express my deep gratitude. Your generous investment of time, offering both guidance and feedback, has been very helpful throughout my journey.

My profound thanks extend to Prof. Amir Jafari for providing me with the opportunity to embark on this Ph.D. odyssey in his lab at the University of Texas at San Antonio. Thank you for your support and guidance.

Prof. Robert Lyle Hood, also from the University of Texas at San Antonio, deserves a special mention as my mentor during the digital extenders project. His guidance and wisdom have contributed to this path.

Dr. Juan Guevara, Dr. Natalia Guevara and Dr. Nazmul Islam from the University of Texas Rio Grande Valley, though my work may have taken a different direction, their mentorship has instilled in me a strong research ethic that I carry with pride.

ACKNOWLEDGMENTS (Continued)

A thank you to my former physics professor, Prof. Gianpietro Cagnoli from the University of Texas at Brownsville. His skepticism only fueled my determination, serving as a reminder that you should never let anyone dictate whether you're good enough.

To my past and present lab mates, my companions on this academic adventure, you have been a source of great camaraderie and friendship. From Christian, Chinonso, Nafiseh, Eric, Robert, Ali, Cori, Jonathan to my current lab mates Salvador, Jeremy, Daniel, Subramanian, Chun-Ming, Safwan, Prashanth, and Abhishek, it's been an honor to walk this path with you.

A special shout-out to the Justice League for their unwavering support and camaraderie and all my close friends who have been pillars of encouragement throughout this journey.

Finally, to my family, the bedrock of my life—my wife, my grandparents, parents, brothers, aunts, uncles, and cousins—your unshakable support and belief in me has carried me through this arduous journey. I owe my every success to your unwavering presence. Thank you all from the depths of my heart.

EHH

PREFACE

This project was partially funded by the American Heart Association Predoctoral Fellowship

ERNESTO HERNANDEZ HINOJOSA
October 31, 2023

CONTRIBUTION OF AUTHORS

Chapter 3: A significant portion of this chapter is reproduced from a published paper with the following citation: [Hernandez-Hinojosa, E., Satici, A., & Bhounsule, P. A. (2021, August). Optimal Control of a 5-Link Biped Using Quadratic Polynomial Model of Two-Point Boundary Value Problem. In International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (Vol. 85451). American Society of Mechanical Engineers.] in which Pranav A. Bhounsule and Ernesto Hernandez Hinojosa equally contributed to conceiving the idea; Ernesto Hernandez Hinojosa carried out the data collection and experimental simulations and wrote the manuscript; and Pranav Bhounsule and E., Satici advised and edited it.

Chapter 4: A significant portion of this chapter is reproduced from a published paper with the following citation: [Hernandez Hinojosa, E., Torres, D., & Bhounsule, P. A. (2022, November). Quadratically constrained quadratic programs using approximations of the step-to-step dynamics: application on a 2D model of Digit. In 2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids) (pp. 96-103). IEEE.]. Ernesto Hernandez Hinojosa and Pranav Bhounsule conceived the idea; Ernesto Hernandez Hinojosa conducted the data collection and simulation experimentation and wrote the manuscript; Pranav Bhounsule and Daniel Torres advised and edited.

CONTRIBUTION OF AUTHORS (Continued)

Chapter 5: A significant portion of this chapter is reproduced from an accepted conference paper with the following citation: [Hernandez Hinojosa, E., and Bhounsule, P. A., "Data-driven Identification of a Non-homogeneous Inverted Pendulum Model for Enhanced Humanoid Control" Proceedings of the 2023 IEEE-RAS International Conference on Humanoid Robots. Austin, Texas. Dec 12–14, 2023.]. Ernesto Hernandez Hinojosa and Pranav Bhounsule conceived the idea; Ernesto Hernandez Hinojosa conducted the data collection, simulation experimentation, hardware experimentation and wrote the manuscript; Pranav Bhounsule advised and edited.

TABLE OF CONTENTS

<u>CHAPTER</u>	<u>PAGE</u>
1 INTRODUCTION	1
1.1 Expected Significance	3
1.2 Contributions	4
1.2.1 Data-driven Approximations of the Step-to-Step Dynamics . . .	4
1.2.2 Application of Data-derived Analytical Models for Stepping Control of Digit	5
1.2.3 Asymmetric Stepping of Digit Using Model Predictive Control With Analytical Constraints for Obstacle Avoidance	6
2 BACKGROUND	8
2.1 Pendulum Dynamics - Phase Portraits	8
2.2 Step-to-step Map	14
2.3 The Linear Inverted Pendulum Phase Portrait	19
3 OPTIMAL CONTROL OF A 5-LINK BIPED USING QUADRATIC POLYNOMIAL MODEL OF TWO-POINT BOUNDARY VALUE PROBLEM	23
3.1 Introduction	24
3.2 Background and Related Work	25
3.3 Robot model	28
3.3.1 Single stance equations	29
3.3.2 Foot-strike equations	30
3.3.3 Simulating a single step	31
3.4 Methods	32
3.4.1 Partial feedback linearization	32
3.4.2 Step-to-step dynamics: Poincaré map	33
3.4.3 Approximating the Poincaré map	35
3.4.3.1 Data generation	35
3.4.3.2 Estimating the boundary of the Poincaré map	35
3.4.3.3 Quadratic polynomial model of the Poincaré map	36
3.4.4 Quadratically constrained quadratic program	37
3.5 Results	38
3.5.1 SVM Classification	38
3.5.2 Quadratic polynomial regression	38
3.5.3 Optimization: Following reference velocity	39
3.5.4 Optimization: Terrain with ditches	40
3.6 Discussion	42

TABLE OF CONTENTS (Continued)

<u>CHAPTER</u>		<u>PAGE</u>
3.7	Conclusions and Future Work	44
4	QUADRATICALLY CONSTRAINED QUADRATIC PROGRAMS USING APPROXIMATIONS OF THE STEP-TO-STEP DYNAM- ICS: APPLICATION ON A 2D MODEL OF DIGIT	46
4.1	Introduction	47
4.2	Background and Related Work	48
4.3	Models	50
4.3.1	Robot model	50
4.3.2	S2S Models	51
4.3.2.1	Analytical map from Linear Inverted Pendulum Model (LIPM)	51
4.3.2.2	Regression-based map from MuJoCo simulator	52
4.4	Stepping Controllers	53
4.4.1	Model-based Stepping	54
4.4.2	Model-based Stepping with Feedback	55
4.4.2.1	Analytical, LIPM based	56
4.4.2.2	Regression, simulator based	56
4.4.3	Footstrike-corrected Stepping	58
4.5	The Linear Inverted Pendulum Model	60
4.6	Joint-level Control	60
4.6.1	Gravity Compensation	61
4.6.2	Joint Trajectory and PD control	62
4.6.2.1	Model-based stepping	62
4.6.2.2	Model-based stepping with feedback	62
4.7	Methods	63
4.7.1	Data Collection using simulator	63
4.7.2	Feasible Boundary from simulator data	64
4.7.3	Polynomial Regression for S2S map from simulator data	65
4.7.4	Quadratically Constrained Quadratic Program	65
4.8	Results	66
4.8.1	Support Vector Machine Classification	66
4.8.2	Quadratic Polynomial Regression	67
4.8.3	Stepping Control: tracking a reference velocity	68
4.8.4	Footstrike-corrected Controller	70
4.8.5	Optimization: Stepping over obstacles	72
4.9	Discussion	73
4.10	Conclusions and Future Work	76
5	DATA-DRIVEN IDENTIFICATION OF A NON-HOMOGENEOUS INVERTED PENDULUM MODEL FOR ENHANCED HUMANOID CONTROL	77
5.1	Introduction	78

TABLE OF CONTENTS (Continued)

<u>CHAPTER</u>		<u>PAGE</u>
5.2	Background and Related Work	79
5.3	Models	81
5.3.1	Robot model	81
5.3.2	Step-to-step (S2S) Models	81
5.3.2.1	Analytical map from the LIPM	82
5.3.3	System homogeneity and state-independence	83
5.3.3.1	Non-homogeneous LIPM	84
5.3.4	Enhancing the NH-LIPM with Forced Inputs	86
5.4	Methods	86
5.4.1	Stepping Controllers	86
5.4.1.1	Model-based Stepping	87
5.4.2	Data Collection	88
5.4.3	Continuous Velocity Tracking Using Toe Torques	90
5.4.4	Walking Stabilization Using F-LIPM Control	94
5.4.5	Recursive Least Squares (RLS)	95
5.5	Results and Discussion	97
5.5.1	Regression Analysis (Simulation)	97
5.5.2	Regression Analysis on External Forces	100
5.5.3	Stepping Control: tracking a reference velocity	102
5.5.4	F-NH-LIPM: tracking a reference velocity with varying ankle damping	103
5.5.5	F-NH-LIPM: velocity and step length control	104
5.5.6	Walking Stabilization Using F-LIPM Control	108
5.5.7	Real-Time Adaptation of Linear S2S Model	111
5.5.8	Hardware Results: Extracting the Non-homogeneous LIPM	113
5.5.9	Hardware Results: Continuous Velocity Tracking Using F-LIPM Control	116
5.6	Conclusion and Future Work	117
6	OPTIMAL ASYMMETRIC STEPPING CONTROL OF DIGIT USING ANALYTICALLY IDENTIFIED MODELS	120
6.1	Introduction	120
6.2	Push Disturbance Recovery	121
6.3	Frontal Asymmetric Stepping	124
6.4	Lateral Asymmetric Stepping	128
6.4.1	Lateral Asymmetric Stabilization Using Toe Torques	130
6.4.2	Lateral Asymmetric Stepping on Hardware	132
6.5	Model Predictive Control	134
6.5.1	MPC Error Propagation	141
6.5.2	NH-LIPM MPC	143
6.5.3	MPC Model Comparisons	145
6.5.4	Multi-Stage Receding Horizon MPC	145

TABLE OF CONTENTS (Continued)

<u>CHAPTER</u>		<u>PAGE</u>
6.5.5	Single-stage MPC vs. Multi-stage MPC Comparison	149
6.5.6	Analytical Discrete Finite-Horizon Control Algorithm	150
6.5.6.1	Analytical Discrete Finite-Horizon Control Simulation Experiment	157
6.5.7	Asymmetric Stepping Controller Comparison	159
6.5.8	Conclusion	160
7	MODEL ADAPTIVE CONTROL USING THE RECURSIVE LEAST SQUARES (RLS) ALGORITHM AND MODEL INTEGRAL FEEDBACK	163
7.1	Introduction	163
7.2	Recursive Least Squares (RLS)	164
7.3	Model Integral Feedback Control (MIF)	166
7.4	Results	168
7.5	Conclusion	171
8	SOFTWARE FRAMEWORK	173
8.1	Introduction	173
8.2	Communication Diagram	174
8.3	Main Program	175
9	CONCLUSION	177
	APPENDICES	181
	Appendix A	182
	CITED LITERATURE	185
	VITA	191

LIST OF TABLES

<u>TABLE</u>		<u>PAGE</u>
I	Regression coefficients	39
II	Model prediction MAE comparison of MPC showing improvement using the NH-LIPM MPC over the LIPM MPC.	145
III	Model prediction average error comparison of MPC showing improvement using the multi-stage RHC MPC over the single-stage MPC in predicting state v_3^-, v_4^+, x_2^+ and x_3^+	149
IV	Table of Tunable Coefficients	157
V	Comparison of the successful trials of the three proposed MPC methods where the analytic finite horizon controller had the greatest success. . . .	160

LIST OF FIGURES

<u>FIGURE</u>		<u>PAGE</u>
1	Phase portrait of a simple inverted pendulum with natural frequency ω_0 . The arrows point in the direction of motion.	9
2	(a) Dynamics of a simple pendulum experiencing damping and frictional forces are shown in the red dotted line. (b) Dynamics of simple pendulum restored to cyclic stability by adding an external force u	10
3	(a) The region of the phase portrait describing the inverted pendulum dynamics. (b) Inverted pendulum moving down a slope under the influence of gravity.	12
4	Illustration of the section of interests of the S2S map for a simple pendulum.	16
5	Illustration of the section of interests of the S2S map for a a bipedal robot.	17
6	(a) Illustration of the Poincare map of a simple pendulum with a discrete control input u that causes a velocity jump in the pendulum when the position of the pendulum is zero. (b) Illustration of the Poincare map of a biped with a discrete control input u that drives the dynamics of the system from the states \mathbf{Q}_k to \mathbf{Q}_{k+1}	18
7	Illustration of the phase portrait of the LIPM with parameter $T_c = 0.296$	21
8	Illustration of the phase portrait of the LIPM with two P1 orbits shown in red and one P2 orbit shown in black.	22
9	Overview of the approach: (a) PFL reduces the stance phase dynamics from $\Theta = [\Theta_u, \Theta_c]$ (10 dimensions) to $\Theta = \Theta_u$ (2 dimensions). (b) A Poincaré section is chosen at mid-stance. We generate random input state at the Poincaré section and controls at the step and simulate till the next Poincaré section to generate data for the Poincaré map given by \mathbf{F} , $\Theta_u^{i+1} = \mathbf{F}(\Theta_u^i, \mathbf{U}^i)$. (c) The Poincaré map is curve fitted $\Theta_u^{i+1} = \overline{\mathbf{F}}(\Theta_u^i, \mathbf{U}^i)$ where $\overline{\mathbf{F}}$ is a quadratic polynomial model and support vector machine is used to identify the boundary of the model. (d) Non-linear programming is used to solve a suitably formulated quadratically constrained quadratic program. For a video see [1].	24
10	Humanoid model: (a) configuration variables describing the degrees of freedom, (b) mass, center of mass, inertia about center of mass, and length parameters	28
11	Pictorial representation of a single step	31
12	Feature space and optimal hyperplane of test set.	38
13	Classification confusion matrix	39
14	Sinusoidal velocity tracking performance	41
15	Sinusoidal velocity tracking control inputs	41
16	Simulation of humanoid walking over four ditches.	42
17	Velocity before and after every step for planning on ditches	42

LIST OF FIGURES (Continued)

<u>FIGURE</u>		<u>PAGE</u>
18	Controls at every step for planning on ditches	42
19	(a) Robot model: Digit bipedal robot with 30 degrees of freedom and 20 actuated joints. (b) A parallelogram closed chain was used to simplify the 6-bar linkage of the leg.	50
20	The general S2S model.	51
21	Model-based stepping. The stepping controller dictates the step size u_k to achieve desired velocity v^*	53
22	Model-based feedback controller block diagram.	58
23	Footstrike-corrected feedback controller block diagram.	60
24	Feature space and optimal hyperplane of test set.	67
25	Surface plots of the S2S modelling errors.	68
26	Velocity tracking using the stepping controller (a) , stepping controller with feedback (b) and comparison between the LIPM (v_{COM}^L) and regression model (v_{COM}^R). (c) Velocity tracking using regression model with feedback (v_{COM}^f) and footstrike-corrected feedback control (v_{COM}^{uf}).	69
27	Position data of COM during perturbation trial using a footstrike-corrected controller (Top) and a model-based controller (Bottom).	71
28	Velocity data of COM during perturbation trial using a footstrike-corrected controller (Top) and a model-based controller (Bottom).	71
29	Simulation of Digit avoiding a trip by stepping over two obstacles (indicated by the red arrow).	74
30	Velocity (top) and COM position (bottom) with respect to stance foot of Digit while walking over obstacles.	74
31	Robot model: Digit bipedal robot with 30 degrees of freedom and 20 actuated joints.	82
32	The S2S model maps the state at s_k^+ with the state at s_k^- and control during the step u_k	82
33	Illustration of the phase portrait of the LIPM with two P1 orbits shown in red and one P2 orbit shown in black.	84
34	Phase asymmetry displayed in NH-LIPM dynamics (black and purple trajectories)	84
35	Model-based stepping. The stepping controller dictates the step size u_k to achieve desired velocity v^*	87
36	LIPM diagram showing the desired continuous dynamics trajectory in green and the actual robot dynamics in red.	91
37	(a) Test data overlaid on the LIPM phase portrait. (b) Test data overlaid on the NH-LIPM phase portrait.	99
38	Surface plots of the S2S modeling errors seen in the regression testing.	100
39	Surface plots of the S2S $v_y(t)$ modelling error seen in the regression testing.	100
40	$v_x(t)$ error band comparison.	102
41	NH-LIPM: Velocity tracking model comparison along the fore-aft (x) direction (a) , and the lateral (y) direction (b)	103

LIST OF FIGURES (Continued)

<u>FIGURE</u>		<u>PAGE</u>
42	F-NH-LIPM: Velocity tracking at 2.85, 11.42, 17.13, and 5.7 Ns/m ankle damping input.	104
43	F-NH-LIPM: Using step length and ankle torque (a) for control of v^+ (b) and x^- (c) using the F-NH-LIPM controller.	105
44	LIPM: Velocity tracking using a LIPM-based controller with passive stance toe joints	106
45	F-LIPM: Velocity tracking using a F-LIPM-based controller with stance toe joint actuation	106
46	NH-LIPM: Velocity tracking using a NH-LIPM-based controller with passive stance toe joints	107
47	F-NH-LIPM: Velocity tracking using a F-NH-LIPM-based controller with stance toe joint actuation	107
48	F-LIPM: Toe torques displayed during the trial.	107
49	F-LIPM Stability: LIPM stepping (a) and F-LIPM stepping (b) applied to walking along the straight dotted line with a stepping period of 0.55 seconds leading to instability in the LIPM stepping controller.	109
50	Lateral (y) state data of walking simulation using LIPM stepping. The red and green dots show some asymmetric v_y^- and y^+ states that arise due to instability.	109
51	Frontal (x) state data of walking simulation using LIPM stepping. The red and green dotted circles show some areas where asymmetric v_x^- and x^+ states arise due to instability.	109
52	Lateral (y) state data of walking simulation using F-LIPM stepping.	110
53	Frontal (x) state data of walking simulation using F-LIPM stepping.	110
54	RLS controller: Lateral (a) and forward (b) walking velocity tracking comparison between the adaptive RLS and the LIPM stepping controllers. Failure occurs during lateral walking with LIPM controller.	112
55	RLS controller: (a) Deliberate damping variations were used in the trial. (b)-(c) the RLS controller performed online tuning of the model parameters	112
56	v_{des}^+ tracking hardware experiment using the NH-LIPM controller.	113
57	Hardware: Velocity tracking comparison between NH-LIPM and the LIPM performed on Digit.	114
58	Surface plots of the continuous S2S $v_x(t)$ modelling error seen in the regression testing.	115
59	Forward walking experiment testing the use of the LIPM-based controller and the NH-LIPM-based controller to follow a velocity trajectory of 0.1 m/s for a distance of approximately 3.0 m. The red dot shows the position of the desired trajectory.	116
60	LIPM: COM position and velocity data during the velocity tracking trial.	116
61	NH-LIPM: COM position and velocity data during the velocity tracking trial.	116

LIST OF FIGURES (Continued)

<u>FIGURE</u>		<u>PAGE</u>
62	Forward walking experiment testing the use of the LIPM-based controller and the NH-LIPM-based controller to follow a velocity trajectory of 0.1 m/s for a distance of approximately 3.0 m. The red dot shows the position of the desired trajectory.	117
63	F-LIPM: COM position and velocity data during the velocity tracking trial.	117
64	Forward Push Recovery	122
65	Forward push recovery data Showing an increase in velocity (red arrow) followed by a long step forward (yellow arrow).	122
66	Backward Push Recovery	123
67	Backward push recovery data Showing a decrease in velocity (red arrow) followed by a long step backward (yellow arrow).	123
68	Sideways Push Recovery	124
69	Rightward push recovery data Showing an increase in rightward velocity (red arrow) followed by a wide rightward step in frame (b), a wide step leftward in frame (c) and a short step rightward in frame (d)	124
70	Asymmetric stepping control for stepping over an obstacle.	125
71	Asymmetric stepping hardware experiment. (a) the robot takes a nominal step size followed by (b) a short asymmetric step to drive the COM forward, then (c) a long step is taken to slow down the velocity of the COM and step over the obstacle on the floor and finally (d) the robot successfully clears the obstacle.	126
72	Data from the hardware experiment using asymmetric stepping control to step over an obstacle. The state s_1^- depicts the short step taken to increase the velocity of the COM. The state s_2^- depicts a velocity increase to over 0.6 m/s and therefore the the robot uses a larger control state x_2^- to slow id town. The state s_3^- depicts another large control state x_3^- however v_3^- has slowed down.	127
73	Lateral asymmetric stepping control used to step over an obstacle laterally.	128
74	Lateral asymmetric stepping simulation.	129
75	Instability arises when the lateral velocity is to high.	131
76	Stabilization of the lateral motion can be stabilized using toe/ankle torques.	131
77	(a) Simulation data of lateral asymmetric stepping with instability leading to a fall. (b) Simulation data of lateral asymmetric stepping where instability is stabilized.	132
78	Asymmetric lateral stepping hardware experiment where a wide step is taken at (b) and stabilizes after four steps at (f).	133
79	Asymmetric lateral stepping hardware experiment where a wide step is taken at (b) and stabilizes after two steps at (d).	133
80	Hardware experiment data of the lateral (y axis) COM states.	134
81	MPC initialization used to plan control states to step over an obstacle in three steps.	135

LIST OF FIGURES (Continued)

<u>FIGURE</u>		<u>PAGE</u>
82	The MPC algorithm flow chart	136
83	Single-Stage Multi-Step MPC illustration of predictive planning for three steps.	139
84	Single-Stage Multi-Step MPC framework applied to walking over obstacles in simulation.	139
85	LIPM-MPC results showing, from top to bottom, the COM velocity, the predicted x_k^+ states, the predicted v_k^+ states, and the control inputs (step widths).	140
86	LIPM-MPC results showing, from top to bottom, the predicted x_k^+ states, the predicted v_k^+ states.	141
87	NH-LIPM-MPC results showing, from top to bottom, the COM velocity, the predicted x_k^+ states, the predicted v_k^+ states, and the control inputs (step widths).	144
88	NH-LIPM-MPC results showing, from top to bottom, the predicted x_k^+ states, the predicted v_k^+ states for the third obstacle.	144
89	Multi-stage receding horizon MPC used to step over an obstacle in three steps.	146
90	A NH-LIPM multi-stage RHC MPC was used to plan the motion of the robot walking over four consecutive floor obstacles.	146
91	A closer look at the robot states leading up to the second obstacle where the MPC stages occur at (b),(c), and (d).	147
92	Multi-stage RHC MPC state estimates.	149
93	Analytical Discrete Finite-Horizon Control to step over an obstacle.	150
94	Finite-horizon control parameters in an environment with an obstacle.	151
95	Feasible space of the optimization function and the analytical center plane.	152
96	A plane fitted to the center of the feasible space of the obstacle distance parameter D and initial states x_1^- and v_1^-	153
97	Analytical Discrete Finite-Horizon Control Algorithm Block Diagram	155
98	Numerical simulation footprint trail of the analytical finite-horizon control algorithm.	157
99	The analytical discrete finite-horizon controller was used to walk over four floor obstacles at different walking speeds.	158
100	X-position, x-velocity, stage number, and control value plots of the obstacle avoidance simulation using the analytical discrete finite-horizon controller.	159
101	Block diagram of the MIF controller.	167
102	Plots of the x-velocity, x-position, and RLS parameter values of the 0.6 m/s forward walking experimental trial showcasing adaptive RLS ability to reduce modeling and tracking errors by updating the model parameters.	169
103	Plots of the x-velocity, x-position, and RLS parameter values of the 0.6 m/s forward walking experimental trial showcasing adaptive RLS ability to reduce modeling and tracking errors by updating the model parameters.	171

LIST OF FIGURES (Continued)

<u>FIGURE</u>		<u>PAGE</u>
104	The example figure C	174

NOTATIONS

Bold lowercase letters are used to denote the vectors and bold uppercase letters for matrices.

The following mathematical notations are used throughout this thesis:

$ x $	the absolute value of a scalar x
$\mathbf{x}_m(k)$	the k^{th} element of vector \mathbf{x}_m
$[\mathbf{X}]_{i,j}$	the $(i, j)^{\text{th}}$ element of matrix \mathbf{X}
$\ \mathbf{x}\ _p$	the l_p -norm of \mathbf{x} , defined as $(\sum_k \mathbf{x}(k) ^p)^{\frac{1}{p}}$
j	the imaginary unit <i>i.e.</i> , $j = \sqrt{-1}$

SUMMARY

The control of bipedal robots poses significant challenges due to their intricate dynamics and limited actuation, often rendering them only partially controllable. Traditional control methods employ full-order models to plan joint movements and achieve desired robot motion. However, this approach involves complex numerical integration of the model, consuming considerable computational resources and impeding real-time control.

Alternatively, simplified models can capture essential aspects of a robot's dynamics and offer faster solutions for real-time control. While this method has effectively controlled lightweight, upper-body-free robots resembling point-mass inverted pendulums, it becomes more complex for robots like biped Digit with heavier torsos. This research project focuses on data-driven linear and nonlinear modeling to better represent the robot's dynamics than conventional models, such as the Linear Inverted Pendulum model.

The developed model supports the creation of a model-based stepping controller, stabilized through appropriate feedback control parameters, enabling stable steady-state walking and velocity tracking. Furthermore, it leverages the analytical models and data-extracted feasible action regions to formulate quadratic optimization problems. These problems find applications in safe and optimal control, particularly in scenarios where precise foot placement, such as navigating environments with obstacles, is essential.

This project introduces the non-homogeneous linear inverted pendulum model (NH-LIPM), enhancing the conventional LIPM by incorporating a non-homogeneous function into the equa-

SUMMARY (Continued)

tion. This addresses dynamics that deviate from the LIPM due to simplifications like assumptions about the center of mass location. The study takes a step further by introducing a forcing function to the model, accommodating biped dynamics influenced by external forces, such as toe/ankle torques or damping.

The forced-NH-LIPM (F-NH-LIPM) is introduced for walking control, capable of modeling known and controllable external forces such as toe/ankle torque and damping. This knowledge enables utilizing footstep and ankle torque as control inputs.

The proposed analytical models facilitate asymmetric stepping, allowing the robot to perform non-standard steps, critical for navigating constrained environments and obstacles. Model-predictive control (MPC) based on the analytical model enables the selection of control states within a discrete time horizon to guide the robot to its desired state.

Finally, the project explores adaptive techniques to stabilize the robot when its dynamics deviate from the model, enhancing the adaptability and robustness of bipedal robots.

CHAPTER 1

INTRODUCTION

Bipedal robots have long sought to replicate their animal counterparts' balance, walking control, and flexibility. However, despite considerable progress, they still face significant challenges in achieving robust locomotion suitable for real-world applications in households and warehouses. Notably, the complex dynamics of these robots, arising from their high dimensionality, nonlinearities, and under-actuation, present formidable hurdles to effective control. Researchers have developed two distinct approaches to tackle these issues, both leveraging the low-dimensional nature of walking. The first method involves creating a controller based on a template, such as the linear inverted pendulum model (see [2] for more details), which is then transferred to the full robot model via force and position control. However, this approach does not account for the crucial effects of torso and angular momentum. The second method focuses on developing a trajectory-tracking controller using a full-order robot model designed to behave like a low-order model by applying hybrid zero dynamics [3]. Yet, this method can encounter difficulties in stabilizing the system if it strays significantly from the reference trajectory. Bipedal robots have long sought to replicate the balance, walking control, and flexibility exhibited by their animal counterparts. However, despite considerable progress, they still face significant challenges in achieving robust locomotion suitable for real-world applications in households and warehouses. Notably, the complex dynamics of these robots, arising from their

high dimensionality, nonlinearities, and under-actuation, present formidable hurdles to effective control.

One of the major issues with controlling bipedal robots is the need to maintain stability in the face of perturbations, such as uneven terrain, unexpected obstacles, or changes in the robot's center of mass. To address this challenge, researchers have developed a variety of control strategies, ranging from simple, rule-based approaches to more sophisticated machine learning algorithms. One promising approach involves using feedback control to adjust the robot's gait in response to sensory input. This can be achieved by using sensors such as accelerometers, gyroscopes, and force sensors, which provide information about the robot's orientation, velocity, and ground contact forces. By continuously adjusting the robot's gait based on this sensory input, it is possible to maintain stability and achieve robust locomotion over a range of conditions. Another important consideration when designing bipedal robots is the need to balance trade-offs between agility, stability, and energy efficiency. For example, a robot that is highly agile and maneuverable may sacrifice stability and energy efficiency, while a robot that is optimized for stability may be less agile and more energy-intensive. Balancing these trade-offs requires careful consideration of the robot's design parameters, such as its mass distribution, joint stiffness, and actuator capabilities. Despite these challenges, bipedal robots hold tremendous promise for a wide range of applications, including search and rescue, manufacturing, and personal assistance. As researchers continue to refine their control strategies and improve the design of these robots, it is likely that we will see even more impressive feats of locomotion in the future.

1.1 Expected Significance

Presently, legged robots face limitations in their adaptability and their ability to maneuver efficiently in constrained environments, a trait observed in humans but often lacking in robotic counterparts [4]. To excel in such environments, legged robots must exhibit non-constant speed, dynamic steering, and employ asymmetric forces and torques, leading to the need for asymmetric gait patterns. The generation and stabilization of these gait patterns present formidable challenges since legged systems are inherently under-actuated, possessing more degrees of freedom than actuators. Consequently, they cannot be instantaneously stabilized, unlike the classic example of an inverted pendulum held upright. However, they can achieve stability over the course of one or more steps [5].

The proposed research takes advantage of the potential inherent in constructing low-order analytical models. These models encapsulate the robot's dynamics and are of sufficient simplicity to yield swift solutions while accommodating kinematic, dynamic, and environmental constraints. This capability enables real-time control in novel and challenging situations. The significance of this approach lies in its potential to foster the development of a versatile, broadly applicable control framework for the creation of practical legged robots suited for real-world applications. Beyond legged robots, this approach holds promise for application in other systems characterized by intermittent contact, such as climbing and juggling robots, as well as artificial devices like prosthetics and exoskeleton

1.2 Contributions

1.2.1 Data-driven Approximations of the Step-to-Step Dynamics

The first contribution of this thesis lies in the development of a novel method for crafting data-driven models that capture the intricate dynamics of bipedal robots during their step-to-step (S2S) transitions, commonly known as the S2S map. This analytical representation charts the evolution of the robot's states from one discrete step to the next. Currently, prevalent methodologies rely on solving the full-order dynamics to generate walking patterns offline, which are subsequently implemented online. Nevertheless, this approach confines the robot's motion to predefined trajectories, and any deviation from these trajectories may lead to unstable dynamics. Furthermore, deriving a step-to-step map through this method can be computationally taxing.

Alternative approaches involve employing reinforcement learning to model the robot's dynamics by iteratively learning the states and control combinations that yield the desired behavior. However, this modeling technique adopts a black-box approach, providing limited insight into the dynamics' behavior or avenues for model refinement. Another commonly adopted method employs simplified models, such as the Linear Inverted Pendulum Model, to forecast the robot's behavior, assuming the center of mass (COM) can be treated as a point mass kept constant. This approach, however, proves less effective for robots whose COM is distributed across multiple points and incorporates passive spring elements that introduce additional complexity to the dynamics.

This project introduces a novel approach that utilizes experimental data of the robot’s states and control combinations to design simplified models. Establishing an analytical model of the S2S map proves invaluable for real-time control, enabling dynamic responses and adjustments. Furthermore, simulation data serves the dual purpose of extracting the boundary distinguishing feasible and infeasible state-controller combinations, which can subsequently be incorporated as constraints in the control law. These contributions are explored in-depth in Chapters 3 and 4.

1.2.2 Application of Data-derived Analytical Models for Stepping Control of Digit

The second key contribution of this thesis is the practical application of data-derived models to design stepping controllers tailored for Digit. The analytical step-to-step (S2S) map serves as the linchpin, enabling the calculation of requisite control inputs, predominantly foot placements, necessary to propel the robot toward a target state. Stepping stability is bolstered by introducing error feedback into the stepping controller. This strategic augmentation ensures that the error dynamics of the controller steadily converge to zero, which can be achieved by ensuring the eigenvalues of the controller coefficient matrix remain stable, characterized by magnitudes less than one.

The insights garnered from data analysis during the model extraction training trials, facilitate the extraction of feasible hyperplanes. These hyperplanes can then be incorporated as inequality constraints within the optimization framework, lending further robustness and precision to the overall control paradigm. Moreover, the flexibility to employ first-order or second-order models as equality constraints in optimization problems represents a notable stride in solving optimal foot placements while meticulously adhering to safety constraints. Remarkably,

this novel approach marks the maiden instance where a data-derived analytical model has been harnessed for the development of stepping controllers and testing on a hardware model like Digit.

Additionally, a novel non-homogeneous LIPM (NH-LIPM) is introduced, which incorporates the conventional LIPM dynamics while incorporating a non-homogeneous term. This innovative addition effectively elucidates and rectifies the modeling discrepancies between the LIPM and the actual robot dynamics. Through the process of fitting a function to the model error and seamlessly integrating it with the LIPM, we achieve a more comprehensive representation of the model, all while preserving the fundamental LIPM characteristics. Worth noting is that this non-homogeneous function exhibits both state and time-dependent attributes, with its polynomial order serving as a dynamic parameter, aligning with our precision requirements. However, it's essential to consider that opting for higher polynomial orders may necessitate linearization to manage the complexities introduced by nonlinear models while maintaining system stability. These contributions are explored in-depth in Chapters 4 and 5.

1.2.3 Asymmetric Stepping of Digit Using Model Predictive Control With Analytical Constraints for Obstacle Avoidance

Walking in unconstrained, open spaces lends itself well to stable symmetric stepping. In such contexts, the pivotal factor for achieving stability lies in the careful selection of appropriate feedback gains to maintain controller stability. However, the demands of navigating through constrained environments extend beyond mere stability considerations.

The third significant contribution of this thesis lies in the application of data-derived simplified Step-to-Step (S2S) models to a Model Predictive Control (MPC) framework, boasting a planning horizon of up to three steps. MPC serves as an effective approach for determining optimal foot placement, facilitating precise, safe stepping while mitigating the risks of slips or falls.

To address the specific challenge of traversing ground obstacles, we introduce three distinct MPC frameworks. Notably, the discrete finite-horizon MPC framework stands out for its efficiency, requiring only one optimization procedure per obstacle. This efficiency is driven by analytical functions that govern foot placement, contingent on the current states, with a focus on the two steps leading up to an obstacle. For a more comprehensive exploration of these contributions, refer to Chapter 5.

CHAPTER 2

BACKGROUND

2.1 Pendulum Dynamics - Phase Portraits

The dynamics of a simple pendulum can provide insight into simple dynamic models for bipedal walking. The equation of motion for a simple pendulum involves the gravitational force and the restoring force of the pendulum's motion, resulting in a sinusoidal motion. This equation of motion can be used to model the dynamics of the inverted pendulum model, which is commonly used to describe the dynamics of bipedal walking. The inverted pendulum model considers the human body as a mass on top of a vertical rod, where the motion is analogous to the simple pendulum. By analyzing the phase portrait of the simple pendulum, one can gain a better understanding of the stability of the model, which can then be applied to the stability and control of bipedal walking. The phase portrait of the simple pendulum is a plot of the angular position versus the angular velocity, which displays the behavior of the pendulum over time. The shape of the phase portrait can provide insight into the stability and behavior of the system.

The phase portrait of a simple pendulum is shown in Figure 1. The phase portrait generated assumes that no external forces other than gravity are acting on the pendulum; therefore, the pendulum's dynamics can be observed by tracing the orbits depicted by the colored arrows along the phase portrait. The orbits that form a circular trajectory are known as cyclic orbits, and

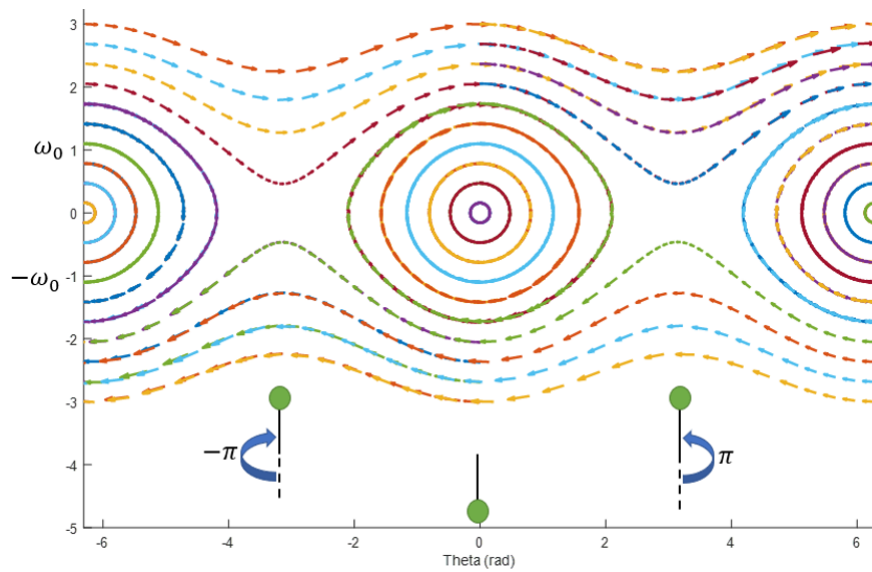


Figure 1: Phase portrait of a simple inverted pendulum with natural frequency ω_0 . The arrows point in the direction of motion.

the orbits that lie outside of the circular orbits are known as oscillating orbits. It is important to observe that the portrait lines never intersect each other. This is due to the conservation of energy in the system. The system's total energy, which is the sum of the kinetic and potential energy, remains constant. The equations of motion of a simple pendulum can be derived by integrating the dynamics equation (Equation 2.1) and are a function of start states θ_0 , $\dot{\theta}_0$ and time as shown in Equation 2.2.

$$\ddot{\theta} = \frac{g}{L} \sin(\theta) + \frac{u}{mL^2} \quad (2.1)$$

$$(\theta, \dot{\theta}) = P(\theta_0, \dot{\theta}_0, t) \quad (2.2)$$

When a simple pendulum experiences friction and/or damping, it will eventually converge to its equilibrium point due to the dissipation of its energy. As the energy of the pendulum decreases, the amplitude of its motion decreases as well until it reaches its equilibrium position as shown in Figure 2(a). However, adding an external force u to the pendulum can help it achieve cyclic stability as shown in Figure 2(b). The external force provides a source of energy that compensates for the energy loss due to friction and damping. This can result in regaining stable periodic motion.

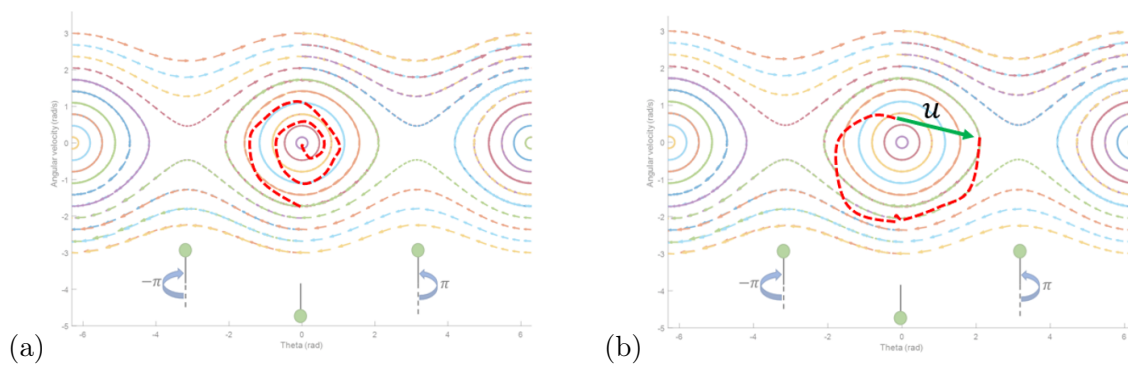


Figure 2: (a) Dynamics of a simple pendulum experiencing damping and frictional forces are shown in the red dotted line. (b) Dynamics of simple pendulum restored to cyclic stability by adding an external force u .

The addition of the external force can be modeled mathematically by adding a term representing the force to the equation of motion for the pendulum as shown in a simplified manner in Equation 2.3. This modified equation of motion can be analyzed to determine the conditions under which the pendulum achieves cyclic stability. The addition of the external force introduces the concept of control, where the control input u is used to stabilize the system and achieve the desired behavior. The equations of motion with external forces are described by Equation 2.4 where d are the external forces like friction and damping and \mathbf{u} is the control.

$$(\theta, \dot{\theta}) = P(\theta_0, \dot{\theta}_0, t, d) \quad (2.3)$$

$$(\theta, \dot{\theta}) = P(\theta_0, \dot{\theta}_0, t, d, \mathbf{u}) \quad (2.4)$$

Like a control input \mathbf{u} that can drive a pendulum to a cyclic orbit, a passive walker can also achieve cyclic motion. A passive walker is a bipedal robot that utilizes the natural dynamics of its body to walk without any external actuation. See [6] for more details. The robot walks down a slope with a fixed angle, and the natural dynamics of the system cause it to achieve cyclic motion also known as a limit cycle.

The passive walker's cyclic motion is achieved by a phenomenon called "dynamic stabilization" [7]. When the robot steps forward, its center of mass shifts forward as well. Due to the slope of the surface, gravity pulls the robot down the slope and accelerates its motion. As the robot accelerates, its trailing foot lifts off the ground and the body pivots about the stance

foot. This causes the center of mass to rise and shift backward (behind the new stance foot), resulting in a deceleration of the robot's motion. As the trailing leg swings forward the robot begins to accelerate again and the process repeats, leading to a cyclic motion. The dynamics of the passive walker can be described by looking at the region of the phase portrait of a simple pendulum where the pendulum position is inverted as shown in Figure 3 (a). The slope angle

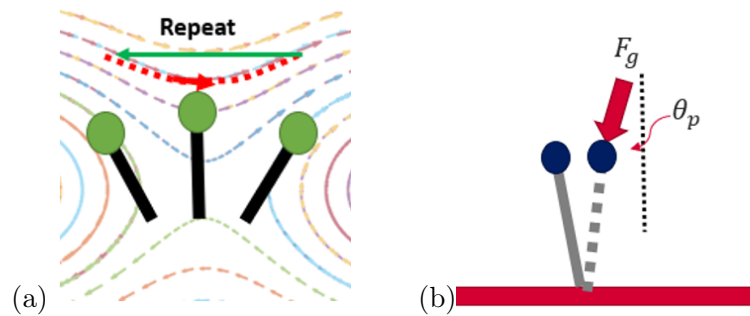


Figure 3: (a) The region of the phase portrait describing the inverted pendulum dynamics. (b) Inverted pendulum moving down a slope under the influence of gravity.

and the natural dynamics of the system determine the stability of the cyclic motion. By adjusting the slope angle or the mass distribution of the robot, the stability of the cyclic motion can be controlled. Like the control input \mathbf{u} in the case of the pendulum, the slope angle can be considered as the control parameter in the case of the passive walker as shown in Figure 3 (b), which can be used to achieve the desired cyclic motion. Assuming the walker behaves like a point-mass inverted pendulum, the equations of motion depend on the gravitational force as

shown in Equation 2.5 where x and \dot{x} are the position and velocity of the point-mass inverted pendulum.

$$(x, \dot{x}) = P(x_0, \dot{x}_0, t, F_g) \quad (2.5)$$

By choosing the appropriate inclination angle θ_p , the pendulum can attain equal start and end states at each cycle, achieving a limit cycle. In the case of cyclic stability, the equations of motion can be described by Equation 2.6, where the gravitational force is removed because it is constant for the case of cyclic stability.

$$(x, \dot{x}) = P(x_0, \dot{x}_0, t) \quad (2.6)$$

The equations of motion of a bipedal robot can be derived by integrating the dynamic equations that describe the motion of the system. For the case of a floating base biped with n joints and m actuators, the dynamic equations are of the form shown in Equation 2.7 where M is the mass-inertia matrix C is the Coriolis force vector, G is the gravitational force, B is the actuator selection matrix, J is the force Jacobian and F_G are the external forces.

$$M(q)\ddot{q} + C(q, \dot{q}) + G(q) = Bu_m + J_G^T F_G \quad (2.7)$$

The resulting equations of motion are a function of the robot's initial state, time and any torque inputs applied to the system as shown in Equation 2.8.

$$(\mathbf{q}, \dot{\mathbf{q}}) = B(\mathbf{q}_0, \dot{\mathbf{q}}_0, t, \mathbf{u}) \quad (2.8)$$

The initial state includes the position, velocity, and orientation of the robot's various body segments, as well as the angular velocities and accelerations of these segments. To derive the equations of motion, the dynamic equations are integrated over time using numerical integration techniques such as the Runge-Kutta method. The integration process involves discretizing time and approximating the system's motion at each time step. The torque inputs, which represent the external forces acting on the system, are also considered during the integration process.

2.2 Step-to-step Map

One way of achieving control of a system is by looking at the behavior of the system as a function of time and external forces. External forces can become control inputs to the system that drive the dynamics to the desired behavior. If a system is simple, the equations of motion can be analytically formulated by integrating the dynamics equation. However, most bipedal systems are more complex with higher-order terms that make solving the equations of motion analytically difficult. Numerical integration of the dynamics of a system is a useful tool to study the behavior of a system over time, given a set of initial conditions and control inputs. However, finding the control input that yields the desired behavior can be challenging, as one would have

to integrate the dynamics of the system for different input conditions until one found the one that works. This can be time-consuming and computationally expensive, especially for complex systems that involve many states and control inputs. Moreover, the dynamics of the system can be highly nonlinear and have multiple equilibria, making it even harder to find a control input that yields the desired behavior.

An alternative approach is using a step-to-step (S2S) map. Given the equations of motion of a system, we can derive a S2S map that describes the states of the system as it goes through cyclic motions from one cycle to the next. It relates the state of the system at the end of one step to the state at the beginning of the next step. This provides a useful tool for analyzing the behavior of the system and for designing control strategies that can achieve stable and efficient locomotion. The map can be used to determine the stability of the cyclic motion and to identify the conditions under which the system can achieve stable locomotion.

To derive the S2S map, we first simulate the system for one cycle using the equations of motion and a given set of control inputs. The resulting state of the system at the end of the cycle is then used as the initial condition for the next cycle. We repeat this process for many cycles with a sparse combination of control inputs and starting conditions and record the states of the system at the beginning and end of each cycle. The resulting data is used to construct the S2S map.

The control inputs used during the simulation can be continuous and can change over time. For example, in the case of a bipedal robot, the control inputs might include the joint torques applied to the legs and the feedback control laws that govern the robot's motion. By varying

the control inputs, we can explore the behavior of the system under different conditions and identify control strategies that can achieve stable and efficient locomotion.

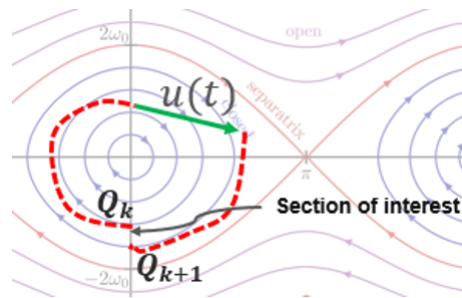


Figure 4: Illustration of the section of interests of the S2S map for a simple pendulum.

Overall, the S2S dynamics is a translation from the continuous dynamics to the discrete map of the uncontrolled states of a system at key states in a cycle. For a pendulum, the uncontrolled states are the angular position and velocity of the pivot. The section of interest indicates the start and end of each cycle and can be time or state-dependent. In Figure 4 the section of interest is state dependent. The section is at the state when the position is zero. In this case, the state Q_k is only the angular velocity of the pendulum as shown

$$(\theta, \dot{\theta}) = P(\theta_0, \dot{\theta}_0, t, d, u) \rightarrow \mathbf{Q}_{k+1} = P(\mathbf{Q}_k, u(t)) \quad (2.9)$$

$u(t)$ is the control input which can be an external force or torque applied to the pendulum. Note that although the position is an uncontrolled state, by choosing position as the parameter for the section of interest, the position becomes a controlled state assuming the section of interest is always achieved.

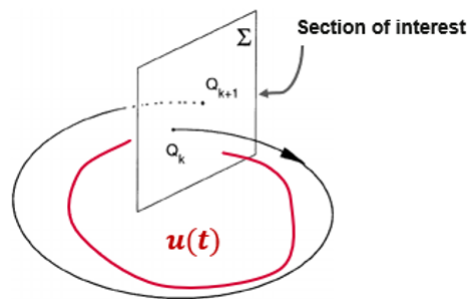


Figure 5: Illustration of the section of interests of the S2S map for a bipedal robot.

For a biped, the uncontrolled states are the virtual floating base joints and other passive joints. In Figure 5 the section of interest can be either time-dependent or state-dependent. We can choose the section of interest to be at the state when the foot strikes the ground. For a biped with 6 unactuated degrees of freedom, such as a floating base model, there are twelve uncontrolled states $\mathbf{Q}_k = (\mathbf{q}_k^6, \dot{\mathbf{q}}_k^6)$ and the control input $\mathbf{u}(t)$ are the actuator torques as shown

$$(\mathbf{q}, \dot{\mathbf{q}}) = B(\mathbf{q}_0, \dot{\mathbf{q}}_0, t, \mathbf{u}) \rightarrow (\mathbf{q}_k^6, \dot{\mathbf{q}}_k^6) = B(\mathbf{q}_k^6, \dot{\mathbf{q}}_k^6, \mathbf{u}(t)) \rightarrow \mathbf{Q}_{k+1} = B(\mathbf{Q}_k, \mathbf{u}(t)) \quad (2.10)$$

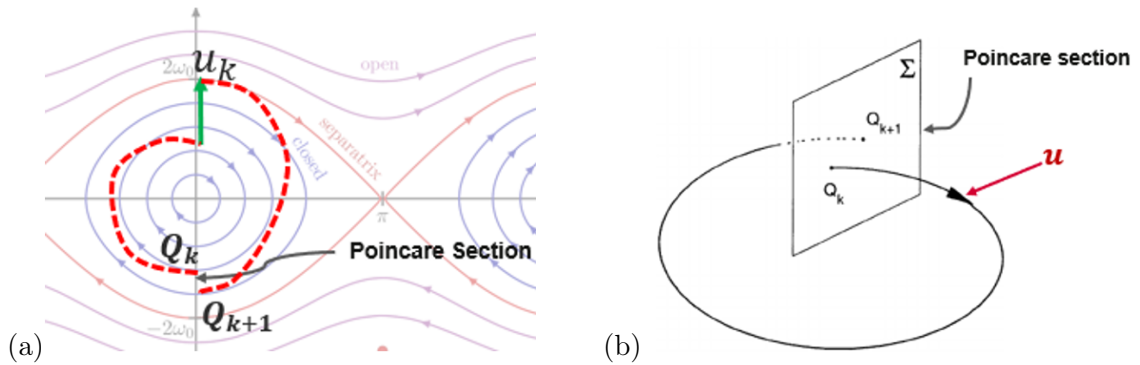


Figure 6: (a) Illustration of the Poincaré map of a simple pendulum with a discrete control input u that causes a velocity jump in the pendulum when the position of the pendulum is zero. (b) Illustration of the Poincaré map of a biped with a discrete control input u that drives the dynamics of the system from the states \mathbf{Q}_k to \mathbf{Q}_{k+1} .

The system can be characterized by a Poincaré map which maps un-controlled states at one instance of the natural cycle (or Poincaré section) to the states at the same section at the next step with discrete controlled inputs. Note that the terms S2S map and Poincaré map are commonly used interchangeably. For this thesis, I will refer to the Poincaré map as a subset of a S2S map where the control input is discrete. When the system is steady, the Poincaré map starts and ends at the same point. The Poincaré map can be used to evaluate states of the system such as the velocity of the base or position of the COM. The Poincaré map of a simple pendulum with a discrete control input u that occurs when the angle of the pendulum is zero and helps drive the uncontrolled states from \mathbf{Q}_k to \mathbf{Q}_{k+1} is shown in Figure 6(a). Similarly, the Poincaré map for a biped with discrete control input \mathbf{u} which can be the step length or the ankle push off of the trailing leg at foot strike that drives the dynamics of the system from the

uncontrolled states \mathbf{Q}_k to \mathbf{Q}_{k+1} is shown in Figure 6(b). The Poincare map equations for the pendulum and the biped models can be respectively expressed as

$$\mathbf{Q}_{k+1} = P(\mathbf{Q}_k, u) \quad (2.11)$$

$$\mathbf{Q}_{k+1} = B(\mathbf{Q}_k, u) \quad (2.12)$$

2.3 The Linear Inverted Pendulum Phase Portrait

When examining the motion of a system, two main approaches can be taken: integrating the system's dynamic equations or analyzing the system's step-to-step map. The former involves solving the system's equations of motion, usually in the form of differential equations, numerically to obtain the motion over time. This approach gives a continuous picture of the system's motion and can be used to determine the system's long-term behavior.

On the other hand, analyzing the step-to-step map involves studying the system's behavior between discrete points in time, such as between impacts or other discrete events. This approach is often used for systems with intermittent contacts, such as bipedal robots or bouncing balls. By analyzing the changes in the system's state at these discrete events, the step-to-step map can be used to understand the system's overall behavior.

There is a third approach that takes the best of the two approaches. Mainly the simplicity of a S2S map and the continuity of the numerical integration method. This approach maps the continuous dynamics of the system as a function of time by looking at specific states that play important roles in control. Two common state pairs independently used in the control of bipedal

robots are the stance leg ankle angle, angular velocity pair, and the center of mass position and velocity pair. The phase portraits of these state pairs commonly display a cyclic motion limit cycle during steady-state walking. In the phase portrait, this corresponds to a closed trajectory that the system follows as it cycles through its periodic motion. The limit cycle represents a stable attractor that the system tends towards, and any initial conditions within the limit cycle will result in the same periodic behavior. This is desirable in bipedal walking, allowing for stable, sustained locomotion without continuous control input.

The linear inverted pendulum model (LIPM) is commonly used to formulate the analytical continuous dynamics map [8]. The LIPM has been widely used to model bipedal robots due to its simplicity and ability to capture the essential dynamics of the system. Although bipedal robots exhibit more complex dynamics, the linear inverted pendulum model provides a useful starting point for understanding the basic principles of bipedal locomotion. Having an inverted pendulum with a constant height makes the dynamics of the pendulum linear with an analytical solution of the form

$$\ddot{x} = \frac{g}{z_c}x + \frac{1}{mz_c}\tau \quad (2.13)$$

where g is gravity, z_c is the constant height of the COM, m is the mass, and τ is the input torque. Assuming no input torques, the equation can be analytically integrated to get the following equations of motion

$$\begin{bmatrix} x(t) \\ v(t) \end{bmatrix} = \begin{bmatrix} C_T \cdot x_0 + t \cdot S_T \cdot v_0 \\ S_T/t \cdot x_0 + C_T \cdot v_0 \end{bmatrix} \quad (2.14)$$

where $S_T = \sinh(t/T_c)$, $C_T = \cosh(t/T_c)$, and $T_c = \sqrt{z_c/g}$.

The phase portrait of a linear inverted pendulum model will vary according to parameter T_c . Figure 7 shows one variation of the phase portrait of the LIPM.

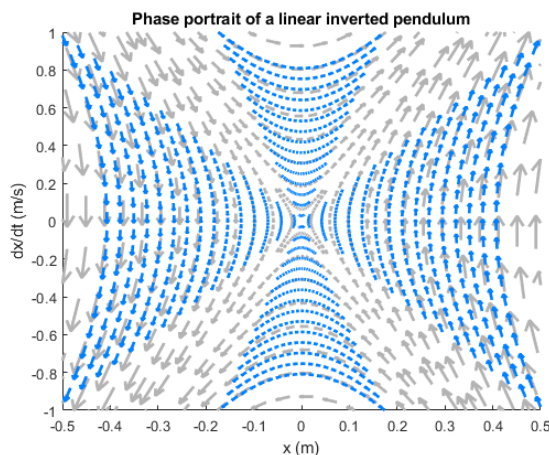


Figure 7: Illustration of the phase portrait of the LIPM with parameter $T_c = 0.296$

The gray arrows in the figure show the trajectories of the orbits of the phase portrait. The blue arrows show the symmetric and antisymmetric trajectories over a constant time interval. The symmetric trajectories display equal start and end velocities, whereas the antisymmetric trajectories display equal but opposite start and end velocities. A limit cycle is achieved in the symmetric orbits by having a discrete control input that changes the position of the center of

mass with respect to the pivot. The foot striking the ground during walking is an example of such a type of discrete control. By alternating the legs with ground contact, the position of the center of mass is changed at every step. The red trajectories shown in Figure 8 depict two limit cycles that repeat after every step. These types of orbits are known as P1 orbits [9]. The black trajectory consists of two antisymmetric trajectories forming a limit cycle that repeats after two steps. These types of orbits are known as P2 orbits.

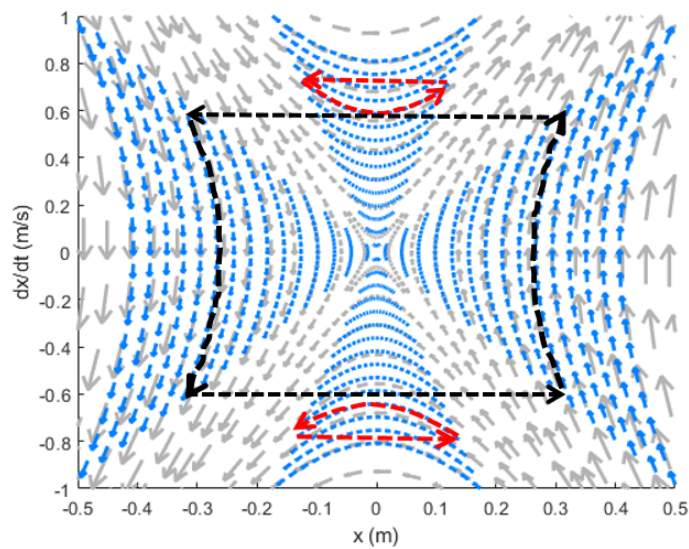


Figure 8: Illustration of the phase portrait of the LIPM with two P1 orbits shown in red and one P2 orbit shown in black.

CHAPTER 3

OPTIMAL CONTROL OF A 5-LINK BIPED USING QUADRATIC POLYNOMIAL MODEL OF TWO-POINT BOUNDARY VALUE PROBLEM

Overview: To walk on constrained environments, bipedal robots must meet concise control objectives of speed and foot placement. The decisions made at the current step need to factor in their effects over a time horizon. Such step-to-step control is formulated as a two-point boundary value problem (2-BVP). As the dimensionality of the biped increases, it becomes increasingly difficult to solve this 2-BVP in real-time. The common method to use a simple linearized model for real-time planning followed by mapping on the high dimensional model cannot capture the nonlinearities and leads to potentially poor performance for fast walking speeds. In this paper, we present a framework for real-time control based on using partial feedback linearization (PFL) for model reduction, followed by a data-driven approach to find a quadratic polynomial model for the 2-BVP. This simple step-to-step model along with constraints is then used to formulate and solve a quadratically constrained quadratic program to generate real-time control commands. We demonstrate the efficacy of the approach in simulation on

Parts of this chapter are taken from the following published journal article:
Hernandez-Hinojosa, E., Satici, A., & Bhounsule, P. A. (2021, August). Optimal Control of a 5-Link Biped Using Quadratic Polynomial Model of Two-Point Boundary Value Problem. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference* (Vol. 85451). American Society of Mechanical Engineers.

a 5-link biped following a reference velocity profile and on a terrain with ditches. A video is here:

<https://youtu.be/-UL-wkv4XF8>

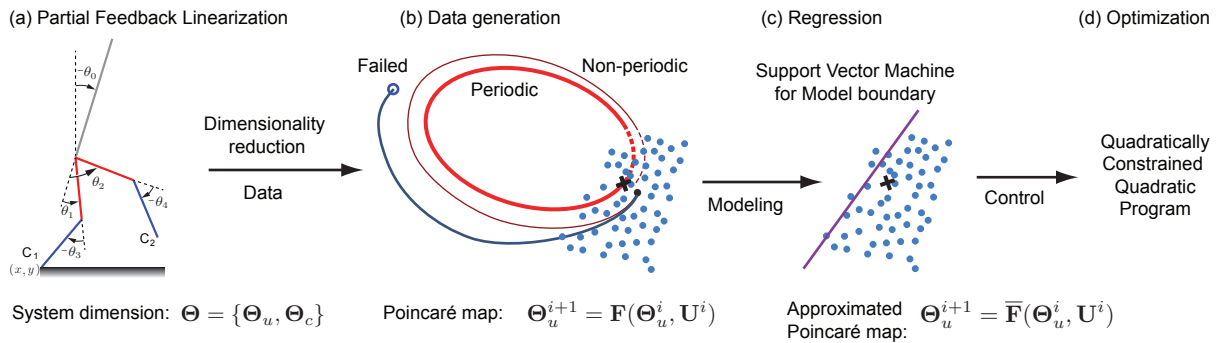


Figure 9: Overview of the approach: (a) PFL reduces the stance phase dynamics from $\Theta = [\Theta_u, \Theta_c]$ (10 dimensions) to $\Theta = \Theta_u$ (2 dimensions). (b) A Poincaré section is chosen at mid-stance. We generate random input state at the Poincaré section and controls at the step and simulate till the next Poincaré section to generate data for the Poincaré map given by \mathbf{F} , $\Theta_u^{i+1} = \mathbf{F}(\Theta_u^i, \mathbf{U}^i)$. (c) The Poincaré map is curve fitted $\Theta_u^{i+1} = \bar{\mathbf{F}}(\Theta_u^i, \mathbf{U}^i)$ where $\bar{\mathbf{F}}$ is a quadratic polynomial model and support vector machine is used to identify the boundary of the model. (d) Nonlinear programming is used to solve a suitably formulated quadratically constrained quadratic program. For a video see [1].

3.1 Introduction

Bipedal walking systems because of their human-like morphology are perhaps more suitable for integration in human environments such as homes and warehouses. However, bipeds are yet to achieve the dexterity and nimbleness seen in human movements. Controlling bipedal systems is a formidable challenge because of their unstable inverted pendulum-like nature that is instantaneously uncontrollable because of lack of adequate actuation at the base of the feet (also known as under-actuation). For example, when a standing robot is pushed, it cannot

stabilize itself in the vertical position, but needs to take a step forward to balance. These issues are further compounded by their high dimensionality (~ 20 degrees of freedom for a 3D biped).

Inspired from humans, the most successful control paradigm is to achieve balance and control over the time scale of a step, also known as step-to-step control [10]. A fundamental challenge with step-to-step control is that it is predictive: it needs to make instantaneous control decisions that affect the dynamics over the time scale of a step or the step-to-step dynamics. Thus, step-to-step control entails solving a suitable two-point boundary value problem (2-BVP). As the dimensionality of the biped increases, it becomes computationally challenging to solve this 2-BVP in real-time. Past control approaches have either used offline design with complete models [11] or used online design with simple models, which are mapped to the complete model all in real-time [12]. The former approach is not generalizable to novel scenarios while the latter is conservative because the simple models do not capture the complete dynamics. In this paper, we present an approach that enables fast computation of controllers using a complete model to enable real-time control. Our key idea is to use PFL to transform the complete model to low dimension, offline approximation of the 2-BVP using a quadratic polynomial model, and finally, online optimal control using this model.

3.2 Background and Related Work

One of the earliest demonstration of step-to-step control was through the concept of passive dynamic walking (PDW) [6]. In PDW, a walking frame resembling the human lower body settles into a cyclic or periodic gait when launched on a shallow slope with no external control. This idea has given rise to powered walkers that exhibit cyclic walking on level ground using hip and

ankle actuation. Some example robots include the Cornell biped [13], Cornell Ranger [14], Delft bipedal robots [15], Michigan State synthetic wheel biped [16], and Twente Dribbel [17]. One feature of these robots is that when executing cyclic walking, they are unstable instantaneously, but are stable over the time scale of a step [5]. This cyclic stability is known as orbital stability and is a primary means of balance in humans [18].

We analyze cyclic gaits and their orbital stability using concepts from non-linear dynamics [19]. First, we find the nominal rhythmic control that achieves cyclic or periodic gait [20]. Then we use the linearization of the cyclic gait to evaluate the orbital stability. To increase the orbital stability of the system, we create a linear model of the step-to-step dynamics and then we choose a suitable linear controller using either pole placement or discrete linear quadratic regulator [21, 22]. This approach is computationally simple as it is based on a linear controller, but it limits the region of stability to a narrow region around the cyclic gait.

One way to enable stability against large perturbation is to compute controllers (e.g., using nonlinear optimization) based on the non-linear model. This approach works well for offline controller development, as the control development requires heavy computation and takes quite some time to converge to the optimum. An alternate approach is to use a simple model (e.g., linear inverted pendulum) to plan the motion over a low-dimensional space, typically the center-of-mass motion and foothold location, and then use inverse kinematics and inverse dynamics to map to the non-linear model [12, 23]. One caveat of this approach is that the results depend on the accuracy of the simple models. These simple models ignore the nonlinearities and/or the

angular momentum of the upper body and have issues when planning for high speed location and/or those involving upper body movement.

The hybrid zero dynamics (HZD) approach does controller synthesis using a high-dimensional model [24,25]. Here one defines continuous-time outputs (also known as virtual constraints) that map the actuated degrees of freedom to the unactuated degrees of freedom. One then designs a controller to drive these outputs to zero. This method is attractive because it uses control to reduce the dimensionality of the system to the unactuated degrees of freedom and is hence scalable. However, it is not very easy to find these virtual constraints that lead to acceptable performance as the system complexity increases [26]. Although one can create exponentially stable continuous time controllers, the orbital stability is only asymptotically stable [27].

We present an overview of our approach and demonstrate it on a 5-link biped. As shown in Figure 9 (a), we use PFL to reduce the dimension from 10D to 2D. Next, as shown in Figure 9 (b), we use a Poincaré map for modeling the step-to-step dynamics which further reduces the system dimension to 1D, the velocity of the stance leg. We note that the step-to-step dynamics has 1 state variable, the mid-stance speed, and 2 control variables, the push-off impulse and the step angle. Next, as shown in Figure 9 (c), we approximate the step-to-step dynamics using a quadratic polynomial model and estimate its region of validity using support vector machine (SVM). Finally, as shown in Figure 9 (d), we use nonlinear programming to solve the resulting quadratically constrained quadratic program. For a video see [1]. The novelty of our work in comparison to past approaches (e.g., [28]) is the use of data-driven methods to derive a quadratic polynomial model of the 2-BVP and identify the region of validity. Unlike past

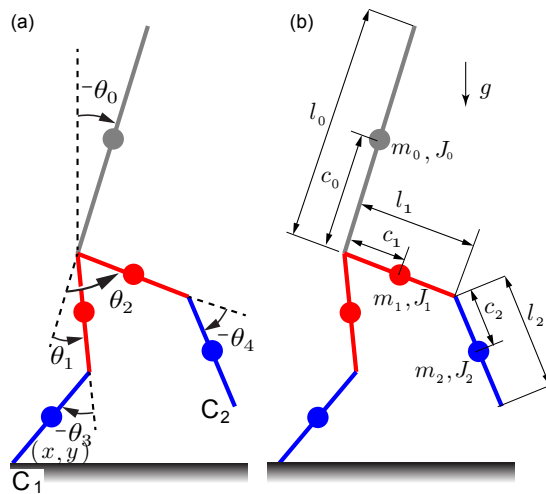


Figure 10: Humanoid model: (a) configuration variables describing the degrees of freedom, (b) mass, center of mass, inertia about center of mass, and length parameters

approaches that are based on linearization, our simple model covers a relatively extensive region around the cyclic gait and enables us to solve the 2-BVP in few iterations/function evaluations enabling real-time control in the future. This work builds upon our past work on control of a humanoid [29] by improving the step-to-step model approximation to compute the region of validity using SVM and subsequently we demonstrate that the resulting quadratic program can be solved in few iterations.

3.3 Robot model

Figure Figure 10 shows the 2D, 5-link model used in this study. This model was previously used in our past work [29] from which this study builds from. We define the stance leg as the one in contact with the ground and the swing leg is the other leg. The foot in contact with the ground has coordinates (x, y) where the x-axis is horizontal and y-axis is vertical. The torso angle θ_0 is the angle between the torso and the vertical direction, θ_1 and θ_2 are the relative

angles made by the thigh links of the stance and swing leg respectively with the torso, and θ_3 and θ_4 are the angles made by the calf links of the stance and swing leg respectively with their respective thigh links. The torso mass is $m_0 = 50$ kg, center of mass is at $c_0 = 0.5$ m, and inertia about the center of mass is $J_0 = 10$ kg-m². The thigh links have a mass of $m_1 = 7$ kg, center of mass is at $c_1 = 0.25$ m, and inertia about the center of mass is $J_1 = 5$ kg-m². The calf links have a mass of $m_2 = 5$ kg, center of mass at $c_2 = 0.25$ m, and inertia about the center of mass is $J_2 = 2$ kg-m². Gravity points downwards and is $g = 9.81$ m/s². The torso length $\ell_0 = 1$ m the thigh link and calf link lengths are equal, $\ell_1 = \ell_2 = 0.5$.

There are two sets of equations: one for the single stance phase where one foot is on the ground and the second for the foot-strike where the legs exchange roles.

3.3.1 Single stance equations

The state variables for derivation are defined as $\mathbf{q} = \begin{bmatrix} x & y & \theta_0 & \theta_1 & \theta_2 & \theta_3 & \theta_4 \end{bmatrix}^T$. We include the floating coordinates x and y to derive the equation, but the simplified equation has only 5 variables, $\theta_0, \theta_1, \dots, \theta_4$. The Lagrangian $\mathcal{L} = \mathcal{T} - \mathcal{V} = 0.5 \sum \left(m_i v_i^T v_i + J_i \omega_i^T \omega_i \right) - \sum \left(m_i g y_i \right)$ where v_i, ω_i, y_i are the linear velocity, angular velocity, and y-position center of mass of link i respectively. We take the summation over all the 5 links. Using the Euler-Lagrange equations gives 7 equations

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{N}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{B}\mathbf{u} + \mathbf{J}_{C_1}\mathbf{P}_{C_1} \quad (3.1)$$

where $\mathbf{M}, \mathbf{N}, \mathbf{B}$ are the mass matrix, accelerations due to Coriolis, centrifugal acceleration and gravity, and torque selection matrices. The control torques are $\mathbf{u} = \begin{bmatrix} \tau_1 & \tau_2 & \tau_3 & \tau_4 \end{bmatrix}^T$ where τ_i

is the torque for joint with stance calf link θ_i , \mathbf{J}_{C_1} is the Jacobian of the contact point C_1 and \mathbf{P}_{C_1} is the ground reaction force on the stance leg.

Without loss of generality, we can assume $x = y = 0$. Also, since C_1 is at rest, $\dot{x} = \dot{y} = \ddot{x} = \ddot{y} = 0$. Using these conditions, we use the first two equations in Equation 3.1 to find the ground reaction forces \mathbf{P}_{C_1} as a function of joint angles, velocities, and acceleration. We may write the remaining 5 equations as

$$\mathbf{M}_\theta(\theta)\ddot{\theta} + \mathbf{N}_\theta(\theta, \dot{\theta}) = \mathbf{B}_\theta \mathbf{u} \quad (3.2)$$

where \mathbf{M}_θ , \mathbf{N}_θ , \mathbf{B}_θ are appropriately versions of the matrices defined earlier. We use this equation for simulating single stance phase and for controller development later.

3.3.2 Foot-strike equations

When the swing foot C_2 touches the ground, the single stance phase ends and the robot transitions to an instantaneous foot-strike phase. We assume that the trailing leg applies an impulsive force along the stance leg, $\mathbf{I}_{C_1} = I \begin{bmatrix} -\sin(\theta_0 + \theta_1 + \theta_3) & \cos(\theta_0 + \theta_1 + \theta_3) \end{bmatrix}^T$ where I is the scalar impulse. This force comes from the ankle motor at C_1 which is passive during the stance phase, except during the foot-strike phase. Our choice of impulsive push-off is to be able to achieve energy-efficient walking compared to hip actuation (see [30]). In this phase, angular

momentum is conserved about new contact point C_2 . We obtain the equations for this phase by integrating Eqn Equation 3.1 and taking the limit as time goes to 0

$$\begin{bmatrix} \mathbf{M}(\mathbf{q}^-) & -\mathbf{J}_{C_2}^T \\ \mathbf{J}_{C_2} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}}^+ \\ \mathbf{I}_{C_2} \end{bmatrix} = \begin{bmatrix} \mathbf{M}(\mathbf{q}^-)\dot{\mathbf{q}}^- + \mathbf{J}_{C_1}^T \mathbf{I}_{C_1} \\ \mathbf{0} \end{bmatrix} \quad (3.3)$$

where the superscript $-$ and $+$ denote the instance before and after collision respectively.

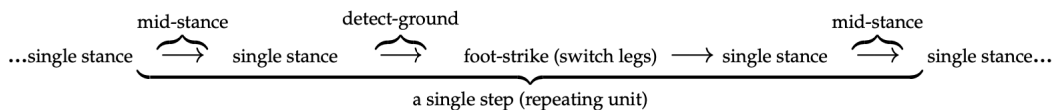


Figure 11: Pictorial representation of a single step

3.3.3 Simulating a single step

Figure Figure 11 shows the general equation that describes a single step, the repeating unit, that starts and ends at mid-stance. We now explain the composition of a single step. We start the step at mid-stance when stance leg thigh link is vertical, $\theta_0 + \theta_1 = 0$. Next, we use the single stance Equation 3.2 to integrate the system till foot-strike. The foot strike occurs when the swing foot C_2 touches the ground, $y_{C_2} = \ell_1 \cos(\theta_0 + \theta_1) - \ell_1 \cos(\theta_0 + \theta_2) + \ell_2 \cos(\theta_0 + \theta_1 + \theta_3) - \ell_2 \cos(\theta_0 + \theta_2 + \theta_4) = 0$. Next, we apply the foot-strike condition given by Equation 3.3. Then we swap the legs, $\theta_0^+ = \theta_0^-$, $\theta_1^+ = \theta_2^-$, $\theta_2^+ = \theta_1^-$, $\theta_3^+ = \theta_4^-$, $\theta_4^+ = \theta_3^-$. Similarly, for the angular velocities we have $\dot{\theta}_0^+ = \dot{\theta}_0^-$, $\dot{\theta}_1^+ = \dot{\theta}_2^-$, $\dot{\theta}_2^+ = \dot{\theta}_1^-$, $\dot{\theta}_3^+ = \dot{\theta}_4^-$, $\dot{\theta}_4^+ = \dot{\theta}_3^-$. Finally, we integrate the equations in single stance given by Equation 3.2 till the next mid-stance given by $\theta_0 + \theta_1 = 0$.

3.4 Methods

3.4.1 Partial feedback linearization

We use PFL to control the actuated degrees of freedom in the stance phase as described next. We invert the mass matrix from Equation 3.2 to get

$$\ddot{\theta} = \mathbf{M}_\theta^{-1}(\theta)(\mathbf{B}_\theta \mathbf{u} - \mathbf{N}_\theta(\theta, \dot{\theta})) \quad (3.4)$$

The system has 5 degrees of freedom, but only 4 actuators. We use PFL to decouple the 4 degrees of freedom, namely the torso θ_0 , the swing leg joints θ_2 and θ_4 , and the stance leg knee θ_3 . Thus, if $\theta_c = \begin{bmatrix} \theta_0 & \theta_2 & \theta_3 & \theta_4 \end{bmatrix}$. Then, we can find a matrix \mathbf{S}_c by inspection such that $\theta_c = \mathbf{S}_c \theta$ where $\theta = \begin{bmatrix} \theta_0 & \theta_1 & \theta_2 & \theta_3 & \theta_4 \end{bmatrix}$. We write

$$\ddot{\theta}_c = \mathbf{S}_c \ddot{\theta} = \mathbf{S}_c \mathbf{M}_\theta^{-1}(\theta)(\mathbf{B}_\theta \mathbf{u} - \mathbf{N}_\theta(\theta, \dot{\theta})) = \mathbf{v} \quad (3.5)$$

where \mathbf{v} is the new control input chosen as

$$\mathbf{v} = \ddot{\theta}_c^{ref} + \mathbf{K}_d(\dot{\theta}_c^{ref} - \dot{\theta}_c) + \mathbf{K}_p(\theta_c^{ref} - \theta_c) \quad (3.6)$$

where θ_c^{ref} , $\dot{\theta}_c^{ref}$, $\ddot{\theta}_c^{ref}$ are the user specified reference position, velocity, and acceleration. We assume a fifth order polynomial for θ_c^{ref} such that the position, velocity, and acceleration at the start and end are specified, many of which are set to 0. The gains \mathbf{K}_p and \mathbf{K}_d are diagonal

matrices. We choose $\mathbf{K}_p = K_p \text{diag}\{1, 1, 1, 1\}$ and $\mathbf{K}_d = 2\sqrt{K_p} \text{diag}\{1, 1, 1, 1\}$ to ensure critical damping. The motor torques are

$$\mathbf{u} = (\mathbf{S}_c \mathbf{M}_\theta^{-1}(\theta) \mathbf{B}_\theta)^{-1} (\mathbf{v} + \mathbf{S}_c \mathbf{M}_\theta^{-1}(\theta) (\mathbf{N}_\theta(\theta, \dot{\theta})) \quad (3.7)$$

The uncontrolled degree of freedom is $\theta_u = \mathbf{S}_u \theta$ where $\theta_u = \theta_1$. We can write an equation for this degree of freedom after suitably including the control input from the above equation

$$\ddot{\theta}_u = \mathbf{S}_u \ddot{\theta} = \mathbf{S}_u \mathbf{M}_\theta^{-1}(\theta) (\mathbf{B}_\theta \mathbf{u} - \mathbf{N}_\theta(\theta, \dot{\theta})) \quad (3.8)$$

We need to integrate this equation in the single stance phase.

3.4.2 Step-to-step dynamics: Poincaré map

We now introduce the idea of Poincaré section and map (see [19] for more details). The Poincaré section is a $2N - 1$ (where N is the total degrees of freedom of the system) dimensional surface denoting an instance in the locomotion cycle (e.g., mid-stance, foot-strike) as shown in blue dots in Figure 9 (b). The Poincaré map is a function \mathbf{F} that maps an initial state at the Poincaré section Θ^i and controls during the step \mathbf{U}^i to the state at the Poincaré section at the next step Θ^{i+1} . This map \mathbf{F} describes the step-to-step dynamics and is found by integrating equations from Poincaré section to the next as shown in Figure 11. Thus, we can write

$$\Theta^{i+1} = \mathbf{F}(\Theta^i, \mathbf{U}^i) \quad (3.9)$$

where i is the step number; $\Theta = \begin{bmatrix} \theta & \dot{\theta} \end{bmatrix}$ is the state, where $\theta = \begin{bmatrix} \theta_0 & \theta_1 & \theta_2 & \theta_3 & \theta_4 & \theta_5 \end{bmatrix}$, \mathbf{U} are the discrete controls that are set once per step (e.g., foot placement angle, impulsive push-off), and \mathbf{F} is the Poincaré map that relates the state from one mid-stance to the next one. For most systems, it is not possible to find an analytical formula for the Poincaré map. It is obtained by numerically integrating the equations of motion and/or applying the algebraic conditions for instantaneous phases (e.g., footstrike). Note that we define the mid-stance as $\theta_0 + \theta_1 = 0$. For a 10 degree of freedom system, the Poincaré map is 9 dimensional.

We can simplify Equation 3.9 as follows. Assuming that the PFL works as intended, the step-to-step dynamics depends only on the uncontrolled degrees of freedom, $\Theta = \begin{bmatrix} \theta_u & \dot{\theta}_u \end{bmatrix}$ (Note, we show this in the results section). Thus, for the 5-link biped, we have $\theta_u = \theta_1$. But, since the Poincaré map is at the mid-stance, there is only one degree of freedom, $\dot{\theta}_1$. We choose two controls to be the step angle at foot-strike $\theta_2 = \alpha$ and the push-off impulse I at footstrike. Thus, we write

$${}^m\dot{\theta}_1^{i+1} = F({}^m\dot{\theta}_1^i, \alpha^i, I^i) \quad (3.10)$$

where ${}^m\dot{\theta}_1$ is the mid-stance speed of θ_1 . Also, note that F is a scalar.

3.4.3 Approximating the Poincaré map

The Poincaré map, Equation 3.10, reduces to a single equation. One caveat is that it is not possible to find an analytical solution to the step-to-step dynamics. Our goal is to find a simple approximation \bar{F} with ${}^m\dot{\theta}_1^{i+1}$ being the approximated mid-stance speed at step $i + 1$

$${}^m\dot{\theta}_1^{i+1} = \bar{F}({}^m\dot{\theta}_1^i, \alpha^i, I^i) \quad (3.11)$$

3.4.3.1 Data generation

First, we prepare the simulator to simulate a single step as shown in Figure 11. The inputs are the mid-stance speed ${}^m\dot{\theta}_1^i$, the foot placement angle α^i , and the push-off impulse I^i and the output is the mid-stance speed at the next step ${}^m\dot{\theta}_1^{i+1}$. For some inputs, the reaction would cause the model to lose balance and fall before taking the next step (infeasible) while for other inputs it would successfully reach the next step (feasible). The resulting feasible/infeasible dataset is used to find the boundary of the region using support vector classification and the feasible dataset is used to find a quadratic polynomial model for the 2-BVP. These are discussed next.

3.4.3.2 Estimating the boundary of the Poincaré map

A SVM classification model is used to estimate the boundary of the Poincaré map. The SVM binary classification algorithm from the MATLAB statistics and machine learning toolbox with a linear kernel function is used to search for an optimal hyperplane that splits the data into two classes. This allows filtering of the feasible and infeasible data. We assume that the data is linearly separable. We generate a training set of 1000 combinations of $\dot{\theta}_1$, α^i and I^i to

train the SVM classifier. We show the decision boundary plane equation in Eqn. Equation 3.12 where w_0 and b_0 are the coefficients of the hyperplane equation. In Eqn. Equation 3.13 A^i is the Lagrange multiplier obtained from the dual cost function used in the SVM training, x^i are the support vectors, y^i is the classification of the support vectors (-1 or $+1$) and j is the number of support vectors. Eqn Equation 4.29 shows the SVM classifier equations. We used a test set of 125 samples for testing.

$$m\hat{\theta}_1^i = \frac{-b_0 - w_0(1)\alpha^i - w_0(2)I^i}{w_0(3)} \quad (3.12)$$

$$w_0 = \sum_{i=1}^j A^i y^i x^i \quad (3.13)$$

$$b_0 = \frac{1}{j} \sum_{i=1}^j (y^i - w_0 \cdot x) \quad (3.14)$$

$$h(x_i) = \begin{cases} +1 \text{ (not feasible)} & \text{if } w \cdot x + b \geq 0 \\ -1 \text{ (feasible)} & \text{if } w \cdot x + b < 0 \end{cases} \quad (3.15)$$

3.4.3.3 Quadratic polynomial model of the Poincaré map

Once we developed the classification model, we used the feasible samples to develop a polynomial regression model. Our regression model is a quadratic polynomial with regression coefficient denoted by β .

$$\begin{aligned} m\hat{\theta}_1^{i+1} &= \beta_0 + \beta_1^m \hat{\theta}_1^i + \beta_2 \alpha^i + \beta_3 I^i + \beta_4^m \hat{\theta}_1^i \alpha^i \dots \\ &\dots + \beta_5^m \hat{\theta}_1^i I^i + \beta_6 \alpha^i I^i + \beta_7 \alpha^i \alpha^i + \beta_8 I^i I^i \end{aligned} \quad (3.16)$$

3.4.4 Quadratically constrained quadratic program

We formulated an optimization problem to find the inputs that yield the approximated ${}^m\dot{\theta}_1^{i+1}$. The cost function is the squared error of the outputs in relation to their nominal states: $\text{Cost} = ({}^m\dot{\theta}_1^{i+1} - {}^m\bar{\dot{\theta}}_1^{i+1})^2 + (\alpha^i - \bar{\alpha}^i)^2 + (I^i - \bar{I}^i)^2$ where ${}^m\bar{\dot{\theta}}_1^{i+1}$, $\bar{\alpha}^i$ and \bar{I}^i have the nominal values $-0.97, 0.375, 0.18$ respectively. These nominal values were chosen to achieve walking at human speed and step length. The constraints include Eqns. Equation 3.12 - Equation 3.16.

The problem can be rewritten as a quadratically constrained quadratic program as given below

$$\underset{\mathbf{y}}{\text{minimize}} \quad 0.5\mathbf{y}^T\mathbf{H}\mathbf{y} + \mathbf{f}^T\mathbf{y} + c \quad (3.17)$$

$$\text{subject to: } 0.5\mathbf{y}^T\mathbf{Q}_i\mathbf{y} + \mathbf{k}_i^T\mathbf{y} + d_i = 0 \quad (3.18)$$

$$\mathbf{p}^T\mathbf{y} < b_0 \quad (3.19)$$

$$\mathbf{LB} \leq \mathbf{y} \leq \mathbf{UB} \quad (3.20)$$

where $\mathbf{y} = \begin{bmatrix} \Theta^{i+1} & \mathbf{U}^i \end{bmatrix}^T$, \mathbf{f} , \mathbf{H} and c are user chosen constants in the cost function. Eq. Equation 4.31 is the polynomial regression equation as shown in Eq. Equation 3.16 where ${}^m\dot{\theta}_1^i$ is treated as a known constant. The matrix \mathbf{Q}_i contains the second order terms, \mathbf{k}_i contains first order terms and d_i contains the constants. Eq. Equation 4.32 accounts for linear inequality constraints and contains the SVM classifier constraint from Eq. Equation 4.29. \mathbf{LB} and \mathbf{UB} are the lower bound and upper bound vectors set to be $[-2.5, 0.087, 0.1]$ and $[-0.5, 0.873, 0.4]$.

3.5 Results

3.5.1 SVM Classification

Figure 24 shows the feature space of the training set along with the SVM classification optimal hyperplane where the red dots are the feasible samples, and the blue are the infeasible samples. We show the optimal hyperplane in purple which corresponds to the decision boundary for the model. Figure 13 shows the confusion matrix of the SVM model. Overall, the model has an accuracy of 93.6%. As seen from the confusion matrix, the precision of predicting feasible data (-1) is 86.2% and the precision for predicting nonfeasible data (+1) is 95.8%.

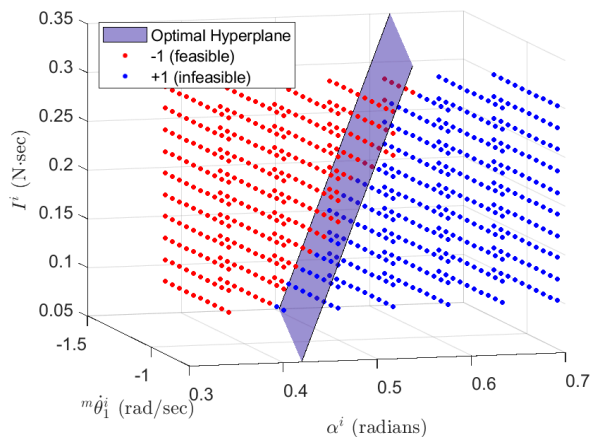


Figure 12: Feature space and optimal hyperplane of test set.

3.5.2 Quadratic polynomial regression

We show the estimated coefficients of the regression with their respective p-values in table Table I. The R-squared value and adjusted R-squared value for the regression analysis was

Output Class	-1	25 20.0%	4 3.2%	86.2% 13.8%
	+1	4 3.2%	92 73.6%	95.8% 4.2%
		86.2% 13.8%	95.8% 4.2%	93.6% 6.4%
		\hat{y}	\hat{x}	Target Class

Figure 13: Classification confusion matrix

	Estimate	p-value
β_0	1.587	<0.01
β_1	0.910	<0.01
β_2	-10.553	<0.01
β_3	1.872	<0.01
β_4	1.429	<0.01
β_5	-2.874	<0.01
β_6	-26.937	<0.01
β_7	25.42	<0.01
β_8	6.116	<0.01

TABLE I: Regression coefficients

0.975 for both. Considering 1 rad/sec to be nominal, 93.1% of the test samples had an accuracy of 90% or better, and 100% of the samples had an accuracy of 80% or better. The mean accuracy of the regression model was 95.61%.

3.5.3 Optimization: Following reference velocity

We tested the model and optimization framework's ability to follow a reference velocity profile $\dot{\theta}_1^{ref}$. We chose a sinusoidal pattern with an amplitude of 0.1 rad/sec for 20 steps with

an offset equal to the nominal velocity of -0.97 rad/sec. The goal of the optimization is to drive the velocity ${}^m\dot{\theta}_1^{i+1}$ to the targeted velocity $\dot{\theta}_1^{ref}$. We modified the cost function such that ${}^m\bar{\dot{\theta}}_1^{i+1} = \dot{\theta}_1^{ref}$. Figure 14 shows the reference velocity and the actual velocity in simulation. Figure 15 shows the behavior of the controls α^i and I^i as the velocity changes. We generate these controls from the optimization problem using the quadratic polynomial model and are inputs to the actual model, Equation 3.10. The mean absolute error was 0.0106 rad/sec. We calculated this by subtracting the reference velocity from the actual at every step and taking the mean. These optimizations took about 15 to 40 function evaluations demonstrating the feasibility for real-time control.

3.5.4 Optimization: Terrain with ditches

We tested the model and optimization framework's ability to plan motion on a terrain with ditches. The swing foot touchdown relative to the stance foot is $2(\ell_1 + \ell_2) \sin 0.5\alpha^i \sim (\ell_1 + \ell_2)\alpha^i$. Thus, swing foot touchdown is a linear constraint that preserves the quadratic form. One issue with this formulation is that we need to solve multiple quadratic programs for the foot step planning one for each feasible foot step location encompassed with ditches on either side. Here the choice of the ditches was such that we only needed to solve two quadratic programs, one for stepping before the ditch and one after the ditch.

After solving both problems, we chose the solution with the lowest cost. Figure 17 shows the velocities ${}^m\dot{\theta}_1^{i+1}$ and ${}^m\dot{\theta}_A^{i+1}$ for every step. ${}^m\dot{\theta}_A^{i+1}$ obtained from the optimization solution and the ${}^m\dot{\theta}_1^{i+1}$ is obtained from the simulator, Eqn Equation 3.10. In the optimization problem the first ${}^m\dot{\theta}_1^i$ input value is the nominal and for every other step the input value is the velocity

${}^m\dot{\theta}_1^{i+1}$ obtained from the Poincaré map in the previous step. The mean absolute error was 0.0077 rad/sec and was calculated by subtracting ${}^m\dot{\theta}_1^{i+1}$ from ${}^m\dot{\theta}_1^{i+1}$ and taking the mean. The largest absolute error was 0.0251 rad/sec. Figure 18 shows the controller deviations from their nominal values. The humanoid from the simulation shown in Figure 16 successfully walked over four ditches without falling. It is clear that the humanoid varies the length of its steps to properly step over the ditches. To walk over the third ditch, two steps were taken. Conversely, to walk over the fourth ditch one long step was enough. The control α was bounded by the boundaries from the training set of the regression model (0.34-0.873 radians). This limited the humanoids ability to take shorter or longer step when necessary. In these simulations, the nonlinear optimization needed 6 to 130 functions evaluations, again quite low demonstrating the feasibility for real-time control. A simulation video is shown in the reference [1].

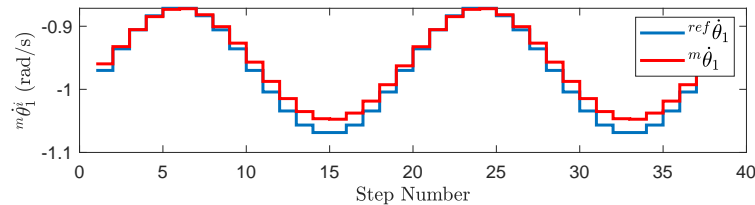


Figure 14: Sinusoidal velocity tracking performance

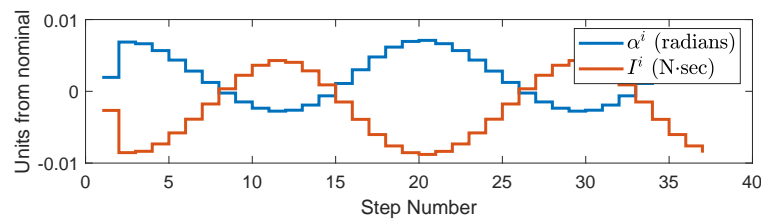


Figure 15: Sinusoidal velocity tracking control inputs

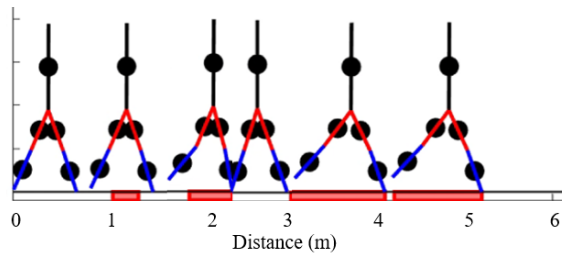


Figure 16: Simulation of humanoid walking over four ditches.

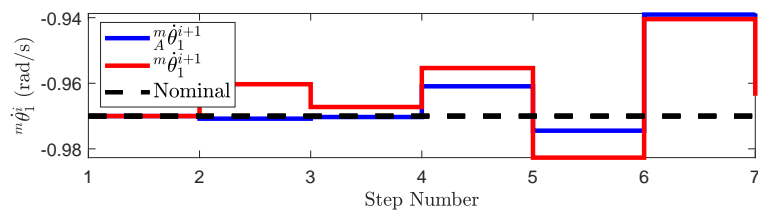


Figure 17: Velocity before and after every step for planning on ditches

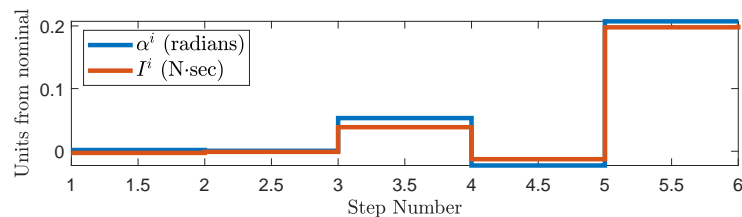


Figure 18: Controls at every step for planning on ditches

3.6 Discussion

Through numerical simulation on a 5-link biped with 10 dimensional state space, we have successfully shown that PFL reduces this to 2 dimensional state space. Furthermore, the Poincaré section reduces this to only a single dimension. Then using data-driven approach first by using extensive simulation for data generation and second by curve fitting using SVM and polynomial regression, we demonstrate a highly accurate approximation of the 2-BVP. This resulted in the

formulation and solution of a suitable quadratically constrained quadratic program that achieves high accuracy and quick solution time.

PFL like the HZD formulation is a model reduction framework. However, HZD requires choosing state based output functions which can be non-intuitive and leads to a time invariant control formulation which does not need an external clock. PFL enables specifying the joint angles/velocity independently of each other, which is more intuitive and leads to a time dependent control formulation and hence needs an external clock for synchronization. Another difference between the two approaches is that HZD is used for local stability or stability in the current step, while in our formulation the PFL with Poincaré map is used for orbital stability or stability in both the current and the future step.

Past approaches for orbital stabilization of legged robots have considered linearized control around the period walking cycle for control, which limits the approach to small disturbances. In contrast, we use data from a relatively large region near the periodic gait for fitting the Poincaré map. This enables the controller to stabilize a relatively wider range of initial conditions. These quadratic approximations of the model and linear approximation of the boundary enables us to formulate a relatively simple quadratically constrained quadratic program that is solved in less than 60 function evaluations, which is promising for real-time control.

The accuracy from the SVM classification model is good enough for our planar humanoid model, but a higher accuracy may be needed for more complex 3D models. Several ways to improve the model are discussed next. The SVM used in this method uses a hard margin and works well with linearly separable data. Using a soft margin SVM might improve the

model by considering nonlinearity. One method of increasing the feasible-predicting precision is by increasing the SVM margin such that the probability of predicting +1 when it is -1 is low. In our approach, the quadratic polynomial regression was used to prevent over-fitting the data. However, a higher order polynomial regression may provide a better fit, but with possibly increased computational time. In addition, using a neural network may yield a similar or better fit. The use of a neural network was shown in [29] to be a more stable control model than a quadratic control model. Future work will investigate the use of neural networks to approximate the Poincaré map. Finally, we limited our planning horizon to a single step look-ahead. However, planning over multiple steps is more desirable for more complex terrain, and this may lead to a combinatorial explosion that would need suitable heuristics to solve quickly.

3.7 Conclusions and Future Work

This paper presented a fast method of computing controllers using a model that enables real-time control of a 5-link planar bipedal robot. PFL was used to transform the complete model to low dimension. Data was generated for Poincaré map. SVM was utilized to obtain a decision hyperplane which separates feasible and infeasible datapoints. A quadratic polynomial regression was used to approximate the Poincaré map leading to a closed form expression for the 2-BVP. Finally, a quadratically constrained quadratic program was formulated and solved. The results demonstrated that the approach was able to follow a reference velocity profile and plan over a terrain with ditches with high accuracy in relatively short amount of time (< 130 function evaluations). To conclude, the approach presented in this paper provides a computationally efficient method of achieving precise control for complex bipedal robots. This work

may ultimately be used to not only control bipedal robots but also lower limb exoskeletons and prosthesis. The evaluation of this approach on a humanoid platform will be performed in future studies. Such platform will be a 3D extension of this work where the humanoid will exhibit 3 uncontrolled degrees of freedom in a 6 dimensional state space with a 6-1 dimension of the Poincaré section.

CHAPTER 4

QUADRATICALLY CONSTRAINED QUADRATIC PROGRAMS USING APPROXIMATIONS OF THE STEP-TO-STEP DYNAMICS: APPLICATION ON A 2D MODEL OF DIGIT

Overview: Bipedal robots are yet to achieve mainstream application because they lack robustness in real-world settings. One of the major control challenges arises due to the ankle motors' limited control authority, which prevents these robots from being fully controllable at a particular instant (e.g., like an inverted pendulum). We show that to stabilize such robots, they must achieve stability over the time scale of a step, also known as step-to-step (S2S) stability. Past approaches have used the linearization of the S2S dynamics to develop controllers, but these have limited regions of validity. Here, we use a data-driven approach to approximate S2S dynamics, including its region of validity. Our results show that linear and quadratic models can approximate the region of validity and S2S dynamics, respectively. We show that the quadratic S2S approximation generated using a data-driven full-body dynamics simulator outperforms those generated using the analytical linear S2S generated from the popularly used linear inverted pendulum model (LIPM). The S2S approximation enables us to formulate and solve a quadratically constrained quadratic program to develop walking controllers. We

Parts of this chapter are taken from the following published journal article:
Hernandez Hinojosa, E., Torres, D., & Bhounsule, P. A. (2022, November). Quadratically constrained quadratic programs using approximations of the step-to-step dynamics: application on a 2D model of Digit. In 2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids) (pp. 96-103). IEEE.

demonstrate the efficacy of the approach in simulation using a 2D model of Digit walking on patterned terrain. A video is linked here: <https://youtu.be/MniABg2jGEA>

4.1 Introduction

Bipedal robots inspired by animal morphology, such as Digit, seek to mimic the balance, walking control, and flexibility displayed by humans. However, such robots are yet to walk robustly to find their way to practical applications. Bipedal robots are challenging to control due to their high dimensionality, nonlinear dynamics, and under-actuation due to springs and limited ankle torques.

Broadly, two control approaches exist that rely on the fact that walking is, in principle, of low dimensions. One uses a template (e.g., linear inverted pendulum model, LIPM [31]) to develop a controller and then transfer this controller to a full robot model using force and position control. The second develops a trajectory tracking controller using a full-order robot model in such a way that the system behaves like a low-order model (e.g., hybrid zero dynamics [24]). The former does not account for angular momentum or center of mass differences from a point-mass model, while the latter has difficulty stabilizing if the system deviates significantly from the reference trajectory.

Our approach takes the best of the two. We use a LIPM template to develop the controller and then use tight trajectory tracking on the full order model to reduce it to low dimensions. To circumvent the issue of significant deviation from the nominal trajectory, we use a data-driven approach to identify the state and control deviations that take the system away from the nominal trajectory over the time scale of a step, also known as the S2S dynamics. We fit a

quadratic regression model to the S2S dynamics and use the model to develop robust controllers. The model is incorporated as a quadratic constraint in a quadratic program which is used to optimize foot placement to walk over obstacles.

4.2 Background and Related Work

Control of bipedal robots with small or point feet is challenging because of under-actuation or the lack of control of all degrees of freedom at all times. Such robots are locally unstable, but with a good control design can be stable over the time scale of a step [5]. This notation of step-to-step (S2S) stability is known as orbital stability [19]. There is evidence that humans use S2S stability to balance while walking and running [10].

Some extreme examples of S2S stability are passive dynamic walkers [6]. Here there is no control, and the robot moves down a ramp relying on its natural dynamics [32]. However, such robots are limited to walking down ramps, usually at low speeds, and are knocked out by the slightest disturbances or terrain variation. Adding actuators and actively regulating the torque/force to add just enough energy to overcome friction and energy losses due to foot-strike enables these robots to walk on level ground [13].

To demonstrate S2S or orbital stability, a Poincaré map is used [19]. A Poincaré map relates how the system's state maps from one step to the next. If the eigenvalues of the linearization of the Poincaré map are all less than 1 then the system is said to be S2S or orbitally stable. When control is added, the linearization of the Poincaré map is a function of the state and control. By choosing the control, the eigenvalues can be manipulated to be less than 1 [14].

The Hybrid Zero Dynamics takes a different approach toward S2S stability [26, 33]. Here, virtual constraints are used where the controlled degrees of freedom are tracked as a function of the uncontrolled degree of freedom. This reduces the Poincaré map to a lower dimension (1 for 2D walking and 5 for 3D walking) but does not guarantee S2S stability. To achieve the S2S stability, one could: find controls such that the eigenvalues are less than 1, change the virtual constraints or add event-based stabilization [27]. However, note that all these methods enable linearized stability, which may be lost for big perturbations.

Since bipedal locomotion is fundamentally a low-dimensional problem, control methods start with an idealized model such as the linear inverted pendulum for walking or the spring-loaded inverted pendulum for running [34]. Then heuristics such as the capture point, which is the location where the robot needs to step to come to a complete stop, may be used to develop control strategies [35]. One of the limitations of such simple models is that they do not capture nonlinearities and changes in angular momentum and may lead to conservative walking when transferred to the full body model or hardware.

Our past work on the spring-loaded inverted pendulum has shown that control based on the nonlinear approximation of the S2S map enables a broader range of perturbations [29]. Then using a 2D humanoid model and approximating the S2S dynamics using polynomials and the region of validity using support vector machines, we were able to demonstrate walking on patterned terrain [36]. Interestingly, it has been shown that approximations to the linear inverted pendulum model (LIPM) can also generate accurate S2S maps for control of the biped Cassie [37]. In this paper, we extend our previous work to the humanoid Digit. Unlike Cassie,

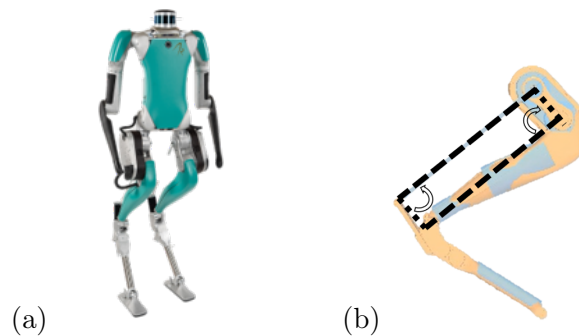


Figure 19: (a) **Robot model:** Digit bipedal robot with 30 degrees of freedom and 20 actuated joints. (b) A parallelogram closed chain was used to simplify the 6-bar linkage of the leg.

Digit has an upper body with a significant mass. We demonstrate that using the LIPM-based S2S map generates poor control compared to using a full nonlinear dynamics-based S2S map. Our contributions are: (1) a method to transfer LIPM models to full body model; (2) use of data-driven approaches to learn the S2S map, including the region of feasibility; and (3) a quadratically constrained quadratic program to plan walking on patterned terrain.

4.3 Models

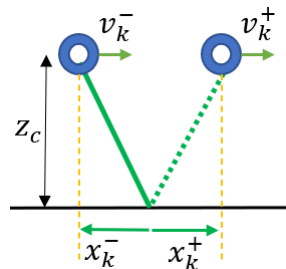
4.3.1 Robot model

Digit is a 30-degree of freedom bipedal robot with 20 actuated joints shown in Figure 19(a) (see [38] for more details). Digit’s legs are similar to its predecessor Cassie [8], but have an additional roll joint at the toe and an upper body consisting of a torso and two arms. The robot kinematics of the legs consists of two closed-loop chains, modeled as distance constraints between the heel spring and the hip pitch joints. For simplification, the knee and tarsus joint position are inversely coupled during the swing leg phase. This simplification arises because

when the heel spring and shin springs are uncompressed, the chain is a parallelogram with two pairs of equal sides as shown in Figure 19(b).

4.3.2 S2S Models

The general form of the S2S models developed in this study is illustrated in Figure 32 and the general S2S map is shown in (Equation 5.1). At the start of the single support phase (SSP), the position and velocity of the center of mass (COM) with respect to the stance foot are x_k^- and v_k^- , respectively, where k is the step number. At the end of the current SSP, the states become x_k^+ and v_k^+ . The S2S dynamics model, ζ , maps the states $\mathbf{s}_k^- = \{\mathbf{x}_k^-, \mathbf{v}_k^-\}$ to $\mathbf{s}_k^+ = \{\mathbf{x}_k^+, \mathbf{v}_k^+\}$.



$$\mathbf{s}_k^+ = \zeta(\mathbf{s}_k^-, \tau_p) \quad (4.1)$$

Figure 20: The general S2S model.

4.3.2.1 Analytical map from Linear Inverted Pendulum Model (LIPM)

In the LIPM the COM height is kept approximately constant using the hip, knee and ankle joints. The equations of the LIPM are linear

$$\ddot{x} = \frac{g}{z_c}x + \frac{1}{mz_c}\tau_p; \quad \ddot{y} = \frac{g}{z_c}y + \frac{1}{mz_c}\tau_r \quad (4.2)$$

where g is gravity, z_c is the constant height of the COM, m is the mass, and τ_p and τ_r are input pitch and roll torques, respectively. In the 2D case, assuming no input torques, the equations about the x -axis can be analytically integrated from 0 to time T_s to get

$$\zeta(x_k^-, v_k^-) = \begin{bmatrix} x_k^{L+} \\ v_k^{L+} \end{bmatrix} = \begin{bmatrix} C_T \cdot x_k^- + T_c S_T \cdot v_k^- \\ S_T/T_c \cdot x_k^- + C_T \cdot v_k^- \end{bmatrix} \quad (4.3)$$

which fits the form of (Equation 5.1) where $S_T = \sinh(T_s/T_c)$, $C_T = \cosh(T_s/T_c)$, and $T_c = \sqrt{z_c/g}$. The state superscripts L and N indicate that the states belong to the LIPM or regression model, respectively.

4.3.2.2 Regression-based map from MuJoCo simulator

A data-derived polynomial regression model was formulated to better capture the S2S dynamics of the robot. The LIPM was used as a starting point to develop a controller for the nonlinear model. The map given by (Equation 5.1) was obtained by numerically simulating the system for different initial conditions and controls. This data was then curve fitted using regression analysis. More details are in the Sec. 4.7. Similar to the LIPM, this model assumes a constant COM height z_c and a constant SSP time T_s of 0.35 seconds. The polynomial S2S map is shown in (Equation 5.7) and (Equation 5.8); which also take the form of

$$\zeta(x_k^-, v_k^-) = \begin{bmatrix} f(x_k^-, v_k^-), g(x_k^-, v_k^-) \end{bmatrix}'$$

$$f(x_k^-, v_k^-) = x_k^{N+} = \alpha_0 + \alpha_1 x_k^- + \alpha_2 v_k^- + \alpha_3 x_k^- v_k^- (x_k^-)^2 + \alpha_5 (v_k^-)^2 + \alpha_4 \quad (4.4)$$

$$g(x_k^-, v_k^-) = v_k^{N+} = \beta_0 + \beta_1 x_k^- + \beta_2 v_k^- + \beta_3 x_k^- v_k^- + \beta_4 (x_k^-)^2 + \beta_5 (v_k^-)^2 \quad (4.5)$$

4.4 Stepping Controllers

The S2S map from the analytical LIPM and regression from nonlinear MuJoCo simulator were used to develop three stepping controllers for stable walking. We assumed that there is no double support phase during walking. The spring constants of the heel and shin were increased to have minimal spring deflection during foot-ground contact. We also assumed that the COM is located at the hip because the arms are fixed and the torso is kept vertically upright. The knee joint of the stance leg is controlled to ensure the position of the hip is at a desired constant height z_c .

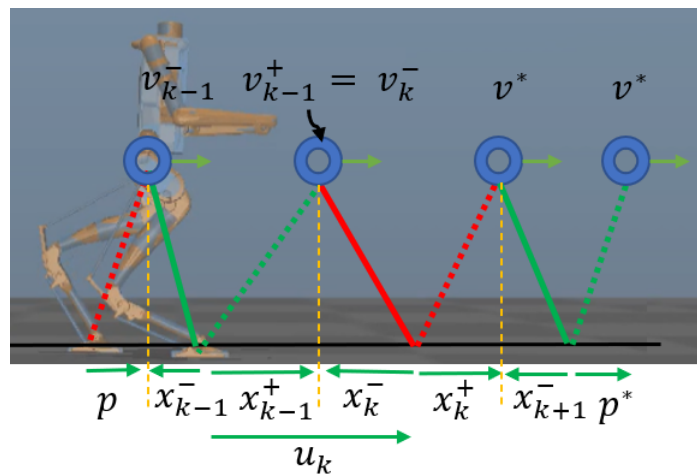


Figure 21: Model-based stepping. The stepping controller dictates the step size u_k to achieve desired velocity v^* .

4.4.1 Model-based Stepping

For this section we will refer to Figure 35. The current state of the model is (x_{k-1}^-, v_{k-1}^-) . The stepping controller is the step length u_k required to achieve the velocity v^* at v_k^+ . Therefore the S2S dynamics map can be expressed in terms of the stepping controller

$$\mathbf{s}_k^+ = \zeta(\mathbf{x}_{k-1}^+ - \mathbf{x}_k^-, \mathbf{v}_k^-) = \zeta(\mathbf{u}_k, \mathbf{v}_k^-) \quad (4.6)$$

The state x_{k-1}^+ is uncontrollable because the robot is underactuated at the toe pitch joint. Therefore the only controllable parameter at every step is the state x_k^- . The states (x_{k-1}^+, v_{k-1}^+) are unknown at the current robot state. One way of proceeding with this controller is by predicting the states using the S2S dynamics maps ($[\tilde{x}_{k-1}^+, \tilde{v}_k^-] = \zeta(x_{k-1}^-, v_{k-1}^-)$) from (Equation 5.3) for the LIPM-based analytical model and (Equation 5.7) and (Equation 5.8) for the simulator-based regression model. Note the tilde over the states denotes a predicted state.

The next step is solving for x_k^- to achieve the velocity v^* at v_k^+ . We can rearrange the S2S equations into (Equation 4.7) to solve for x_k^- using the LIPM.

$$x_k^- = (v_k^+ - C_T \tilde{v}_k^-) T_c / S_T \quad (4.7)$$

Similarly, we can rearrange the S2S equations into a quadratic equation to solve for x_k^- using the regression model

$$x_k^- = -A + 0.5\beta_4(A^2 - 4(\beta_4)B)^{0.5} \quad (4.8)$$

where $A = \beta_1 + \beta_3 \tilde{v}_k^-$ and $B = \beta_5 (\tilde{v}_k^-)^2 + \beta_2 \tilde{v}_k^- + \beta_0 - v_k^+$. The stepping controllers are u_k^L for the LIPM and u_k^N for the regression model

$$u_k^L = \tilde{x}_{k-1}^+ - x_k^- = C_T \cdot x_{k-1}^- + T_c S_T v_{k-1}^- - x_k^- \quad (4.9)$$

$$u_k^N = \tilde{x}_{k-1}^+ - x_k^- = f(x_{k-1}^-, v_{k-1}^-) - x_k^- \quad (4.10)$$

Cyclic States

A cyclic gait is when the state at an instant maps back to itself after a step. The cyclic state x^{*-} and velocity v^* may be solved by setting $v_k^+ = v_k^- = v^*$ and solving for $x_k^- = x^{*-}$ in (Equation 5.3) for LIPM-based stepping and (Equation 5.8) for simulator-based stepping.

4.4.2 Model-based Stepping with Feedback

We assume that the model S2S dynamics are different from the robot dynamics by an additive state error term $m\omega$

$$\zeta_R = \zeta(x_k^-, v_k^-) + m\omega \quad (4.11)$$

where ζ_R is the robot S2S dynamics. The predicted states $(\tilde{x}_{k-1}^+, \tilde{v}_{k-1}^+)$ will likely not equal the actual states (x_{k-1}^+, v_{k-1}^+) .

4.4.2.1 Analytical, LIPM based

To add robustness to the stepping controller a feedback term can be implemented with gains K properly tuned to drive the error S2S dynamics between the model and the robot model to zero

$$u^R = u_k^L + \mathbf{K}(\mathbf{s}(\mathbf{t})^{\mathbf{R}} - \mathbf{s}_{\mathbf{k}}^{\mathbf{L}}) \quad (4.12)$$

where u^R is the step size realized on the robot, $\mathbf{s}^{\mathbf{R}}$ is the current state vector of the robot and $\mathbf{s}_{\mathbf{k}}^{\mathbf{L}}$ is the state vector of the LIPM-predicted states x_{k-1}^+ and v_{k-1}^+ . Stable values for K were found as in [9].

4.4.2.2 Regression, simulator based

Similarly, for regression-based stepping, we can add a feedback term with gains K .

$$u^R = u_k^N + \mathbf{K}(\mathbf{s}(\mathbf{t})^{\mathbf{R}} - \mathbf{s}_{\mathbf{k}}^{\mathbf{N}}) \quad (4.13)$$

(Equation 4.13) is a discrete nonlinear controller which can be linearized about an equilibrium point. Using (Equation 5.7) and (Equation 5.8), the error S2S dynamics becomes (Equation 4.14).

$$\mathbf{e}_{\mathbf{k}+1} = \begin{bmatrix} f\left(\left(x_k^{+R} - x_k^{+N}\right) - \left(u_k^R - u_k^N\right), \left(v_k^{+R} - v_k^{+N}\right)\right) \\ g\left(\left(x_k^{+R} - x_k^{+N}\right) - \left(u_k^R - u_k^N\right), \left(v_k^{+N} - v_k^{+R}\right)\right) \end{bmatrix} = \begin{bmatrix} f\left(\left(e_k^x\right) - \left(K\left(\mathbf{e}_{\mathbf{k}}\right)\right), \left(e_k^v\right)\right) \\ g\left(\left(e_k^x\right) - \left(K\left(\mathbf{e}_{\mathbf{k}}\right)\right), \left(e_k^v\right)\right) \end{bmatrix} \quad (4.14)$$

An equilibrium $\mathbf{e}_{\mathbf{k}+1} = \mathbf{e}_{\mathbf{k}}$ exists at $(e^{x^*}, e^{v^*}) = (0, 0)$. Linearizing the error dynamics near the equilibrium point yields

$$\vec{\mathbf{q}}_{k+1} = \mathbf{J}\vec{\mathbf{q}}_k \quad (4.15)$$

where \mathbf{J} is the Jacobian of the system, $q_k = (e_k^x - e^{x^*})$, $r_k = (e_k^v - e^{v^*})$, and $\vec{\mathbf{q}}_k = \begin{bmatrix} q_k \\ r_k \end{bmatrix}$. The system is now in the form $x_{k+1} = \mathbf{A}x_k$. \mathbf{K} can then be solved for such that all the eigenvalues (λ) of \mathbf{J} evaluated at the equilibrium point are $|\lambda| < 1$ making the system asymptotically stable. The block diagram of the model-based controller is shown in Figure 22.

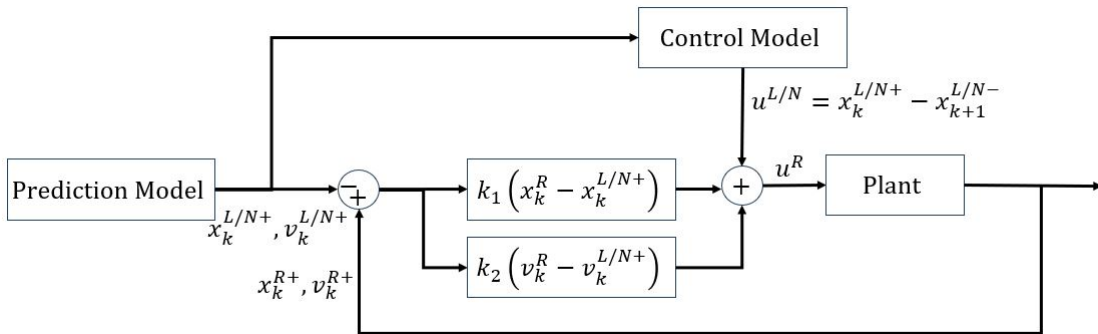


Figure 22: Model-based feedback controller block diagram.

4.4.3 Footstrike-corrected Stepping

Although the stepping controllers shown in (Equation 4.12) and (Equation 4.13) achieve stable walking patterns when properly tuned, there are two terms that accumulate errors. The first term is in the approximated states \tilde{x}_{k-1}^+ and \tilde{v}_{k-1}^+ and the second term is in the S2S dynamics error of states x_k^+ and v_k^+ . We introduce a footstrike-corrected stepping controller, (Equation 4.16), and a footstrike-corrected stepping controller with feedback, (Equation 5.12), to improve the regression-based S2S approximation.

$$u_k'^N = x(t)_{k-1}^{+R} - \tilde{x}_k^- \quad (4.16)$$

$$u^R = u_k'^N + \mathbf{K} (\mathbf{s}(t)_k^R - \mathbf{s}_k^N) \quad (4.17)$$

In this approach, the step size $u_k'^N$ is updated at every time step using the state x_{k-1}^{+R} of the robot.

This updated state is then used to solve for \tilde{x}_k^- such that $v_k^+ = v^*$ is achieved. (Equation 4.8)

is used to solve for \tilde{x}_k^- given v_k^+ and v_k^- . State \tilde{x}_k^- can be expressed in terms of x_k^{-N} plus an error term c_1 that arises from the differences between x_{k-1}^{+R} and x_{k-1}^{+N} .

$$\tilde{x}_k^- = c_1 + x_k^{-N} \quad (4.18)$$

The value of c_1 will be small for models where x_{k-1}^{+R} is close to x_{k-1}^{+N} . Solving for $x_k^{-R} - x_k^{-N}$ yields (Equation 4.21)

$$x(t)_{k-1}^{+R} - x_k^{-R} = x(t)_{k-1}^{+R} - \tilde{x}_k^- + \mathbf{K} (\mathbf{s}(t)_k^{\mathbf{R}} - \mathbf{s}_k^{\mathbf{N}}) \quad (4.19)$$

$$\tilde{x}_k^- - x_k^{-R} = \mathbf{K} (\mathbf{s}(t)_k^{\mathbf{R}} - \mathbf{s}_k^{\mathbf{N}}) \quad (4.20)$$

$$x_k^{-R} - x_k^{-N} = c_1 + \mathbf{K} (\mathbf{s}_k^{\mathbf{N}} - \mathbf{s}(t)_k^{\mathbf{R}}) \quad (4.21)$$

which is used to solve for the error S2S dynamics.

$$\mathbf{e}_{k+1} = \begin{bmatrix} f\left(\left(x_k^{-R} - x_k^{-N}\right), \left(v_k^{-R} - v_k^{-N}\right)\right) \\ g\left(\left(x_k^{-R} - x_k^{-N}\right), \left(v_k^{-R} - v_k^{-N}\right)\right) \end{bmatrix} = \begin{bmatrix} f\left(\left(\mathbf{K}(\mathbf{e}_k) - c_1\right), (e_k^v)\right) \\ g\left(\left(\mathbf{K}(\mathbf{e}_k) - c_1\right), (e_k^v)\right) \end{bmatrix} \quad (4.22)$$

The nonlinear controller can be linearized about the equilibrium point and gains \mathbf{K} can be solved for such that for small values of c_1 the system is stable. The block diagram of the footstrike-corrected controller is shown in Figure 23.

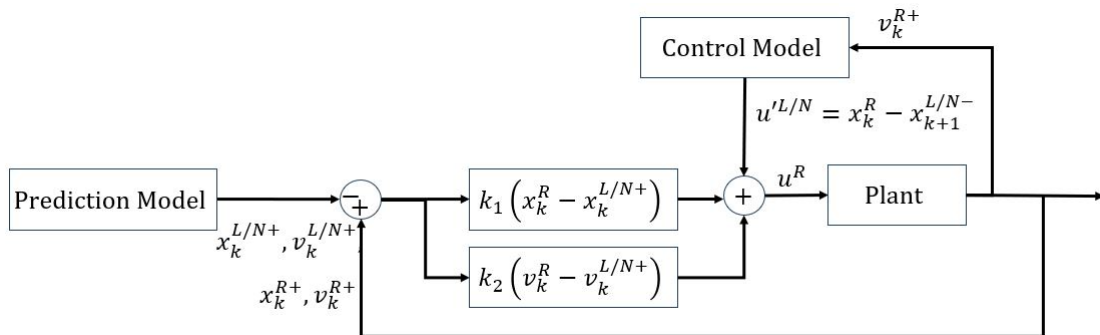


Figure 23: Footstrike-corrected feedback controller block diagram.

4.5 The Linear Inverted Pendulum Model

The phase portrait of the linear inverted pendulum model (LIPM) can be The lateral stepping controller used in this study uses LIPM-based stepping. For side-stepping, the LIPM limit cycle consists of two phases, where the center of mass shifts to one side and then back to the other side

$$y_s = y_R^+ - y_d^- + K_1 (y_{LIP}^+ - y_R^+) + K_2 (v_{LIP}^+ - v_R^+)$$

4.6 Joint-level Control

The joint-level controllers consist of a gravity compensation feed-forward controller and a PD (proportional-derivative) feedback controller for position and velocity control of the leg actuators. The PD controller drives the actuators to the desired joint trajectory. The joint trajectories of the swing leg are generated by running inverse kinematics on the Cartesian space foot trajectory. The stance leg uses the knee actuator to maintain a desired constant height and the hip pitch and roll actuators to keep the torso upright. The toe actuators of the stance

foot are given zero torque and the toe actuators of the swing foot keep the foot parallel to the ground. The arm actuators are kept at a fixed configuration.

4.6.1 Gravity Compensation

Gravity compensation at the joints is achieved by modeling the robot with a floating base. The equations of the robot's dynamics, and holonomic force constraints are over-defined because they include dependent degrees of freedom due to the closed-loop chain. F_h are the holonomic forces and F_G are the ground reaction forces.

$$M(q)\ddot{q} + C(q, \dot{q}) + G(q) = B\tau_{\theta,s} + J_G^T F_G + J_h^T F_h \quad (4.23)$$

$$J_h \ddot{q} + \dot{J}_h \dot{q} = 0 \quad (4.24)$$

The holonomic constraints include a distance constraint on the closed chain of each leg and a position constraint on the stance foot. For simplicity, the toe rod closed chains are modeled as an open chain with actuation at the toe pitch and roll joints. The open-chain dynamics of the legs are projected onto the constrained dynamics to yield the system in minimal coordinates (see [39] for more details). Once the system is in minimal coordinates, we assume the floating base's six virtual joints provide no force or torque, and the shin and heel spring joint torques are calculated using their respective spring constants K_s and deflection x_s . The leg motor torques and constraint forces are solved for using inverse dynamics on the gravity terms similar to what was done in [9].

4.6.2 Joint Trajectory and PD control

The biped is modeled as a compass walker with a point mass at the COM and point feet. Inverse kinematics is done on the foot position for proper foot placement. During the stance phase, the toe motors are given zero torque to allow the toe joints to rotate: allowing an inverted pendulum-like behavior. The knee is used to maintain an approximately constant desired COM height, and the hip pitch joint of the stance leg is used to keep the torso vertically upright. In the 2D case, the position of the hip roll and yaw joints are fixed.

A position and velocity trajectory of the swing foot is generated using a 5th-order polynomial. In the vertical direction, the trajectory is split into two such that the foot moves up to a specified step height and down to the floor in T_s seconds. The horizontal trajectories are model-dependent and are expressed next.

4.6.2.1 Model-based stepping

When using model-based stepping, a trajectory is generated at every footstrike event. The final target velocity is zero, and the targeted final position along the x-axis is dictated by the stepping controller output $u_k^{L/N}$.

$$x(t)_{k+1}^{foot} = \text{traj}\left(u_k^{L/N}\right) \quad (4.25)$$

4.6.2.2 Model-based stepping with feedback

Two trajectories are generated in the x direction when using a model-based stepping controller with feedback. The first trajectory is generated at every footstrike event using model-

based stepping like (Equation 4.25). The second trajectory is updated at every time step; where it begins at the starting position of the foot and finishes at the step length dictated by feedback stepping u^R . Both polynomial trajectory equations are summed with time-dependent weights such that the first trajectory has more weight at the beginning of the step and the second trajectory has more weight at the end of the step.

$$x(t)_{k+1}^{foot} = \left(1 - \frac{t}{T_s}\right) \text{traj}\left(u_k^{L/N}\right) + \left(\frac{t}{T_s}\right) \text{traj}\left(u^R\right) \quad (4.26)$$

Newton's method is used to solve the inverse kinematics of the foot trajectory during the swing phase. The shin and heel springs are assumed to be stiff and have zero deflection simplifying the closed chain to a parallel four-bar linkage as shown in Figure 19(b). Similarly, body Jacobians are used to compute the velocity. The joint-level controller is

$$\tau_m = \tau_g + K_p(\theta_{des} - \theta) + K_d(\dot{\theta}_{des} - \dot{\theta}) \quad (4.27)$$

where τ_m is the motor torque, τ_g is the gravity compensation torque, θ_{des} and $\dot{\theta}_{des}$ are the desired joint position and velocity, respectively.

4.7 Methods

4.7.1 Data Collection using simulator

MuJoCo is an advanced simulator for multi-body dynamics and contacts. MuJoCo was used to run 2D forward walking simulations with different walking parameters and collect S2S data. LIPM-based stepping was applied so the robot could achieve several steps without falling.

x_k^- , v_k^- , and v^* were varied between trials and for every step taken x_k^- , v_k^- , x_k^+ , and v_{k+1}^- were stored parameters. The stepping controller, (Equation 4.12), was used to vary the step size and transition to an orbital state at v^* . The robot was allowed to take up to six steps, but the trial could end sooner if a fall or a slip were detected. Such parameters leading to a fall or slip were labeled as infeasible. The resulting feasible/infeasible dataset was used to define a classification boundary using support vector machine classification. The feasible dataset was used to test the accuracy of the LIPM dynamics and develop an improved model using polynomial regression. An overview of the data collection method is shown in video [40] at 0:05.

4.7.2 Feasible Boundary from simulator data

A support vector machine (SVM) classification model was used to approximate the boundary of the feasible data set. The SVM binary classification algorithm from the MATLAB stats toolbox with a polynomial kernel function was used to search for the optimal hyperplane that splits the data into two classes, feasible and infeasible. A training set of 693 combinations of x_k^- and v_k^- was obtained from the LIPM-stepping simulation and was used to train the SVM classifier. The decision boundaries of the SVM classifier were approximated using two lines of best fit of the form shown in (Equation 4.28) where $\omega_i(x)$ is the left ($i = 1$) or right ($i = 2$) decision boundary line. (Equation 4.29) shows the SVM classifier equations.

$$\omega_i(x) = c_i + d_i \cdot x \quad (4.28)$$

$$h(x_j) = \begin{cases} +1 \text{ (feasible)} & \text{if } c_1 + d_1 \cdot x_j \geq \omega_1 \\ & \text{and } c_2 + d_2 \cdot x_j \leq \omega_2 \\ -1 \text{ (not feasible)} & \text{if } c_1 + d_1 \cdot x_j < \omega_1 \\ & \text{or } c_2 + d_2 \cdot x_j > \omega_2 \end{cases} \quad (4.29)$$

4.7.3 Polynomial Regression for S2S map from simulator data

Although the LIPM yields a linear set of the S2S equations, there will be incongruences between the LIPM and full nonlinear dynamics. Regression analysis was performed on the simulation data of the robot in MuJoCo to better capture the robot dynamics. The result was a polynomial regression model shown in (Equation 5.7) and (Equation 5.8).

4.7.4 Quadratically Constrained Quadratic Program

An optimization problem was formulated to find the inputs x_k^- and v_k^- that transition the robot to a cyclic gait while minimizing a cost function. The cost function is the squared difference between the outputs and the desired states: $\text{Cost} = A(\tilde{v}_{k+1}^- - v_{k+1}^-)^2 + B(\tilde{x}_{k+1}^- - x_{k+1}^-)^2$. The weights A and B can be tuned to give more importance to achieving the desired foot placement or COM velocity. The optimal x_k^- is used as the input \tilde{x}_k^- to the model-based stepping controllers. The constraints include the regression equations as well as the linear decision boundary lines.

The problem can be expressed as a quadratically constrained quadratic program shown here in matrix form

$$\underset{\mathbf{y}}{\text{minimize}} \quad 0.5\mathbf{y}^T \mathbf{H}\mathbf{y} + \mathbf{f}^T \mathbf{y} \quad (4.30)$$

$$\text{subject to: } 0.5\mathbf{y}^T \mathbf{Q}_i \mathbf{y} + \mathbf{k}_i^T \mathbf{y} + j_i = 0 \quad (4.31)$$

$$\mathbf{p}^T \mathbf{y} < b_0, \quad \mathbf{p}^T \mathbf{y} > b_1 \quad (4.32)$$

$$\mathbf{LB} \leq \mathbf{y} \leq \mathbf{UB} \quad (4.33)$$

where $\mathbf{y} = \begin{bmatrix} x_{k+1}^- & v_{k+1}^- \end{bmatrix}^T$, \mathbf{f} , and \mathbf{H} are user chosen constants in the cost function. (Equation 4.31) accounts for the nonlinear equality constraints composed of the polynomial regression equation (Equation 5.8). (Equation 4.32) accounts for the linear inequality constraints composed of the SVM classifier constraints (Equation 4.29). The lower (LB) and upper bounds (UB) are set to be $[-0.5, 0]$ and $[0.6, 1.2]$, respectively.

4.8 Results

4.8.1 Support Vector Machine Classification

Figure 24 shows the feature space of the training set along with the SVM classification boundary lines where the blue dots are feasible samples and the red dots are infeasible samples. The black line is the SVM optimal hyperplane contour line, and the dashed green line is the approximated linear boundary used as the decision boundary for the model. The overall classification model has an accuracy of 95.8%, and the precision in predicting feasible is 99.7%.

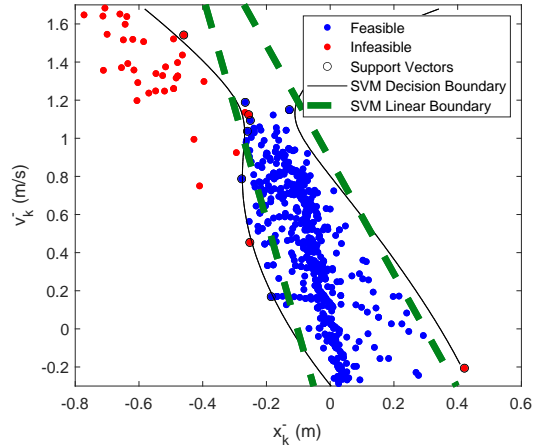


Figure 24: Feature space and optimal hyperplane of test set.

4.8.2 Quadratic Polynomial Regression

The coefficients of the regression model are $\alpha_{0:5} = [-0.0002, 1.7871, 0.4268, 0, 0.0968, 0]$ and $\beta_{0:5} = [-0.0245, 4.0076, 1.5064, 0.5879, 1.1168, 0.1221]$ all with p-values < 0.01 . The adjusted R-squared value for the regression analysis was 0.988 for x_k^+ and 0.987 for v_k^+ . The mean absolute error of the regression model was 0.008 m for x_k^+ and 0.0285 m/s for v_k^+ marking an improvement of 22.5% and 60.8%, respectively, over the LIPM. Figure 38 shows the S2S modeling absolute errors of the test set using the LIPM (left) and the regression model (right). The S2S COM position is captured well using both models. As shown in Figure 38(a) and Figure 38(b), both models yield an absolute error of less than 0.02 m for most of the parameter space. The S2S COM velocity is better captured across all of the test set using the regression model as shown in Figure 38(d).

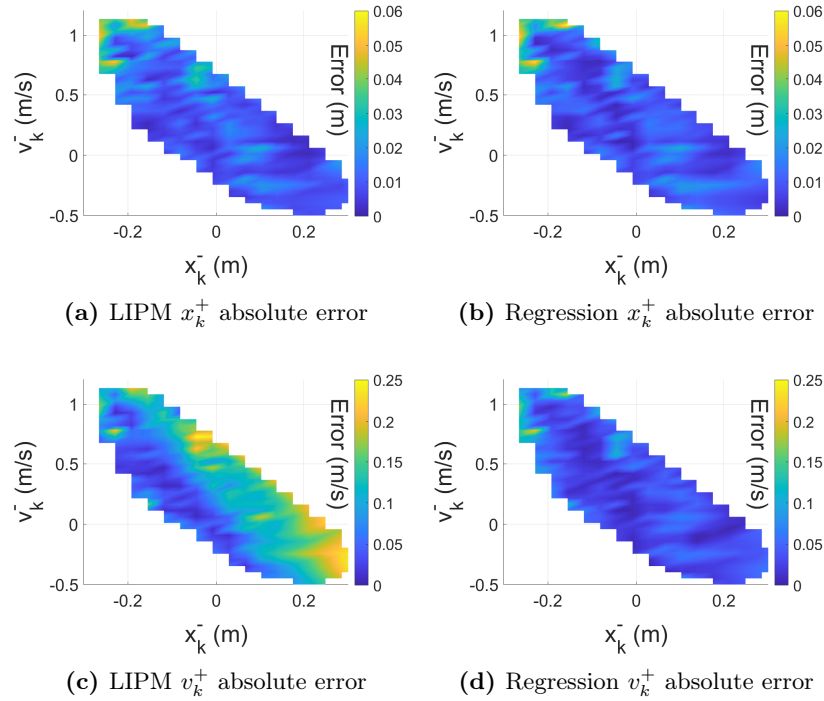


Figure 25: Surface plots of the S2S modelling errors.

4.8.3 Stepping Control: tracking a reference velocity

We tested the model and controller framework's ability to follow a reference velocity profile $v_{desired}$. The mean absolute error was used as a performance metric; it was calculated by subtracting the reference velocity from the actual at every step and taking the mean. The LIPM stepping controller with no feedback, (Equation 5.10), gave a mean absolute error of 25.96% while the regression model stepping controller with no feedback, (Equation 5.11), had a mean absolute error of 12.69%; an improvement of 13.27% over the LIPM as shown in 26a. The LIPM stepping controller with feedback, (Equation 4.12), gave a mean absolute tracking

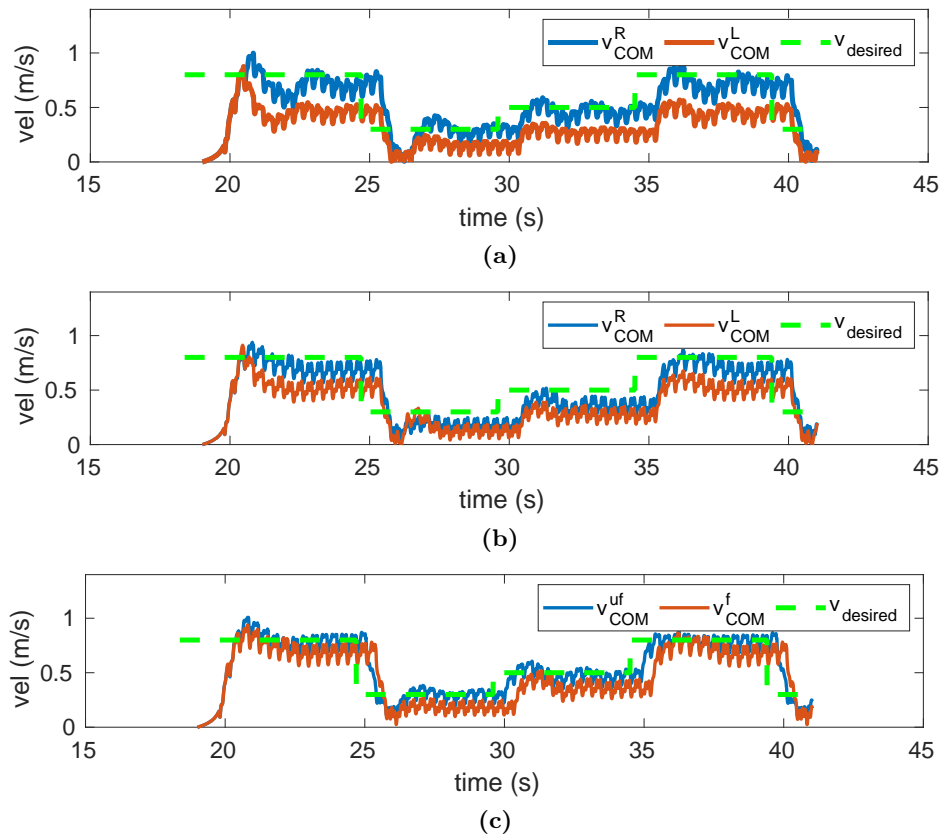


Figure 26: Velocity tracking using the stepping controller (a), stepping controller with feedback (b) and comparison between the LIPM (v_{COM}^L) and regression model (v_{COM}^R). (c) Velocity tracking using regression model with feedback (v_{COM}^f) and footstrike-corrected feedback control (v_{COM}^{uf}).

error of 22.45% while the regression model stepping controller with feedback, (Equation 4.13), gave a mean absolute error of 14.3%; an improvement of 8.15% over the LIPM-based stepping controller with feedback as shown in 26b. The new stepping controller with footstrike correction, (Equation 5.12), yielded a mean absolute error of 9.48%, which is an improvement of 4.82% over the regression stepping controller with feedback (Equation 4.13) as shown in 26c.

4.8.4 Footstrike-corrected Controller

Using a footstrike-corrected stepping controller reduces the velocity tracking errors, as seen in 26c. Updating the stepping controller using the robot state x_{k-1}^{+R} as opposed to the model-predicted state $\tilde{x}_{k-1}^{+L/N}$ eliminates the prediction error seen in model-based stepping control. Furthermore, it makes the controller robust to perturbations that the robot may experience during the stance phase. A simulation was conducted where the robot walked forward at steady-state velocity. Two controllers were used: a model-based stepping controller with feedback and a model-based footstrike corrected controller. To test the controller's ability to withstand perturbations while walking, we applied a 175N force for 100ms at the torso along the x-direction (forward direction) in the middle of the stance phase. The force caused the robot dynamics to deviate from the S2S dynamics prediction.

The COM position x_{com} and footstrike state x_k^- are shown in Figure 27. The footstrike COM velocity state v_k^- is shown in Figure 28. The perturbation occurs at point (a) causing v_k^- of the next step to increase; this is illustrated by points (b) and (d). However, at point (b), the footstrike-corrected controller has updated the model's state and takes a longer step x_k^- . Meanwhile, at point (d), the model-based controller does not update the model and takes the step size x_k^- it would normally take without a perturbation. The result is that at (c), the longer step of the footstrike-corrected controller has reduced the velocity; conversely, at point (e), a short step causes the COM velocity to spike, causing the robot to become unstable [40](video at 1:26).

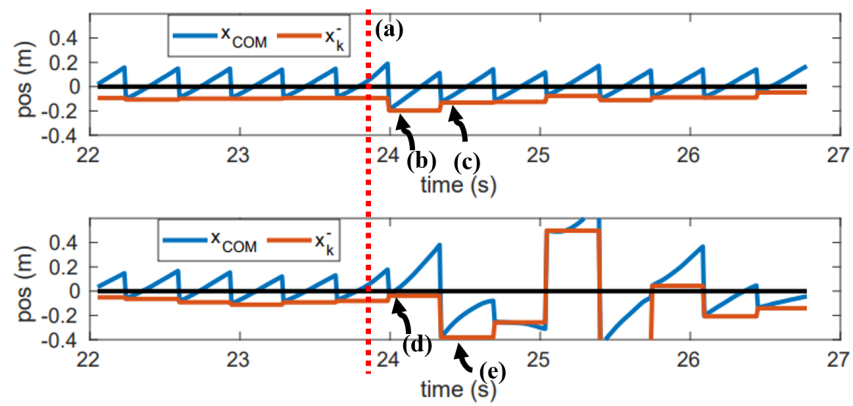


Figure 27: Position data of COM during perturbation trial using a footstrike-corrected controller (Top) and a model-based controller (Bottom).

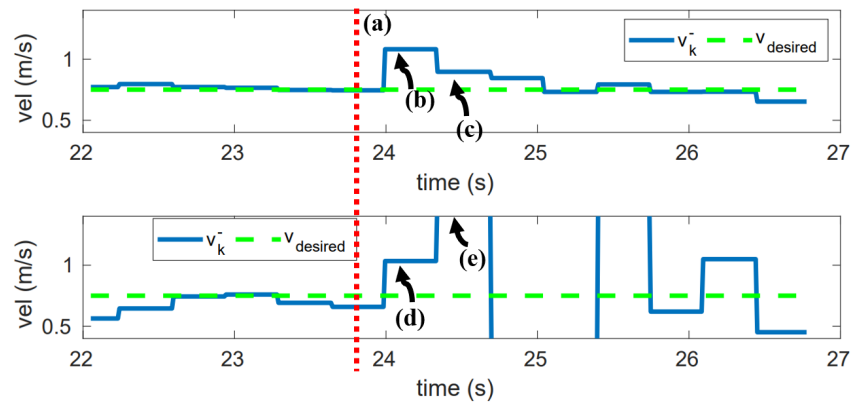


Figure 28: Velocity data of COM during perturbation trial using a footstrike-corrected controller (Top) and a model-based controller (Bottom).

4.8.5 Optimization: Stepping over obstacles

We tested the model and optimization framework’s ability to plan motion on terrain with floor obstacles. The optimization’s cost gains vary depending on the robot’s environment and the task requirements. Bigger weights are given to velocity cost when foot placement is not critical; such is the case when no obstacles are ahead and a given walking velocity is desired. When foot placement is vital, bigger weights are given to position cost; such is the case when obstacles are ahead and the robot must precisely place its feet to avoid tripping. When the robot is closer than 1.5 m from the obstacle, the larger gains are switched from velocity to foot placement.

An additional linear constraint is added to the optimization problem constraining the foot-strike location to be before or after the obstacle. One issue with this formulation is that we need to solve multiple quadratic programs for the footstep planning, one for each feasible footstep location encompassed by obstacles on either side. The choice of the obstacles was such that only two quadratic programs had to be solved, one for stepping before the obstacle and one after the obstacle. After solving both problems, the solution with the lowest cost was chosen as \tilde{x}_k^- in the regression model-based stepping controller with feedback (Equation 4.13). The control \tilde{x}_k^- was bounded by the feasibility boundary lines and the training set boundary, limiting the robot’s ability to take longer or shorter steps when necessary. The gains of the cost function were varied accordingly to give priority to velocity or foot placement as the robot approached the obstacles.

Figure 29 shows a step-by-step simulation of Digit successfully walking over two obstacles using the regression model stepping controller with feedback (see video at 2:31 in Ref. [40]). Figure 30 illustrates the states of the robot during the simulation where the time point (*a*) is the location of the first obstacle and the time point (*b*) is the location of the second obstacle. From the figure, we see that Digit takes a very short step before stepping over the obstacle; this causes v_k^- of the following step to increase. Subsequently, a larger step is taken to clear the obstacle. The robot transitions back to the desired steady-state velocity momentarily before encountering the next obstacle. Again, it takes a very small step almost in line with the other foot before taking a longer step to clear the obstacle.

4.9 Discussion

A polynomial regression model that better predicts the S2S dynamics of the robot was created via 2D forward walking simulations using LIPM-based stepping controllers. Although the LIPM is a good one-size-fits-all approach to modeling bipedal walking, it assumes a point mass which may produce modeling errors for heavy bipeds with heavy torsos and swinging arms like Digit. Modeling errors can be fixed at the controller level using an integral feedback term or a state-dependent bias. We proposed an alternative solution: using simulated data from the region near the periodic gait to fit the S2S dynamics of the robot into a polynomial regression model to obtain a wider control region. The model was extrapolated using only two parameters: footstrike position and COM velocity at footstrike. Adding more parameters to the model, such as angular momentum, can further enhance the model. Although the model was tested in the

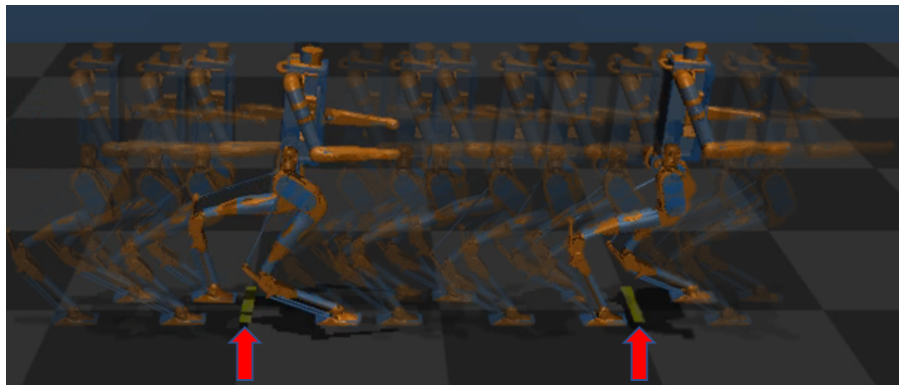


Figure 29: Simulation of Digit avoiding a trip by stepping over two obstacles (indicated by the red arrow).

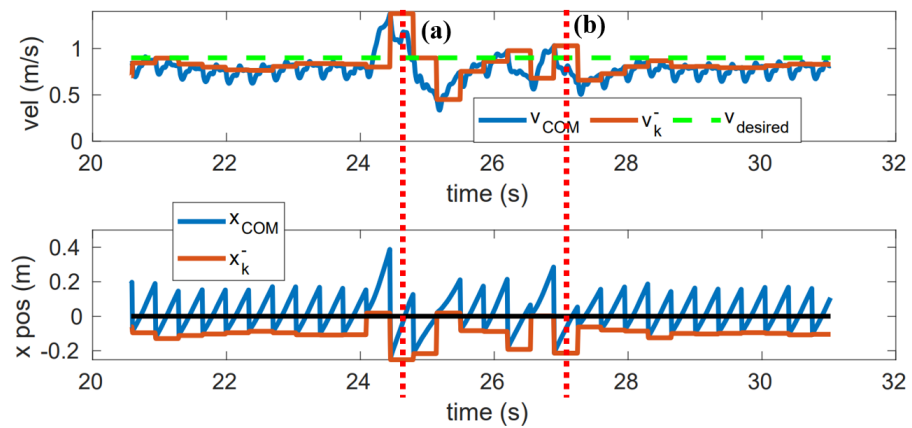


Figure 30: Velocity (top) and COM position (bottom) with respect to stance foot of Digit while walking over obstacles.

2D case, the methodology can be extrapolated to the 3D case as was done using the LIPM in [8, 9, 37].

The proposed approach captures some of the non-linearities not captured by the LIPM. Comparing the LIPM stepping controllers and regression model stepping controllers shows that the regression model stepping controller performs better at velocity control. Improving the model, therefore, also enhances the controller. One limitation of the current model is that it relies on a constant stepping frequency and COM height. This limits the ability to use any step width and walking velocity combination, making motion planning and obstacle avoidance challenging. Future work will focus on addressing these limitations.

A footstrike-corrected feedback controller was introduced to reduce the errors in predicting the footstrike states. This controller is more robust than the model-based stepping controller and can withstand certain perturbations occurring at midstance. One limitation that this controller does not address is that sometimes the steps it takes to recover might not lie inside the feasible set. The robot may eventually become unstable and fall. One possible solution is optimizing the foot placement over a window of several steps such that the robot can recover from a perturbation after several steps lying inside the feasible boundaries. Adding more constraints such as kinematic limits may also enhance the control robustness.

The simple quadratic models and linear constraints allowed us to solve the quadratically constrained quadratic program in less than 100 iterations using a sequential quadratic programming (SQP) algorithm; this is promising for real-time control and for further exploring alternative cost functions such as reduction in motor power or cost of transport. A simple optimization

problem was formulated for stepping over obstacles. We limited the planning window to one step. Adding more steps for planning may reduce costs, but the sizeable stepping combination would need suitable heuristics to be solved quickly.

4.10 Conclusions and Future Work

This paper presented a data-driven approach to fitting the S2S dynamics of Digit into a polynomial regression model using data generated from a LIPM-based control walking simulation. The regression model was used to develop three types of controllers: model-based, model-based with feedback, and footstrike-corrected controllers. The proposed model is more precise at predicting the S2S dynamics of the robot, which in turn improves the performance of the controllers. We formulated a quadratically constrained quadratic program to solve for the optimal control and used SVM modeling to approximate a feasibility boundary line constraint. We demonstrated that all three stepping controllers developed with the regression model display fewer velocity tracking errors than LIPM-based controllers. In addition, we showed that we could use an optimization problem to plan over terrain with obstacles with high accuracy and few iterations. We presented a more precise data-driven approximation of the S2S dynamics of Digit, which in turn improves the model-based controllers compared to the LIPM. A footstrike-corrected stepping controller was presented, further enhancing the model-based controllers by reducing the points of error in the control formulation. Computationally friendly models and controls will be developed with a similar data-driven approach in future work. The evaluation of this approach on Digit will be performed in hardware.

CHAPTER 5

DATA-DRIVEN IDENTIFICATION OF A NON-HOMOGENEOUS INVERTED PENDULUM MODEL FOR ENHANCED HUMANOID CONTROL

Overview: This work aims to enhance the linear inverted pendulum model (LIPM) for bipedal robot control. While the LIPM simplifies the dynamics by assuming homogeneity, it fails to capture important nonlinear dynamics observed in real-world scenarios. To address this limitation, we propose the non-homogeneous LIPM (NH-LIPM), which incorporates a non-homogeneous term in the traditional LIPM dynamics. The NH-LIPM is augmented with controllable inputs, allowing for greater parameter control compared to the LIPM. Through regression analysis and the use of the Recursive Least Squares algorithm with forgetting, we extract and adaptively tune the NH-LIPM parameters. Evaluation through high-fidelity simulation and experimentation on a 30-degree-of-freedom humanoid demonstrates that the NH-LIPM offers improved velocity tracking control, particularly when ankle torque with damping control is added. This model provides a flexible framework for simultaneously controlling the center of mass velocity and position, enabling precise reference tracking and enhanced bipedal locomotion. A video is linked here: <http://tiny.cc/NHLIPM>

Parts of this chapter are taken from the following accepted journal article:
Hernandez Hinojosa, E., and Bhounsule, P. A., "Data-driven Identification of a Non-homogeneous Inverted Pendulum Model for Enhanced Humanoid Control" Proceedings of the 2023 IEEE-RAS International Conference on Humanoid Robots. Austin, Texas. Dec 12–14, 2023.

5.1 Introduction

Bipedal robots, such as Agility Robotics’s Digit, strive to replicate human-like balance, walking control, and adaptability. However, achieving robust walking capabilities for these robots remains a significant challenge. Factors such as high dimensionality, nonlinear dynamics, and under-actuation due to limited ankle torques contribute to the complexity of bipedal robot control.

In the field of bipedal robot control, two main approaches have emerged. The first approach, the linear inverted pendulum mode (LIPM), uses a point mass model, ignoring most of the nonlinearities, to develop controllers [31]. This controller is then transferred to a full robot model using force and/or position control. However, this approach overlooks the influence of angular momentum and center of mass differences, limiting its ability to capture the full dynamics of the system. The second approach involves developing a trajectory tracking controller using a full-order robot model that behaves as a low-order model, typically through the concept of hybrid zero dynamics [24]. This approach faces challenges when the system deviates significantly from the reference trajectory, leading to stabilization difficulties.

In this study, we propose an approach that leverages the strengths of both approaches. We employ tight inverse kinematics trajectory tracking on the feet and an inverse-dynamics-based torso orientation control such that the dynamics of the center of mass (COM) of the robot are predictable. We then use the LIPM as the base function of the Step-to-Step (S2S) dynamics. The S2S dynamics refer to the time-evolving behavior and state transition of the system within the time-scale of a step. The LIPM exhibits modeling inaccuracies that contribute

to errors in the S2S control of the robot. To mitigate such errors, we propose adding a non-homogeneous equation to the homogeneous LIPM dynamics producing the non-homogeneous linear inverted pendulum model (NH-LIPM). Two methods are used to extract the solution to the non-homogeneous equation. The first is by extracting an analytical polynomial regression on the LIPM S2S error, and the other is utilizing the recursive least squares (RLS) algorithm to adaptively find the optimal parameters of the non-homogeneous equation.

5.2 Background and Related Work

The control of bipedal robots poses significant challenges due to their high degree of freedom and underactuation at the ankle joint. Achieving stability in bipedal locomotion often relies on orbital stability, where control is achieved at the time-scale of each step [5]. Passive dynamic walkers, for example, demonstrate orbital stability by utilizing natural dynamics to walk stably down slopes [6] [32]. Actuators can be employed to achieve similar behavior on level ground by replacing gravitational energy with joint torques [13]. Stability evaluation of such cyclic systems can be performed using a S2S map or Poincaré map, which relates states at one instance of the cycle to the same state in the next cycle [19]. Stability analysis based on eigenvalue analysis of the map helps determine the stability of the system and design control strategies to achieve desired stable eigenvalues [14].

Hybrid zero dynamics is an alternative method for achieving S2S stability by utilizing virtual constraints that track controlled degrees of freedom as a function of uncontrolled degrees of freedom [26, 33]. Although this reduces the dimensionality of the Poincaré map, it does not guarantee S2S stability. Various methods, such as finding controls with eigenvalues less than

1, modifying virtual constraints, or introducing event-based stabilization, can be employed to achieve S2S stability [27]. However, linearized stability achieved through these methods may be compromised in the presence of significant perturbations.

To simplify control and develop orbitally stable controllers, several simple models have been proposed, such as the Linear Inverted Pendulum Model (LIPM) for walking and the Spring-Loaded Inverted Pendulum Model for running [34] [35] [41]. These models provide insights into the dynamics and appropriate foot placement for achieving desired walking velocities. However, limitations arise from modeling errors due to nonlinearities that are not captured by these simplified models [41].

Previous studies have explored enhancements to the LIPM by incorporating time-varying dynamics and forcing functions derived from ankle torque inputs. For example, an angular momentum-based LIP (ALIP) model with a forcing function was developed to simulate walking over dynamic surfaces [42]. In our previous work, polynomial regression analysis was employed to extract an analytical S2S map of the COM position and velocity, improving dynamic predictions and velocity tracking in 2D simulations [41]. This study expands upon the previous work by extending it to the 3D case. This study introduces the concept of a NH-LIPM that captures the time and state dependency of the dynamics. It incorporates forcing functions derived from ankle torque inputs, enabling further control enhancement.

Other research efforts have explored adaptive control approaches to address model mismatches and improve control robustness [43]. These include using neural networks for adaptive control and developing adaptive virtual models to respond to parameter variations and external

disturbances [44]. However, these approaches focus on adaptation at the controller level and may not yield significant improvements in forward speed control or continuous dynamics in the x-axis.

In this study, we propose the use of the recursive least squares algorithm with forgetting to adaptively tune the parameters of the discrete NH-LIPM S2S model in real-time for forward and lateral walking. The forgetting factor in the controller allows us to select a model that adapts quickly to changes or at a slower rate, retaining the memory of previously sampled data.

5.3 Models

5.3.1 Robot model

Digit is an advanced bipedal robot with impressive capabilities. It has 30 degrees of freedom and features 20 actuated joints, as illustrated in Figure 1 (refer to [38] for more comprehensive information on Digit). The design of Digit’s legs draws inspiration from its predecessor, Cassie [8], but incorporates notable enhancements. These enhancements include the addition of a roll joint at the toe and the inclusion of an upper body comprising a torso and two arms.

To understand the leg kinematics of Digit, we can visualize it as two interconnected closed-loop chains. These chains are represented by distance constraints between the heel spring and the hip pitch joints. During the swing leg phase, for the sake of simplicity, the knee and tarsus joint positions are inversely coupled as explained in [41].

5.3.2 Step-to-step (S2S) Models

The general form of the S2S models developed in this study is illustrated in [45] and the general S2S map is shown in (Equation 5.1). During the initiation of the single support phase

(SSP), the COM position and velocity relative to the stance foot are denoted as x_k^- and v_k^- , respectively, where k represents the current step number. As the current SSP concludes, these states transition to x_k^+ and v_k^+ . The dynamics model, represented as ζ , captures the S2S relationship, mapping the initial states $\mathbf{s}_k^- = \{x_k^-, v_k^-\}$ to the resulting states $\mathbf{s}_k^+ = \{x_k^+, v_k^+\}$.



Figure 31: Robot model: Digit bipedal robot with 30 degrees of freedom and 20 actuated joints.

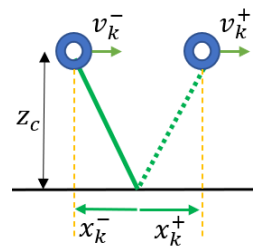


Figure 32: The S2S model maps the state at s_k^+ with the state at s_k^- and control during the step u_k .

$$\mathbf{s}_k^+ = \zeta(\mathbf{s}_k^-, \tau_p) \quad (5.1)$$

5.3.2.1 Analytical map from the LIPM

The LIPM maintains the height of the COM at an approximately constant level by controlling the hip, knee, and ankle joints. The LIPM equations are linear and involve parameters such as gravity (g), the constant height of the COM (z_c), and the mass of the robot (m).

$$\ddot{x} = \frac{g}{z_c}x + \frac{1}{mz_c}\tau_p, \quad \ddot{y} = \frac{g}{z_c}y + \frac{1}{mz_c}\tau_r \quad (5.2)$$

The input pitch and roll torques are denoted as τ_p and τ_r , respectively. Assuming no input torques, the equations can be integrated analytically from time 0 to T_s , resulting in an equation that conforms to the form of (Equation 5.1).

$$\zeta(\mathbf{s}_k^-) = \begin{bmatrix} x_k^{L+} \\ v_k^{L+} \end{bmatrix} = \begin{bmatrix} C_T \cdot x_k^- + T_c S_T \cdot v_k^- \\ S_T/T_c \cdot x_k^- + C_T \cdot v_k^- \end{bmatrix} \quad (5.3)$$

Here, $S_T = \sinh(T_s/T_c)$, $C_T = \cosh(T_s/T_c)$, and $T_c = \sqrt{z_c/g}$. The superscripts L and N appended to the state variables indicate whether they belong to the LIPM or the regression model, respectively. It is important to note that the equations apply to both the x (frontal) axis and y (lateral) axis.

5.3.3 System homogeneity and state-independence

The homogeneity property of the LIPM, displayed in Equation 5.2 when $\tau = 0$, enables analytical solutions and simplifications, making it a viable model for investigating bipedal walking dynamics. Additionally, the pendulum dynamics exhibit state independence and symmetry, relying solely on the COM height. This behavior is illustrated in the phase portrait shown in Figure 33, where the blue lines depict possible system dynamics at a given stepping frequency and COM height. The dotted red lines represent the forward and backward walking of the LIPM, completing a cycle in a single step, while the dotted black lines demonstrate lateral walking, completing a cycle in two steps. Figure 34 illustrates the phase trajectories for the LIPM (red) and a NH-LIPM (purple and black). The NH-LIPM displays asymmetry. An initial state A would normally lead to the final state B in the LIPM. However, in a NH-LIPM the final

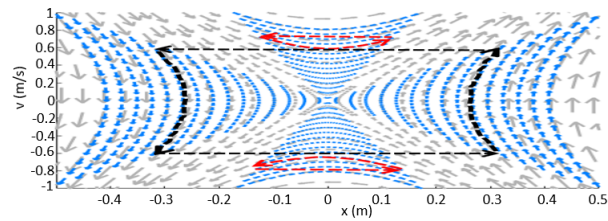


Figure 33: Illustration of the phase portrait of the LIPM with two P1 orbits shown in red and one P2 orbit shown in black.



Figure 34: Phase asymmetry displayed in NH-LIPM dynamics (black and purple trajectories)

state might be at C. Similarly, a starting state at D would take the same path as starting state A in the LIPM and end at state E, but would take the purple path in the NH-LIPM to F.

5.3.3.1 Non-homogeneous LIPM

The LIPM falls short of capturing the time and state-dependent variations and nonlinearities present in real-world dynamics as illustrated in Fig Figure 34. The NH-LIPM incorporates a time-varying and state-dependent component that captures the deviations from the idealized linear inverted pendulum dynamics. To derive the time and state-dependent equation, data was collected to conduct a regression analysis and extract the non-homogeneous term that compliments the LIPM, also known as the particular solution.

$$\mathbf{s}' = \mathbf{A}\mathbf{s} + \mathbf{q}(t, \mathbf{s}) \quad (5.4)$$

$$\mathbf{s}(t) = e^{\mathbf{A}t}\mathbf{s}(\mathbf{0}) + \int_0^t e^{\mathbf{A}(t-\gamma)}\mathbf{q}(\gamma, \mathbf{s}(\gamma))d\gamma \quad (5.5)$$

$$\boldsymbol{\eta}(t) = \zeta(\mathbf{s}^-, t) + \mathbf{h}(\mathbf{s}^-, t) \quad (5.6)$$

Equation 5.4 is the general dynamics equation of the NH-LIPM where \mathbf{q} is the time-variant nonhomogeneous term. Equation 5.5 is the system equation of motion which can be expressed as the sum of Equation 5.1 and the particular solution \mathbf{h} as shown in Equation 5.6. The non-homogeneous model incorporates a time-varying component that captures the deviations from the idealized linear inverted pendulum dynamics. The objective is to find the analytical equation, \mathbf{h} , that captures the time-variant dynamics of the system and is a solution to the nonhomogeneous system in Equation 5.4.

$$h_1(s_k^-, t) = x_k^{N+} = \alpha_0 + \alpha_1 x_k^- + \alpha_2 v_k^- + \alpha_3 t.. \quad (5.7)$$

$$... + \alpha_4 (x_k^-)^2 + \alpha_5 (v_k^-)^2 + \alpha_6 t^2 + ... + \alpha_7 x_k^- v_k^- ..$$

$$... + \alpha_m (x_k^-)^n (v_k^-)^n t^n$$

$$h_2(s_k^-, t) = v_k^{N+} = \beta_0 + ... \quad (5.8)$$

The polynomial regression equation of degree n will fit the form in Eqns. Equation 5.7 and Equation 5.8.

5.3.4 Enhancing the NH-LIPM with Forced Inputs

The NH-LIPM's control method is limited to foot placement and COM height. To further enhance the model, a forcing term, \mathbf{g} , is introduced to account for input parameters, such as ankle torque/damping, which are not considered in the traditional LIPM. Regression analysis is applied to extract a function incorporating these additional parameters. This refined model called the forced NH-LIPM (F-NH-LIPM), offers an improved representation of the bipedal robot's dynamics.

$$\boldsymbol{\eta}(t) = \zeta(\mathbf{s}^-, t) + \mathbf{h}(\mathbf{s}^-, t) + \mathbf{g}(\mathbf{s}^-, t, \boldsymbol{\tau}) \quad (5.9)$$

where $\boldsymbol{\tau}$ are control inputs that may be desirable for better continuous inter-step control.

5.4 Methods

5.4.1 Stepping Controllers

The S2S map from the analytical LIPM and data-driven NH-LIPM were used to develop stepping controllers for stable walking. We assumed that there is no double support phase during walking and that the COM is located at the hip. Inverse dynamics is used to apply a force on the base to ensure the hip position is at a desired constant height z_c .

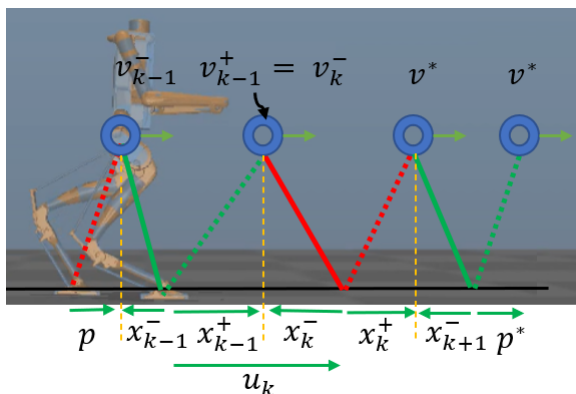


Figure 35: Model-based stepping. The stepping controller dictates the step size u_k to achieve desired velocity v^* .

5.4.1.1 Model-based Stepping

The stepping controller inputs are step lengths u_k^L for the LIPM and u_k^N for the NH-LIPM.

$$u_k^L = x(t)_{k-1}^{+R} - x_k^{L-} = x(t)_{k-1}^{+R} - (v_{des}^+ - C_T v_k^-) T_c / S_T \quad (5.10)$$

$$u_k^N = x(t)_{k-1}^{+R} - x_k^{N-} = x(t)_{k-1}^{+R} - \hat{h}_2^-(v_k^-, x_k^-) \quad (5.11)$$

where $x(t)_{k-1}^{+R}$ is the continuous state of the robot, \hat{h}_2^- is the inverse function of $\hat{h} = h_2(s_k^-, t) + \zeta_2(s_k^-, t)$ used to solved for x_k^- given a desired $v_k^+ = v_{des}$.

Feedback may be added to the controller to add stability, as shown below.

$$u_x^R = u_k^N + \mathbf{K} (s(\mathbf{t})_{\mathbf{k}}^{\mathbf{R}} - s_{\mathbf{k}}^{\mathbf{N}}) \quad (5.12)$$

where u_x^R is the updated controller.

Along the lateral direction, the controller u_y^R is employed.

$$r_p = (w_s)/2 \quad (5.13)$$

$$r_v = (v_s) + j(r_p) \quad (5.14)$$

$$\begin{aligned} u_y^R = & y(t)_{k-1}^{+R} + w_s + \mathbf{K}_1 (\mathbf{s}(t)_k^R - \mathbf{s}_k^N) \dots \\ & \dots + \mathbf{K}_2 (\mathbf{s}(t)_k^R - \mathbf{r}) \end{aligned} \quad (5.15)$$

where w_s is the desired step width of the robot, r_v is the velocity vy_k^+ when walking in a cyclic gait with y_k^- equal to r_p . The vector $\mathbf{r} = [r_p, r_v]^T$. The function j gives the cyclic velocity as a function of the desired state y^- which in this case is half the step width, $w_s/2$. To move laterally, the term v_s is added and, therefore, can be used to control lateral walking speed. The step width w_s should be negative during the left stance and positive during the right stance. More details on LIPM-based stepping control can be found in [41] and [9].

5.4.2 Data Collection

To gather the necessary training data, we employed a high-fidelity simulator, which emulates the behavior and characteristics of the robot. This simulator is proprietary to Agility Robotics. During these data collection trials, we followed a procedure akin to what would be done on the actual hardware. Running every trial at close to real-time speed and starting and stopping each trial individually.

The simulator was used to run forward and lateral walking simulations, which allowed us to gather S2S data in both discrete and continuous formats. Similarly, on hardware, a study was

done on forward/backward (frontal) walking. By adjusting the desired walking speed during training, with control inputs x_k^- and v_k^- , we inherently introduced variability across different trials (see video [46]). This variation enabled us to examine the impact of different walking speeds on the robot's behavior. For every step taken x_k^- , v_k^- , $x(t)$, and $v(t)$ were stored parameters.

To derive the equation of the particular solution of the NH-LIPM, regression analysis was employed on the error between the LIPM dynamics and actual robot dynamics shown in Equation 5.16. The NH equation will then capture some of the dynamics not captured by the LIPM and the summation of the LIPM and the NH term will better approximate the robot dynamics as shown in Equation 5.17.

$$\mathbf{e}_k^L = \mathbf{s}(t)_k^{+R} - \mathbf{s}(t)_k^{+L} \quad (5.16)$$

$$\mathbf{s}(t)_k^{+R} \approx \mathbf{s}(t)_k^{+L} + \mathbf{h}(s_k^-, t) \quad (5.17)$$

Data was collected using the Agility Robotics simulator. For every training trial, the robot walked using LIPM-based control at various speeds between 0 and 0.7 m/s, ensuring every speed, with 0.1 increments, was trialed for at least 5 seconds. The regression analysis was done with initial states (x^-, v^-) and time (t) as parameters and the continuous state errors $(\mathbf{e}_k^L(t))$ as regression outputs.

The F-NH-LIPM equation was formulated by fitting a function to the model error with inputs to the equation as shown in Equation 5.19.

$$\mathbf{e}_k^N = \mathbf{s}(t)_k^{+R} - \mathbf{s}(t)_k^{+L} - \mathbf{h}(\mathbf{s}_k^-, t) = \mathbf{g}(\mathbf{s}_k^-, t, \tau) \quad (5.18)$$

$$\mathbf{s}(t)_k^{+R} \approx \mathbf{s}(t)_k^{+L} + \mathbf{h}(\mathbf{s}_k^-, t) + \mathbf{g}(\mathbf{s}_k^-, t, \tau) \quad (5.19)$$

where \mathbf{e}_k^N is the model error of the NH-LIPM and \mathbf{g} captures the dynamics associated with the force inputs τ .

The LIPM and NH-LIPM controllers can control v_k^+ or x_k^+ , but not both. The F-NH-LIPM (Equation 5.19) has two control inputs, u_k^N and τ , so it can be used to control the states v_k^+ and either x_k^- or x_k^+ . We use τ to control v_k^+ and leave x_k^- as a free parameter to which we may assign an arbitrary value.

Two control inputs were tested: ankle damping and ankle torque. The ankle torque was a ramp torque applied during the first quarter of the cycle. For every training trial, the robot walked using LIPM-based control at speeds between 0 and 0.7 m/s with seven different force input values. Every combination was tested for at least ten steps. The regression analysis was done with initial states (x^-, v^-) , inputs τ , and time (t) as parameters and the continuous state errors $(\mathbf{e}_k^N(t))$ as regression outputs.

5.4.3 Continuous Velocity Tracking Using Toe Torques

Another approach tested in this study is the use of toe/ankle torques to force the continuous dynamics to follow a desired trajectory. Figure 36 shows a simple depiction of a desired trajectory of the continuous position, $p(t)$, and velocity, $v(t)$ states of the robot's COM. If we

set a control law that applies an ankle/toe torque proportional to the deviation from the robot states $S_{x,y}(t)$, we can drive the robot states toward the desired trajectory $traj_{x,y}$.

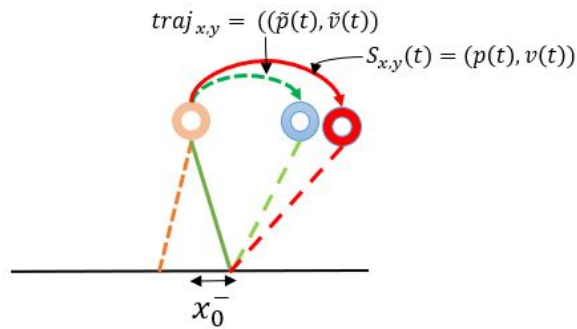


Figure 36: LIPM diagram showing the desired continuous dynamics trajectory in green and the actual robot dynamics in red.

In the model-based stepping control applied in this study, the toe torques of the stance foot are given zero torque and are allowed to passively move under the natural dynamics of the robot. Applying a pitch, τ_p , and roll, τ_r , torque of the form

$$\tau_p = K_p(traj_x - S_x(t)) \quad (5.20)$$

$$\tau_r = K_r(traj_y - S_y(t)) \quad (5.21)$$

allows the F-LIPM controller some control authority to follow a reference position or velocity in the continuous domain. K_p and K_r are tunable gains. Because Digit does not possess actuators

at the toe pitch and roll the following relations are used to map τ_{au_p} and τ_{au_r} to actuator torques τ_{RTA} , τ_{RTB} , τ_{LTA} , and τ_{LTB} .

$$\tau_{RTA} = 0.25(\tau_R + \tau_p) \quad (5.22)$$

$$\tau_{RTB} = 0.25(\tau_R - \tau_p) \quad (5.23)$$

$$\tau_{LTA} = 0.25(\tau_R - \tau_p) \quad (5.24)$$

$$\tau_{LTB} = 0.25(\tau_R + \tau_p) \quad (5.25)$$

To enhance LIPM control, in this study, $traj_x$ and $traj_y$ are chosen to be the continuous LIPM x and y velocities $v_x^L(t)$ and $v_y^L(t)$. The toe torque controller becomes

$$\tau_p = K_p(v_x^L(t) - v_x(t)) \quad (5.26)$$

$$\tau_r = K_r(v_y^L(t) - v_y(t)) \quad (5.27)$$

Hardware trials were conducted to evaluate the effectiveness of the F-LIPM controller in tracking a reference velocity of 0.1 m/s and 0.2 m/s. Data was collected to compare tracking effectiveness with regular LIPM and NH-LIPM stepping control.

The F-LIPM controller's ability to regulate two separate states v^- and x^+ was also tested. The F-LIPM stepping controller has two control inputs, toe torque and foot placement. To

control the desired states, the LIPM stepping controller was used for foot placement. However, the control state x^- was set according to the following control law

$$x_1^{L-} = (v^* - C_T \cdot v_1^-) \cdot T_c / S_T \quad (5.28)$$

$$u^L = x_0^+ - x_1^{L-} \quad (5.29)$$

where v^* is the desired walking speed of the LIPM stepping controller. However, to test the ability to regulate x^- and v^- the desired velocity v^* was set to 0 m/s. The same approach can be utilized to regulate x^- and v^- using a (F-NHLIPM) controller with the following control law

$$x_1^{N-} = \hat{h}_2^-(v_1^-, x_1^-) \quad (5.30)$$

$$u^L = x_0^+ - x_1^{N-} \quad (5.31)$$

where \hat{h}_2^- is an inverse function as explained for Equation 5.11.

The stance foot toe torque controller can then be used to drive the velocity to the desired state v_{xdes}^+ using the control function

$$\tau_p = K_p(v_{xdes}^+ - v_x(t)) \quad (5.32)$$

This proposed method allows for control of the walking speed v^+ that is decoupled to a certain degree from the control state x^- . A simulation experiment will be run to test the controller's

ability to track a reference velocity and data will be collected and compared the a trial where no stance toe torque control is applied (regular LIPM-based stepping).

5.4.4 Walking Stabilization Using F-LIPM Control

The previous section discussed the use of toe/ankle torques to enhance the continuous LIPM along the frontal axis leading to improved control authority. Similarly, toe torques may be used to reduce the frontal and lateral instability that arises from inadequate modeling. For lateral control, Equation 5.14 requires proper knowledge of the cyclic states of the robot. Similarly, for frontal walking control, Equation 5.10 also relies on there being small deviations between the modeled dynamics and actual robot dynamics. A model that deviates too much from the actual robot can lead to inaccuracies in r_v and r_p in Equation 5.14 or an erroneous control state x_k^{L-} in Equation 5.10 leading to instabilities. One such example occurs when LIPM-based stepping is applied to the robot with an increased stepping period of 0.55 seconds. The stepping LIPM controller works well with a period of 0.4-0.45 seconds. However, when the period increases to 0.55 seconds, the model prediction accuracy decreases. An experiment is done to test the efficacy of the toe torque F-LIPM controller in stabilizing a controller with an improper model. The controller from Equation 5.26 will in essence drive the velocity of the robot to follow the predicted x-y velocities of the LIPM.

5.4.5 Recursive Least Squares (RLS)

The RLS algorithm was utilized in this study to enable real-time adaptation of the discrete S2S model parameters. The nonhomogeneous term (h in Equation 5.6) of the NH-LIPM can be represented as \hat{y} in the linear form.

$$\hat{\mathbf{y}}^T = [\boldsymbol{\theta} \mathbf{x}^T] \quad (5.33)$$

$$\mathbf{y} = [p_{k+1}, v_{k+1}] \quad (5.34)$$

$$\mathbf{x} = [p_k, v_k] \quad (5.35)$$

$$\boldsymbol{\theta} = \begin{bmatrix} a_{11} & a_{21} \\ a_{21} & a_{22} \end{bmatrix} \quad (5.36)$$

y contains the predicted center of mass position and velocity error states $[p_{k+1}, v_{k+1}]$, x are the current error states, and θ are the model parameters. The RLS LIPM would then take the discrete form of $\boldsymbol{\eta}_k = \zeta_k(\mathbf{s}^-) + \hat{\mathbf{y}}^T$.

The algorithm consists of an initialization state where the error covariance matrix \mathbf{P} is computed, often initialized as an identity matrix, and the initialization parameters $\boldsymbol{\theta}$ are set. The coefficients of the NH function are initialized to zero. For illustration purposes, the RLS algorithm below predicts the state v where a_{21} and a_{22} are the linear parameters of the non-

homogeneous term, h_2 , in the NH-LIPM. This method can similarly be applied to the state p to predict parameters a_{11} and a_{12} in h_1 .

$$\text{Initialize} \left\{ \begin{array}{l} \mathbf{P} = \frac{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}{\lambda} \\ \boldsymbol{\theta}_2 = \begin{bmatrix} a_{21} & a_{22} \end{bmatrix}^T \end{array} \right. \quad (5.37)$$

$$\left. \begin{array}{l} \boldsymbol{\phi} = [p_k, v_k] \end{array} \right\} \quad (5.38)$$

$$\left. \begin{array}{l} \mathbf{K} = \frac{\mathbf{P}\boldsymbol{\phi}^T}{\Delta + \boldsymbol{\phi}\mathbf{P}\boldsymbol{\phi}^T} \end{array} \right\} \quad (5.39)$$

$$\text{Iterate} \left\{ \begin{array}{l} e = v_{k+1} - \boldsymbol{\theta}_2^T \boldsymbol{\phi}^T \end{array} \right. \quad (5.40)$$

$$\left. \begin{array}{l} \boldsymbol{\theta}_2 = \boldsymbol{\theta}_2 + Ke\sigma \end{array} \right\} \quad (5.41)$$

$$\left. \begin{array}{l} \mathbf{P} = \frac{\mathbf{P} - \mathbf{K}\boldsymbol{\phi}\mathbf{P}}{\lambda} \end{array} \right\} \quad (5.42)$$

The algorithm is iterated at every footstep where the regressor, $\boldsymbol{\phi}$, consisting of the inputs, is updated as shown in Equation 5.38, and the Kalman gain, \mathbf{K} , is computed from Eqn Equation 5.39. Δ is the regularization parameter used to prevent overfitting, where larger values reduce the variance but increase the bias. In Equation 5.40, the prediction error, e , is computed by subtracting the current state, p_{k+1} , from the predicted state. The parameters are then updated in Equation 5.41 where σ is the error factor used to smoothen out parameter outputs. Finally, in Equation 5.42 the error covariance matrix is updated by implementing a forgetting factor λ , a value between zero and one where a higher value means less forgetting.

The RLS algorithm operates by iteratively updating the model parameters using a weighted least squares approach. The objective is to minimize the difference between the predicted outputs of the model and the actual measurements obtained during walking as shown in Equation 5.40.

By applying the forgetting factor in Equation 5.42, older data points are gradually assigned less weight in the parameter update process. This allows the model to adapt more quickly to changes occurring in the immediate past while still retaining a degree of information from earlier steps. The forgetting factor can be adjusted to strike an appropriate balance between adaptability and stability, ensuring that the model remains robust despite variations in the robot's dynamics.

To evaluate the performance of the RLS controller, we intentionally varied several parameters that impact the dynamics of the robot. This deliberate variation was carried out to create scenarios where the LIPM no longer accurately captures the system dynamics. Specifically, we adjusted the robot's height by reducing it by 0.1 m, doubled the step width, and modulated the toe damping between 20% and 60% of the maximum damping value of 28.5 Ns/m.

To assess the RLS controller's adaptive tuning capability for model parameters, we conducted a velocity tracking trial and compared its performance to that of the LIPM.

5.5 Results and Discussion

5.5.1 Regression Analysis (Simulation)

The non-homogeneous regression equation consists of a 3rd-order polynomial function of the x^+ (Equation 5.7) and a 7th-order polynomial function of the v^+ map (Equation 5.8). The

adjusted R-squared value for the regression analysis was 0.81 for x_k^+ and 0.79 for v_k^+ . The mean absolute error (MAE) of the continuous NH-LIPM regression was 0.0022 m for x and 0.0116 m/s for v . A reduction of 6% and 80%, respectively, when compared to the LIPM. The MAE of the discrete NH-LIPM regression was 0.0039 m for x and 0.012281 m/s for v . A reduction of 28% and 82%, respectively, when compared to the LIPM. Figure 38 shows the continuous S2S modeling absolute errors of the training set using the LIPM (left) and the NH-LIPM (right). The y-axis on the plot depicts the initial states x_k^- and v_k^- of the tested data. The S2S COM position is captured poorly at faster speeds and towards the end of the step, as denoted by the large error seen at wider x_k^- and higher time (t) in Figure 38(a). Conversely, the NH-LIPM displays errors below 1 cm, as seen in Figure 38(b). The S2S continuous COM velocity prediction does poorly at higher v_k^- and around the middle of the cycle. However, the discrete prediction, marked by the prediction made at time $t = 0.4$ in Figure 38(c) displays a lower error.

Figure 37 depicts the phase portraits of the LIPM and the NH-LIPM. Although the LIPM phase portrait is state-independent, it can only predict the states of the model loosely. However, it is evident that for the given step frequency, the portrait trajectory, and dynamics closely intersect at the end of the cycle. This might be why it is possible to employ LIPM-based stepping for our system, as the error of the discrete S2S map is low. Figure 37(b) shows the phase portrait of the NH-LIPM, which better tracks the continuous dynamics of the system. It is worth mentioning that, unlike the LIPM portrait, the phase lines in the NH-LIPM portrait

will change depending on the initial states x^- and v^- . This is because the actual dynamics of the robot are state-dependent and nonlinear, which the LIPM does not account for.

The regression analysis of the lateral dynamics yielded a 3rd-order time polynomial function for the v^+ map with an adjusted R-squared value of 0.62. The lateral continuous NH-LIPM displayed a mean absolute error of 0.026 m/s, a reduction of 46% when compared to the LIPM. The discrete NH-LIPM displayed a mean absolute error of 0.0408 m/s, a reduction of 41%. Figure 39 shows the absolute error in the continuous velocity models.

This polynomial regression analysis demonstrated the effectiveness of the NH-LIPM in capturing the dynamics of the robot. The R-squared values indicate a high correlation between the regressed parameters and the dynamics. Furthermore, the NH-LIPM showed greater accuracy in predicting the dynamics of the system compared to the LIPM. The model may be further enhanced by employing advanced learning techniques such as Gaussian Processes or neural networks.

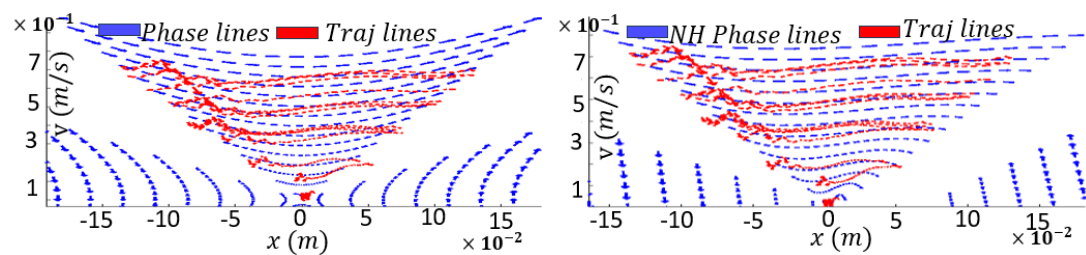


Figure 37: (a) Test data overlaid on the LIPM phase portrait. (b) Test data overlaid on the NH-LIPM phase portrait.

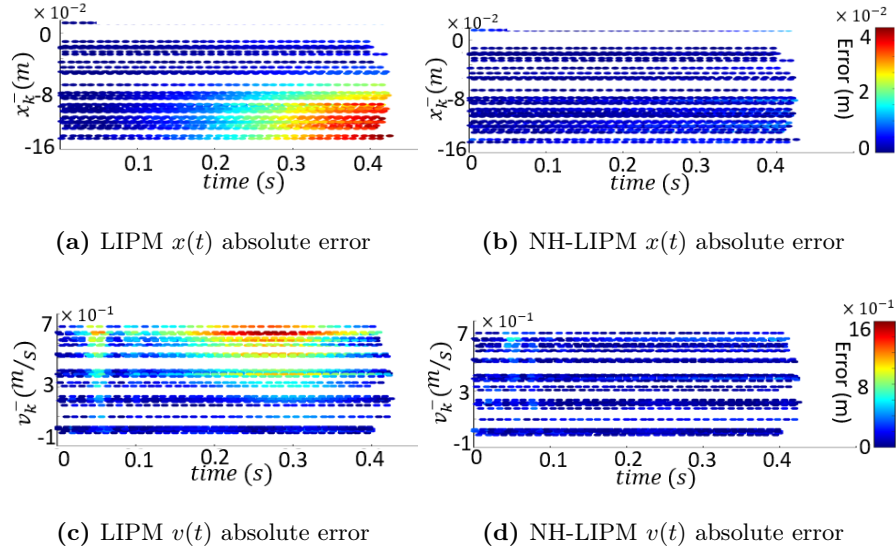


Figure 38: Surface plots of the S2S modeling errors seen in the regression testing.

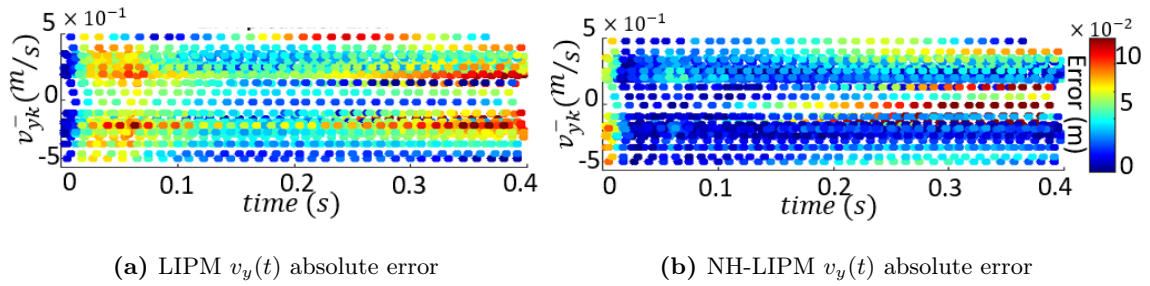


Figure 39: Surface plots of the S2S $v_y(t)$ modelling error seen in the regression testing.

5.5.2 Regression Analysis on External Forces

Regression analysis was conducted to extract the F-NH-LIPM with two separate external forces as control. The joint-damping-control model (**g** in Equation 5.9) is a 7th-order time

polynomial function with squared states and 1st-order damping . The R-squared value for the regression was 0.82. The MAE of the continuous velocity dynamics was 0.008859 m/s. This is an improvement of 57% when compared against the NH-LIPM and 76% against the LIPM. The MAE of the discrete model yielded an improvement of 74% and 75% compared to the NH-LIPM and LIPM, respectively.

The ankle-torque-control model is a 7th-order time polynomial function with squared states and 1st-order damping. The R-squared value for the regression was 0.86. The MAE of the continuous velocity dynamics was 0.012147 m/s. This is an improvement of 60% when compared against the NH-LIPM and 78% against the LIPM. The MAE of the discrete model yielded an improvement of 54% and 77% compared to the NH-LIPM and LIPM, respectively. Figure 40(b) shows the test set continuous $v(t)$ error bands of the three models.

Although the NH-LIPM was improved for the general case, it showed significant prediction errors for the case when the model experiences force inputs. The F-NH-LIPM is able to capture these forces as input parameters in the model, showing improved prediction accuracy. This layered model can be expanded to include other favorable control parameters, such as a toe push-off force which may give the controller greater flexibility and the capability of performing task-specific control.

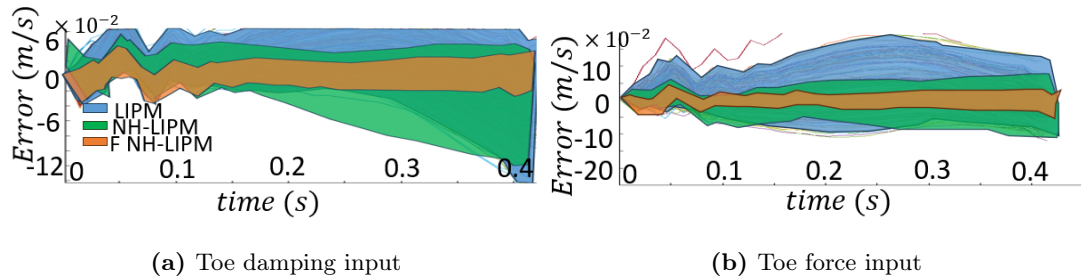


Figure 40: $v_x(t)$ error band comparison.

5.5.3 Stepping Control: tracking a reference velocity

We evaluated the performance of the model and controller framework in tracking a reference velocity profile, denoted as v_{des}^+ . The MAE was utilized as a performance metric, calculated by subtracting the reference velocity from the actual velocity at each step and taking the mean.

When considering the frontal (x) axis, the LIPM stepping controller, as represented by (Equation 5.10), yielded a MAE of 0.116 m/s. In contrast, the NH-LIPM stepping controller, described by (Equation 5.11), achieved a MAE of 0.010 m/s, demonstrating a 91% improvement over the LIPM. These results are illustrated in 41a.

For the lateral (y) axis, velocity tracking was performed on the mid-cycle state $v(T_s/2)$, where $v(T_s/2)$ is employed to solve for v_s in Equation 5.14. The LIPM stepping controller exhibited a mean absolute error of 0.02 m/s, whereas the NH-LIPM stepping controller achieved a mean absolute error of 0.016 m/s. An improvement of 21% over the LIPM can be observed in 41b.

The NH-LIPM improved velocity tracking for both the continuous and discrete models. This improvement is due to the increased model prediction in the NH-LIPM, making it better suited

not only for steady-state control but also for model predictive control requiring task-specific acyclic gaits.

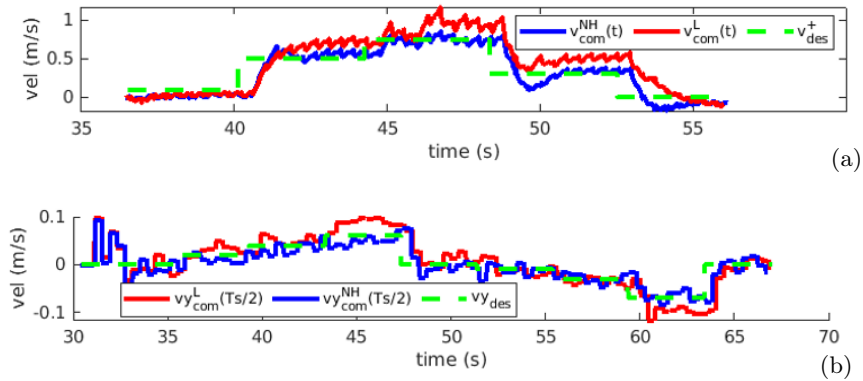


Figure 41: NH-LIPM: Velocity tracking model comparison along the fore-aft (x) direction (a), and the lateral (y) direction (b) .

5.5.4 F-NH-LIPM: tracking a reference velocity with varying ankle damping

The F-NH-LIPM was employed to achieve velocity tracking by adjusting the damping coefficient on the stance ankle joint. The bipedal robot underwent walking experiments at three different speeds, while the toe actuators of the stance foot were augmented with four random damping coefficients (2.85, 11.42, 17.13, and 5.7 Ns/m). By incorporating the damping value as inputs to the F-NH-LIPM model, the system demonstrated superior tracking performance, as depicted in Figure 42, exhibiting a MAE of 0.1 m/s. This represented an improvement of 21% and 16% compared to the performance of the conventional LIPM and NH-LIPM models, respectively.

The F-NH-LIPM controller with ankle-damping input tracks the reference velocity better than the NH-LIPM and LIPM. It contains parameter information that the other models don't have. This extra control parameter can expand the feasible stepping space and walking velocity achieved by the robot, as adding damping to the ankle can result in slow walking with wider steps. However, as seen in Figure 42, even the F-NH-LIPM tracks poorly with high damping values. The model could be further enhanced by adding a higher-order polynomial to the parameter to capture the nonlinearities better.

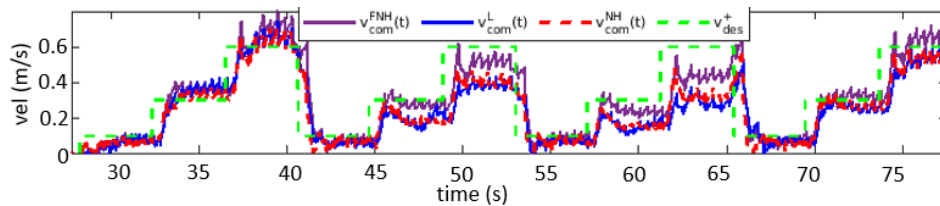


Figure 42: F-NH-LIPM: Velocity tracking at 2.85, 11.42, 17.13, and 5.7 Ns/m ankle damping input.

5.5.5 F-NH-LIPM: velocity and step length control

The F-NH-LIPM was utilized to develop a two-input controller capable of controlling both the desired COM velocity (v^+) and position (x^-). The control inputs for this controller are the foot location at the footstrike (x_k^-) and the ankle torque (τ). Throughout the trial, the desired velocity (v_{des}^+) was maintained at a constant value of 0.2 m/s, while the desired position (x_{des}^-) was modulated.

As shown in Figure 43, the robot's COM velocity, $v_{com}(t)$, closely follows the desired velocity (v_{des}^+), with a MAE of 0.032 m/s. Similarly, the state x^- successfully tracks the trajectory of

the desired position (x_{des}^-), achieving a MAE of 0.016 m. Next, the F-LIPM and F-NH-LIPM

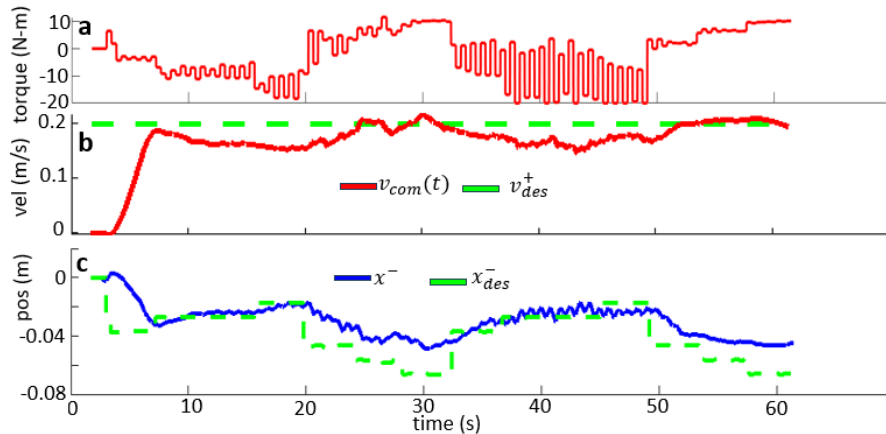


Figure 43: F-NH-LIPM: Using step length and ankle torque (a) for control of v^+ (b) and x^- (c) using the F-NH-LIPM controller.

controllers from Equation 5.29 and Equation 5.31 were used to track a desired reference velocity v_{des}^+ . To test, their ability to regulate both v^+ and x^- four experiments were conducted. Each experiment consisted of tracking a reference velocity ranging from 0 m/s to 0.6 m/s. They also consisted of either a LIPM-based controller, a NH-LIPM-based controller, a F-LIPM controller (Equation 5.29) or a F-NH-LIPM controller (Equation 5.31).

Figure 44 shows the states v^- and x^- of the LIPM trial where the stance toe joints are fully passive. There is a velocity-tracking error at speeds greater than 0.1 m/s. Furthermore, the states x^- and x^+ at the walking speed of 0.6 m/s are -0.12 and 0.12 respectively. Figure 45 shows the states v^- and x^- of the F-LIPM trial where the stance toe joints are actuated according to the control law of Equation 5.32. The velocity tracking is improved, but there is some instability that occurs at a walking speed of 0.6 m/s. The states x^- and x^+ at the

walking speed of 0.6 m/s are -0.17 and 0.05 respectively. The control state x^- increased in magnitude by 0.05, confirming the ability to regulate the control state and the velocity state simultaneously. However, the cost of increasing x^- is that x^+ will increase. This large shift can lead to instability or slippage as the COM is further away from the foot base of support. This instability is visible in Figure 45 where at faster walking speed some velocity oscillation begins to occur. Similarly, Figure 46 shows the states v^- and x^- of the NH-LIPM trial where the

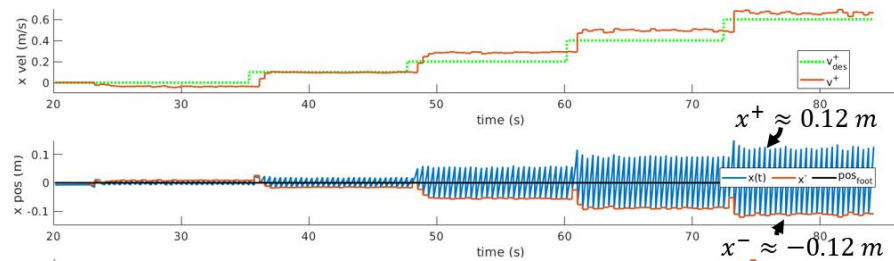


Figure 44: LIPM: Velocity tracking using a LIPM-based controller with passive stance toe joints

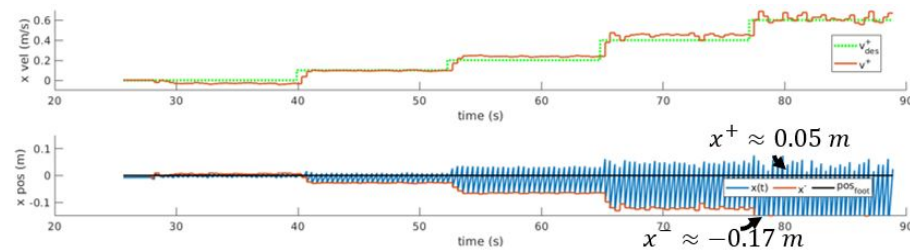


Figure 45: F-LIPM: Velocity tracking using a F-LIPM-based controller with stance toe joint actuation

stance toe joints are fully passive. The states x^- and x^+ at the walking speed of 0.6 m/s are -0.1 and 0.1 respectively. Figure 47 shows the states v^- and x^- of the F-LIPM trial where the stance toe joints are actuated according to the control law of Equation 5.32. The velocity tracking accuracy is reduced at faster speeds. This happens because the state x^+ drops to near

0 m and x^- increases in magnitude to -0.16 m meaning that the robot is more prone to slipping and instability as the COM is so far away from the base of support. Although regulation of v^+ and x^- was achieved, it is important to take caution as some values of the control state x^- may lead to unstable outcomes. Figure 48 shows the left toe pitch and roll torque values during the trial. To walk at a speed of 0.6 m/s the toe pitch torque peaks at near 100 n-m. Caution must also be placed on the magnitude of this torque as too large of a torque may be infeasible for some toe or ankle actuators.

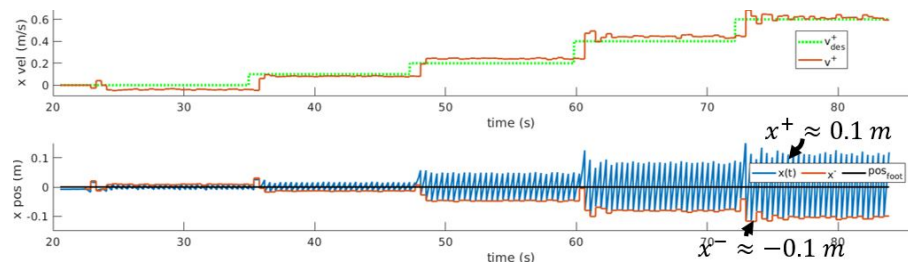


Figure 46: NH-LIPM: Velocity tracking using a NH-LIPM-based controller with passive stance toe joints

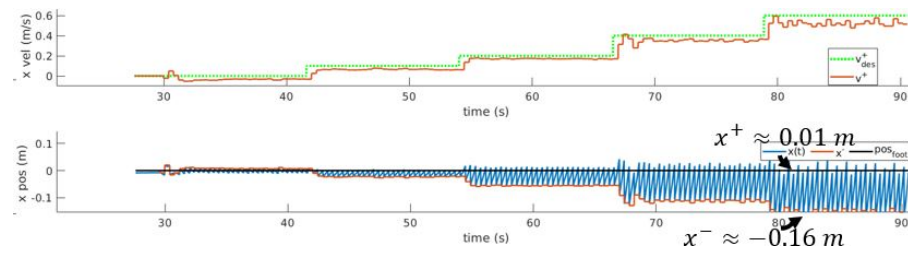


Figure 47: F-NH-LIPM: Velocity tracking using a F-NH-LIPM-based controller with stance toe joint actuation

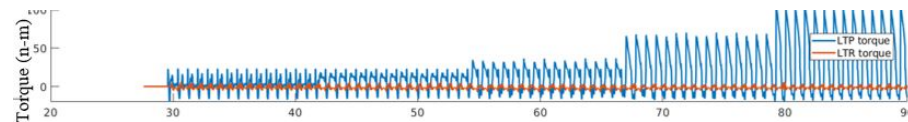


Figure 48: F-LIPM: Toe torques displayed during the trial.

These results demonstrate the effectiveness of the F-NH-LIPM controller in accurately controlling both v^+ and x^- . This additional control input can help navigate a constrained environment where several robot states must be achieved for proper navigation. The controller is in its most basic form and can be improved with feedback control to achieve more stability and fewer steady-state errors. It is worth noting that not every control pair combination will yield a feasible response, and as such, it is essential to have boundary conditions on the controller inputs.

5.5.6 Walking Stabilization Using F-LIPM Control

Forward walking simulations were done where LIPM-based stepping with a stepping period of 0.55 seconds was applied to test the F-LIPM controller's ability to stabilize the dynamics of an unstable walking controller. Although the LIPM does a decent job at stabilizing the dynamics of the system at the nominal period of 0.45 seconds, its performance decreases when the period is increased to 0.55 seconds. Figure 49(a) shows an instance during the simulation where the robot loses its lateral stability and begins swaying to the sides and away from the desired dotted straight line. The COM y-position data shown in Figure 50 illustrates how the COM begins to drift and display oscillatory behavior as the walking velocity increases. The red and green circles in the y-position and y-velocity plots show some of the most asymmetric states which largely deviate from those displayed in stable walking.

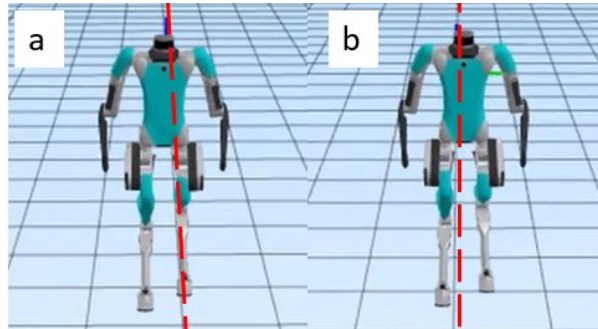


Figure 49: F-LIPM Stability: LIPM stepping (a) and F-LIPM stepping (b) applied to walking along the straight dotted line with a stepping period of 0.55 seconds leading to instability in the LIPM stepping controller.

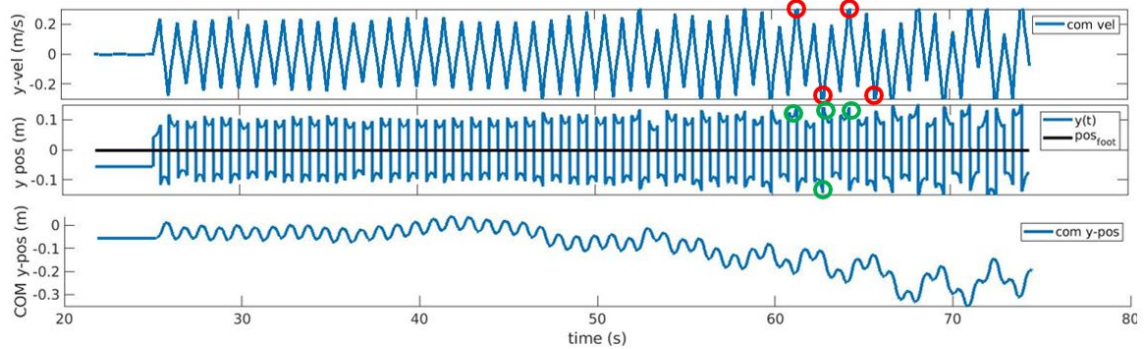


Figure 50: Lateral (y) state data of walking simulation using LIPM stepping. The red and green dots show some asymmetric v_y^- and y^+ states that arise due to instability.

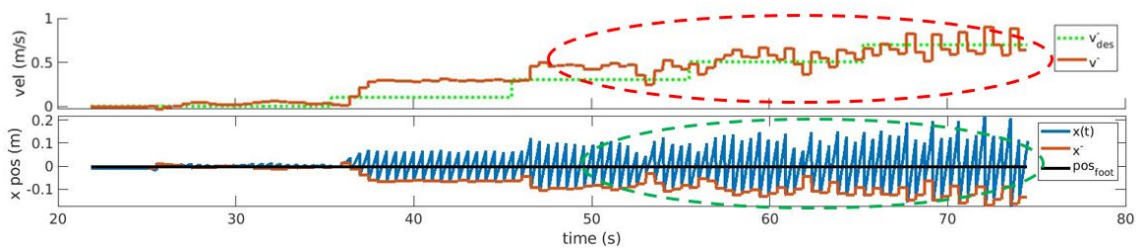


Figure 51: Frontal (x) state data of walking simulation using LIPM stepping. The red and green dotted circles show some areas where asymmetric v_x^- and x^+ states arise due to instability.

Once F-LIPM stepping is applied, most of the instability goes away as shown in Figure 49(b) where the robot walks closer to the dotted straight line. In Figure 52 the COM y-position data shows some deviations at the beginning of the trial which may have been due to instability during the stand-to-walk transition. This instability is also seen in the y-position and y-velocity state data outlined by the red and green circles. However, stability is displayed shortly afterward. The asymmetric velocity spikes displayed in the LIPM trial are gone using F-LIPM stepping. Even at the faster walking speeds the robot maintains stability. In Figure 53, the initial region of instability also affects the states v_x^- and x^+ . However, the robot regains its stability afterward as indicated by the reduction of oscillatory behavior along the frontal axis as well as fewer deviations in the states x^- and x^+ .

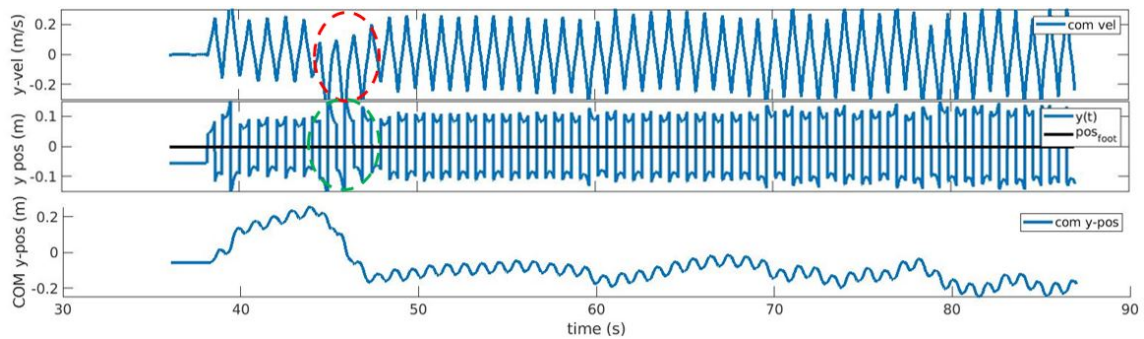


Figure 52: Lateral (y) state data of walking simulation using F-LIPM stepping.

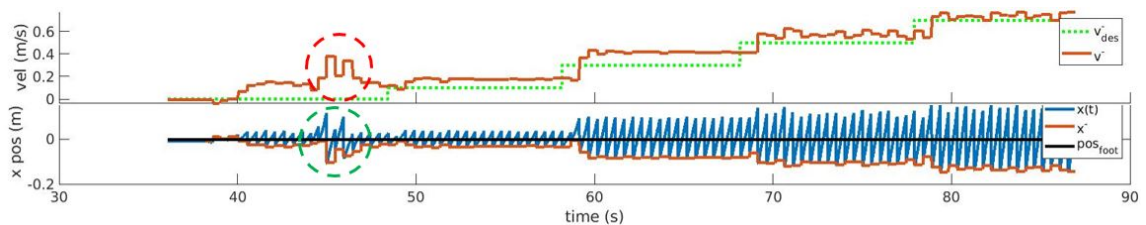


Figure 53: Frontal (x) state data of walking simulation using F-LIPM stepping.

These results validate the use of F-LIPM stepping to enhance a model-based controller that contains modeling errors. The F-LIPM can be applied directly as a controller to use for bipedal walking or it can be used as a starting point to then extract or improve the model. Obtaining a better model will reduce the amount of torque applied by the toe actuators. Furthermore, an improved model will more efficiently use the passive natural dynamics for walking.

5.5.7 Real-Time Adaptation of Linear S2S Model

Implementing the RLS controller involved several tuning trials to determine suitable parameter values. For the frontal axis, the forgetting factor (λ) was set to 0.95, and the regularization parameter (Δ) was set to 0.1. Along the lateral axis, λ was adjusted to 0.97, and Δ was set to 0.99.

Figure 54 illustrates the velocity tracking performance of the RLS controller compared to the LIPM controller. In the forward direction, the MAE achieved by the RLS controller was 0.0604 m/s, which is 43% lower than the MAE obtained by the LIPM controller. In the lateral direction, the MAE achieved by the RLS controller was 0.0648, an improvement of 21% over the LIPM. During lateral walking using the LIPM controller, the model discrepancy caused the robot to tip over, as shown in Figure 54. These results demonstrate the superior tracking capabilities of the RLS controller in comparison to the traditional LIPM controller.

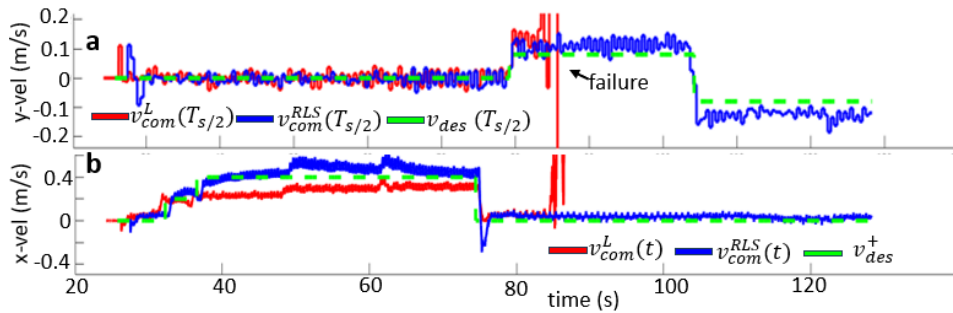


Figure 54: RLS controller: Lateral (a) and forward (b) walking velocity tracking comparison between the adaptive RLS and the LIPM stepping controllers. Failure occurs during lateral walking with LIPM controller.

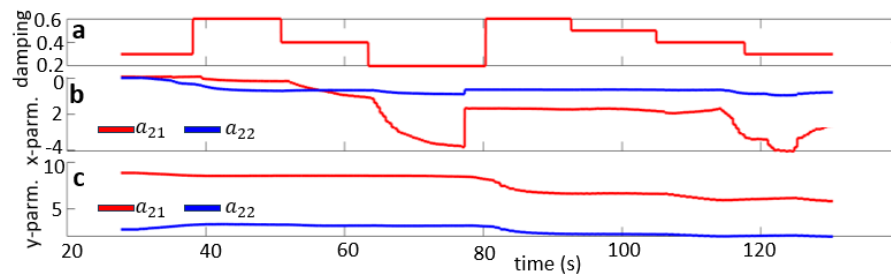


Figure 55: RLS controller: (a) Deliberate damping variations were used in the trial. (b)-(c) the RLS controller performed online tuning of the model parameters

Additionally, Figure 55 showcases the adaptive tuning of the RLS parameters a_{21} and a_{22} along the x and y axes, respectively, as the damping is varied. This adaptive tuning capability allows the RLS controller to dynamically adjust its model parameters to handle changes in the damping conditions effectively. Parameter convergence is achieved in less than 5 seconds on average.

These findings highlight the effectiveness of the RLS controller in achieving improved velocity tracking and its ability to adaptively tune its parameters based on the changing conditions of

the system. With an adaptive model, a "close-enough" representation of the robot's dynamics can be extracted, which can then be utilized to develop stepping controllers. This eliminates the need for a time-consuming trial-and-error process typically associated with developing accurate models for walking controllers.

5.5.8 Hardware Results: Extracting the Non-homogeneous LIPM

Two regression analysis experiments with different robot heights were carried out. The first experiment was using a COM height of 0.86 m (vertical distance from the stance toe pitch joint to the COM). The second experiment was using a COM height of 0.92 m. In the first experiment regression analysis on hardware yielded an R-squared value of 0.87 with a MAE of the continuous NH-LIPM of 0.0156 m/s, which is an improvement of 55% compared to the LIPM. The MAE of the discrete NH-LIPM was 0.01369 m/s, an improvement of 75% compared to the LIPM.



Figure 56: v_{des}^+ tracking hardware experiment using the NH-LIPM controller.

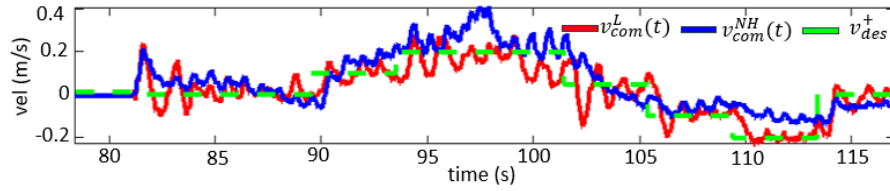


Figure 57: Hardware: Velocity tracking comparison between NH-LIPM and the LIPM performed on Digit.

The results of the velocity tracking experiment are shown in Figure 57. The tracking MAE of the NH-LIPM controller was 0.043 m/s, an improvement of 48% compared to the LIPM controller. This preliminary experiment shows the effectiveness of the regression-based NH-LIPM on hardware. The tracking MAE was reduced when compared to the traditional LIPM controller. The NH-LIPM controller, however, displayed less steady-state stability. The stability will be addressed via feedback control and controller tuning in future studies.

For the second experiment, the low-level control of the robot was significantly modified for stabilization improvements which potentially led to the robot dynamics further deviating away from the LIPM modeled dynamics. A polynomial function of the form *poly113* from MATLAB was fitted to the training data to map the initial states of the robot to the approximated time-dependent continuous dynamics. The regression fit gave an r-squared value of 0.915 indicating a good fit. Figure 58 shows the velocity prediction errors of the testing data set using the LIPM (a) and the NH-LIPM (b). The MAE of the continuous LIPM velocity prediction was 0.142 m/s and the MAE of the continuous NH-LIPM velocity prediction was 0.020 m/s. This is an improvement of 86% using the NH-LIPM.

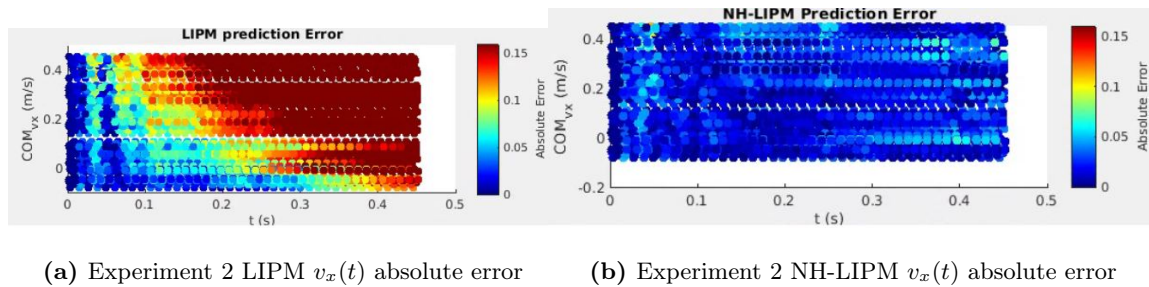


Figure 58: Surface plots of the continuous S2S $v_x(t)$ modelling error seen in the regression testing.

Both models were applied on a forward-walking experiment shown in Figure 59. The trial consisted of following a reference velocity of 0.1 m/s for a distance of approximately 3.0 m. It is clear from the figure that the LIPM rushes past the desired trajectory (red dot) indicating a large tracking error. Figure 60 shows the COM position and velocity data collected during the LIPM-stepping controller trial. The tracking MAE displayed during the trial was 0.22 m/s. The NH-LIPM improved the tracking by roughly 90% with a tracking MAE of only 0.02 m/s.

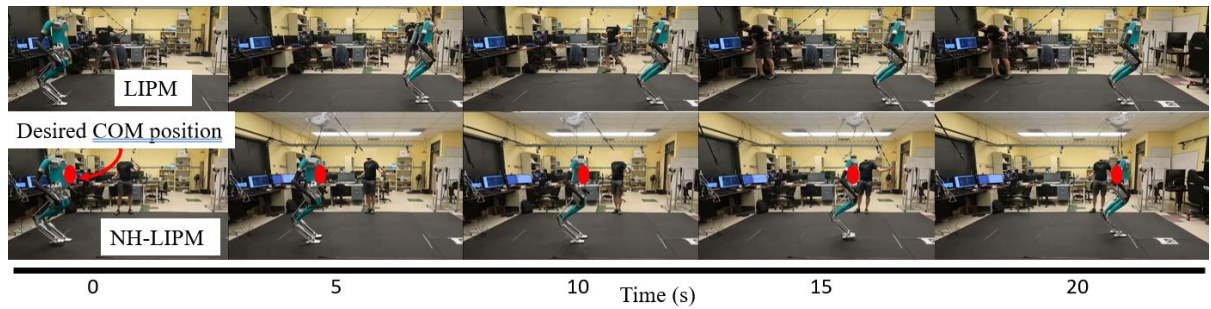


Figure 59: Forward walking experiment testing the use of the LIPM-based controller and the NH-LIPM-based controller to follow a velocity trajectory of 0.1 m/s for a distance of approximately 3.0 m.

The red dot shows the position of the desired trajectory.

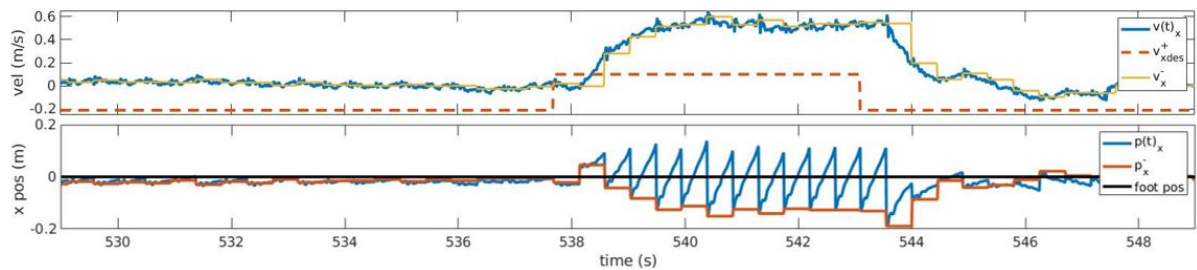


Figure 60: LIPM: COM position and velocity data during the velocity tracking trial.

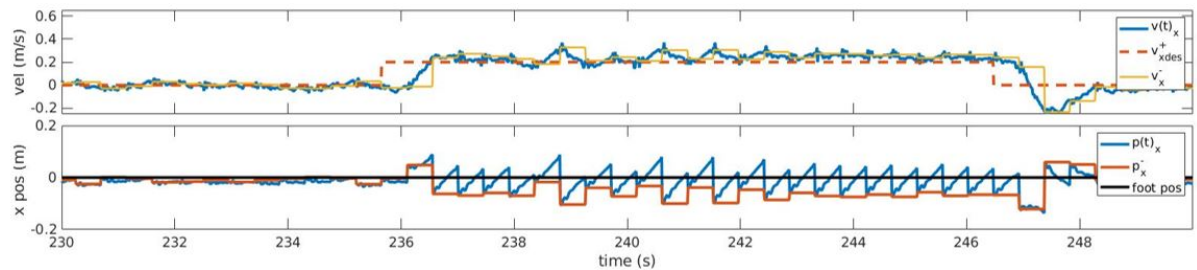


Figure 61: NH-LIPM: COM position and velocity data during the velocity tracking trial.

5.5.9 Hardware Results: Continuous Velocity Tracking Using F-LIPM Control

A velocity tracking trial was conducted to test F-LIPM control for continuous velocity tracking using toe torques (Equation 5.26 & Equation 5.27). Figure 62 shows images from the

trial comparing the proposed F-LIPM controller to the LIPM controller. The red dot shows the desired COM state during the tracking trial. The F-LIPM controller's model improvement can be appreciated by noticing the reduction in walking speed. Figure 63 shows an improvement of 73% in velocity tracking with a MAE of 0.06 m/s. Furthermore, stepping stability is achieved as indicated by the uniform and repetitive states x^- , x^+ , and v^- .

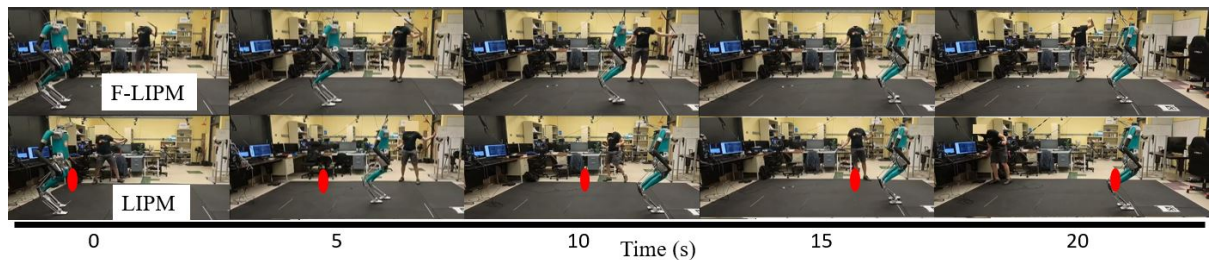


Figure 62: Forward walking experiment testing the use of the LIPM-based controller and the NH-LIPM-based controller to follow a velocity trajectory of 0.1 m/s for a distance of approximately 3.0 m. The red dot shows the position of the desired trajectory.

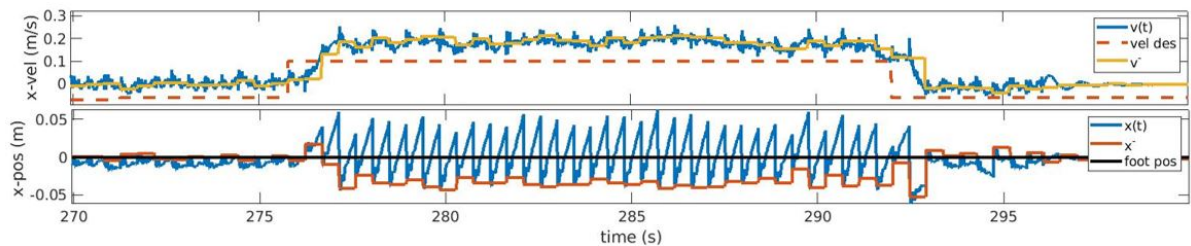


Figure 63: F-LIPM: COM position and velocity data during the velocity tracking trial.

5.6 Conclusion and Future Work

Our findings demonstrate the enhanced performance and effectiveness of the NH-LIPM in predicting the dynamics of the bipedal robot Digit. The NH-LIPM introduces a time and

state-dependent component that enhances the accuracy of the robot's S2S dynamics. By extracting the solution to the non-homogeneous term through regression analysis, we account for the time/state-varying aspects of the robot's behavior, resulting in a more comprehensive model that better captures the real-world dynamics. Hardware testing yielded promising results.

The NH-LIPM was supplemented with a forced term, with ankle damping and torque as input parameters, to develop the F-NH-LIPM. The F-NH-LIPM controller accurately and simultaneously controlled the COM velocity and COM position of the robot with respect to the stance foot. This refined model enables a more effective control strategy facilitating navigation in environments that require simultaneous control of multiple states, such as foot placement and COM velocity.

The RLS algorithm was used to adaptively tune the parameters of the particular solution to the NH-LIPM. The RLS controller with forgetting, effectively modulated the parameters and reacted to deliberate changes in the robot dynamics like height, step width, and ankle damping. A velocity-tracking experiment showed the RLS controller's ability to track a reference velocity with changing ankle damping.

Overall, the findings of this study support the use of the NH-LIPM as a valuable model for understanding and controlling bipedal walking. The incorporation of non-homogeneous terms through regression analysis improves the accuracy of the model, enabling more precise predictions of the COM position and velocity.

Future work will center on implementing the discussed methodologies on hardware, with a key focus on enhancing the stability of the hardware NH-LIPM. To bolster the predictive

capabilities, a more extensive training set will be utilized, augmenting the convex hull of the regression function. Experiments will be conducted to examine the NH-LIPM controller on lateral stepping. Additionally, the integration of ankle damping and torque as control inputs to the F-NH-LIPM will be explored on hardware. Lastly, to validate the model's practical applicability, a model predictive stepping controller will be deployed, employing the F-NH-LIPM to navigate challenging terrains with obstacles. This comprehensive evaluation of real-world hardware will further enhance the model's efficacy and facilitate its integration into bipedal robotic research.

CHAPTER 6

OPTIMAL ASYMMETRIC STEPPING CONTROL OF DIGIT USING ANALYTICALLY IDENTIFIED MODELS

6.1 Introduction

During stable symmetric walking, the primary focus is on ensuring stability, and foot placement precision is of lesser importance. However, when a bipedal robot needs to navigate constrained environments, achieving precise foot placement becomes crucial. The robot must maintain its balance, avoid actions that could lead to trips or falls, and precisely control its foot placement. Traditionally, one common approach for walking in constrained environments is to plan a trajectory offline. This trajectory planning involves considering the full-body dynamics of the robot and constrained conditions. The robot then performs tight trajectory tracking of the pre-planned motion on hardware, as demonstrated in prior work [47].

Another approach involves creating a library of trajectories that guide the robot effectively towards its desired goal while avoiding obstacles. An algorithm selects the optimal trajectory based on obstacle location and size, as observed in a different study [48].

More recently, Model Predictive Control (MPC) strategies have gained popularity, allowing online trajectory planning. These strategies derive closed-form solutions from dynamic models, enabling the real-time control of quadrupedal robots [49]. Trajectory optimization (TO) has also become prominent in controlling complex-legged systems. TO helps predict and plan a

robot's future dynamics, considering parameters like the center-of-mass (COM) positions and control inputs. Some approaches leverage MPC TO to design the robot's motion and behavior while adhering to feasible constraints that can be executed on hardware to achieve the desired behavior [50], [51].

In this chapter, we explore Digit's asymmetric stepping capabilities, which are essential for effective obstacle avoidance. We evaluate Digit's ability to recover from push disturbances on hardware, allowing the controller to transition from symmetric stepping to an asymmetric response when external disturbances occur. We introduce the application of MPC for real-time optimization of the robot's future states and corresponding control inputs. Our planning horizon spans three steps, and we employ both single-stage and multi-stage optimization techniques. Additionally, we enhance this framework by integrating an analytical function that establishes a relationship between feasible control inputs and the robot's current state. This innovative approach reduces the number of optimization instances required within the multi-stage MPC and significantly improves the robot's ability to navigate obstacles successfully.

6.2 Push Disturbance Recovery

Three experiments were conducted to test the stepping controller's ability to recover from an external push perturbation applied from the back, front, and side. For these trials, the robot was commanded to walk in place using NH-LIPM stepping control as a user applied a force by pushing the robot at the base. Figure 64 shows an experimental trial where the robot is walking in place at frame (a) and is then pushed forward along the red arrow at frame (b). To recover, the robot takes a step forward as shown in frame (c) and the robot regains stability by frame

(f). Figure 65 shows the COM velocity and position data. The red arrow shows the increase in forwards velocity as a result of the forward push. The yellow arrow on the position plot depicts the long step taken indicated by a more negative x^- state to recover from the push.

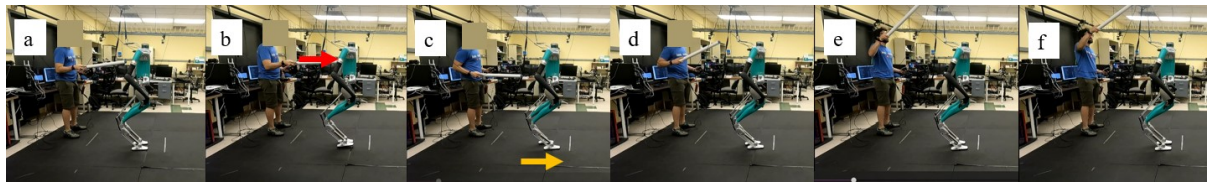


Figure 64: Forward Push Recovery

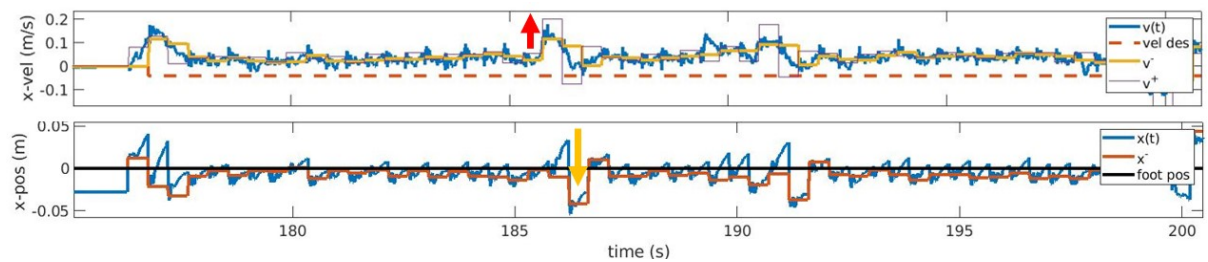


Figure 65: Forward push recovery data Showing an increase in velocity (red arrow) followed by a long step forward (yellow arrow).

Figure 66 shows an experimental trial where the robot is walking in place at frame (a) and is then pushed backward along the red arrow at frame (b). To recover, the robot takes a step backward as shown in frame (d) and the robot regains stability by frame (f). Figure 67 shows the COM velocity and position data. The red arrow shows the increase in backwards velocity as a result of the backward push. The yellow arrow on the position plot depicts the long step

backward taken to recover from the push. The backward step is indicated by a more positive x^- state.

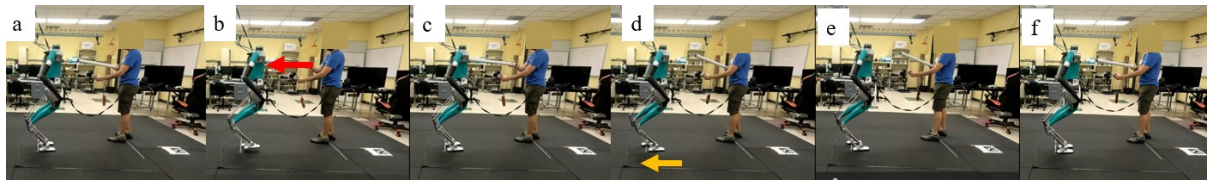


Figure 66: Backward Push Recovery

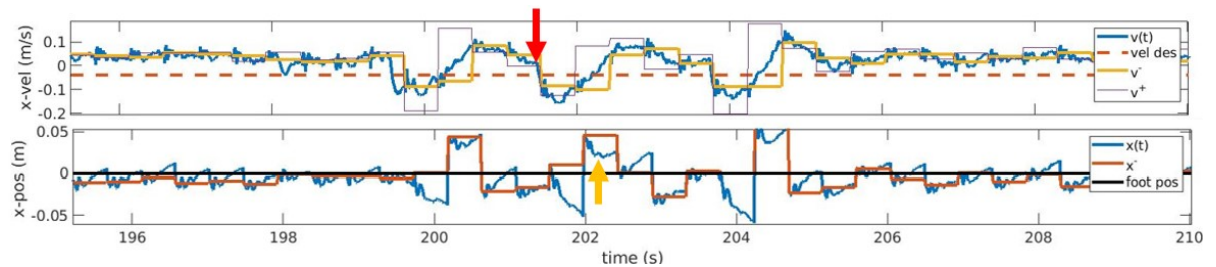


Figure 67: Backward push recovery data Showing an decrease in velocity (red arrow) followed by a long step backward (yellow arrow).

Figure 68 shows an experimental trial where the robot is walking in place and is then pushed sideways to its right along the red arrow at frame (a). To recover, the robot takes a wide step to its right as shown in frame (b), followed by another wide step to the left in frame (c) and a short step to the right of its COM in frame (d) and the robot regains stability by frame (e). Figure 69 shows the COM velocity and position data. The red arrow shows the decrease in leftward velocity as a result of the rightward push. The yellow arrow on the position plot depicts the wide step taken to the right and the short step taken to the left.

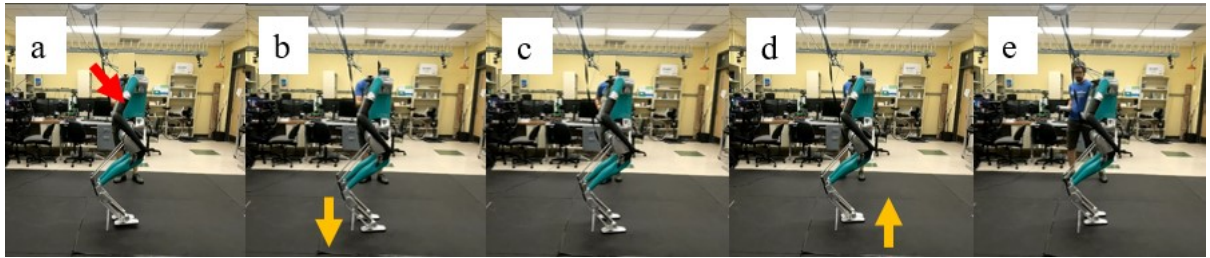


Figure 68: Sideways Push Recovery

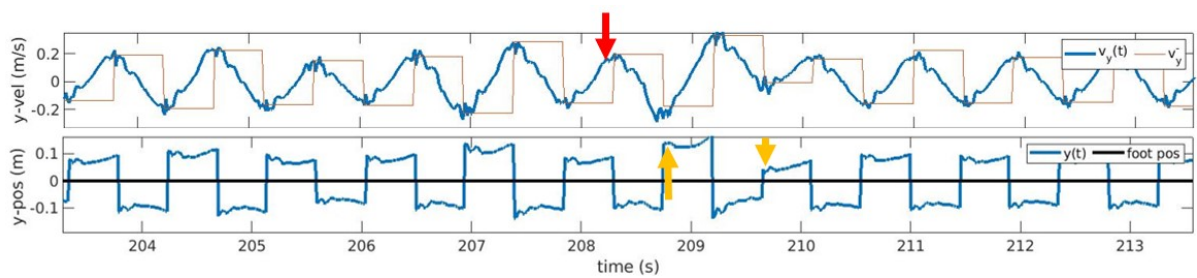


Figure 69: Rightward push recovery data Showing an increase in rightward velocity (red arrow) followed by a wide rightward step in frame (b), a wide step leftward in frame (c) and a short step rightward in frame (d)

6.3 Frontal Asymmetric Stepping

In a real-world setting, a robot must traverse its environment in a manner that reduces the likelihood of tripping and falling. In an environment with open spaces, symmetric stable walking using model-based stepping with feedback can guarantee stability. However, in an environment where the space is constrained by objects and areas where the robot cannot step on, symmetric walking might not be enough. The analytical dynamic models can be used to predict the dynamic behavior of the robot regardless of motion symmetry.

Figure 70 shows an example of an asymmetric stepping controller used to step over an obstacle. The stepping instructions are as follows. As the robot approaches an obstacle at a distance d' with a constant walking velocity such that $v_0^- = v_1^-$ the next foot placement, x_1^- , must be shorter than the nominal symmetric stable walking x_0^- state to accelerate the COM forward to states x_1^+ and v_2^- . The next foot placement x_2^- must be properly placed such that it clears the obstacle yet does not slip and that v_3^- is positive, indicating that the robot is not moving backward towards the obstacle.

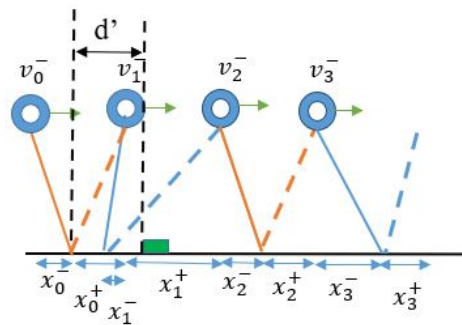


Figure 70: Asymmetric stepping control for stepping over an obstacle.

An experiment was conducted to test the model-based controller's ability to generate asymmetric stepping. The F-LIPM stepping controller was used for this trial. The experiment consisted of walking the robot at a constant walking velocity along a straight path in the direction of an obstacle. The walking velocity was set at 0.25 m/s. The edge of the obstacle was marked by a strip of tape (shown in Figure 71) on the ground and the obstacle was assumed to be 10 cm in length starting from the tape. Once the distance of the edge of the obstacle was

within a distance $d' = 10\text{cm}$ away from the stance foot, the asymmetric stepping controller was activated by the user with the push of a button on a gamepad controller. Figure 71(a) shows the state when the asymmetric stepping controller is activated. The following footstep location was manually chosen to be in line with the previous footstep as shown in Figure 71(b). This allows the COM of the robot to accelerate and shift forward so that the next step can clear the obstacle. The next step of the asymmetric stepping controller is dictated by optimizing the foot placement such that the step length is at least 30cm, the predicted velocity state v_3^- is greater than zero and x_2^+ is greater than 0.05m. The NLOPT library in C was used for the optimization. The optimal foot placement is shown in Figure 71(c). COM position and velocity data was



Figure 71: Asymmetric stepping hardware experiment. (a) the robot takes a nominal step size followed by (b) a short asymmetric step to drive the COM forward, then (c) a long step is taken to slow down the velocity of the COM and step over the obstacle on the floor and finally (d) the robot successfully clears the obstacle.

collected during the experiment and is shown in Figure 72. The top plot shows the continuous COM velocity in blue and the discrete velocity state v^- in yellow along with the desired walking velocity in orange. The bottom plot shows the continuous position of the COM, $x(t)$, with

respect to the stance foot and the discrete state x^- in blue and orange respectively. The black line is at the zero position indicating the position of the stance foot. A negative x^- indicates a footstep in front of the COM and a positive x^- indicates a footstep behind the COM. The robot begins to walk forward at a time of about 180 seconds and transitions into a stable symmetric gait depicted by the nearly constant and repeating v^- and x^- states. The asymmetric controller is activated at the state s_0^- . At the state of s_1^- the footstep is behind the COM of the robot as seen in state x_1^- in the bottom plot of Figure 72. The COM velocity increases to roughly 0.8 m/s as seen in the top plot at v_2^- . To slow down the robot's COM and clear the obstacle, the optimal footstep state x_2^- is employed. This is a long footstep as depicted by the large negative value of x_2^- in the plot. After the obstacle is cleared, the COM continues to move forward with an approximate velocity of $v_3^- = 0.6$ m/s as shown in the plot. The model-based stepping controller is again activated after the obstacle is cleared to transition the robot into a stable gait with a slower walking speed.

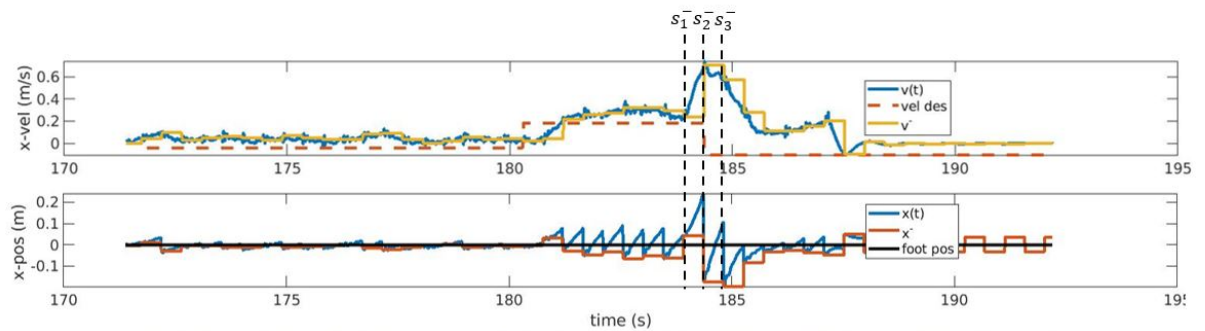


Figure 72: Data from the hardware experiment using asymmetric stepping control to step over an obstacle. The state s_1^- depicts the short step taken to increase the velocity of the COM. The state s_2^- depicts a velocity increase to over 0.6 m/s and therefore the the robot uses a larger control state x_2^- to slow id town. The state s_3^- depicts another large control state x_3^- however v_3^- has slowed down.

6.4 Lateral Asymmetric Stepping

Similar to asymmetric stepping along the frontal direction, asymmetric stepping control can be applied to lateral stepping. This can specifically be useful for avoiding obstacles on the sides of the robot while walking laterally. One big distinction between frontal and lateral asymmetric stepping is that along the lateral direction, the range of motion of the legs is more limited since their motion is along the same line. Therefore, a different strategy for side-stepping must be employed.

Figure 73 depicts the proposed strategy. The robot starts with a walk-in-place stable symmetric gait where the footstep locations are the points p_{-1} and p_0 . At this symmetric gait, the velocity states $v_0^- = -1 \cdot v_1^- = v^*$. Similarly, $x_{-1}^+ = -1 \cdot x_0^+ = x^{*+}$. For the COM to move towards the left, a wide step towards the right will cause the velocity v_2^- to increase such that $v_2^- > v^*$ and $x_1^+ > x^{*+}$. The following step would apply a control, x_2^- , such that the symmetric velocity, $-1.0 \cdot v^*$, is achieved at v_3^- . If adequate foot placements are achieved without violating collision constraints, the robot should transition back into walk-in-place symmetric stepping. The point p_3 is now the footstep location of the right leg and it has moved further left from the point p_{-1} indicating the robot has successfully moved to the left.

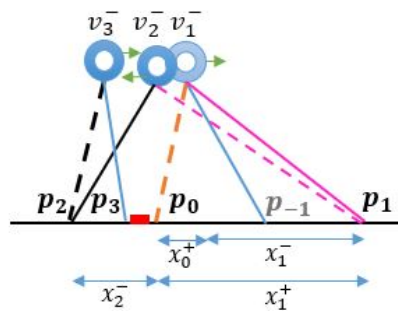


Figure 73: Lateral asymmetric stepping control used to step over an obstacle laterally.

Assuming the step-to-step dynamics can be expressed analytically, we can derive an expression,

$$H(p_0, x^{*+}, v^*, x_1^-) \approx p_3 \quad (6.1)$$

that relates the control input, x_1^- , to the final position of the right foot after three steps. The accuracy of Equation 6.1 depends on the accuracy of the S2S dynamics analytical model. The accuracy of the LIPM and the lateral NH-LIPM were tested in simulation. For the experimental trial, the robot was walked in place with a step width of 0.2 m as shown in Figure 74(a). The dotted black lines show the position of the right and left foot during walk-in-place symmetric stepping and can be related to points p_{-1} and p_0 from Figure 73. An asymmetric step that is 0.1 m further left than p_{-1} is taken shown in Figure 74(b). For the second step, shown in Figure 74(c), x_2^- is chosen using model-based stepping to drive the v_3^- to the symmetric state v^* . Finally the robot should theoretically be back to a symmetric gait after the third step in Figure 74(d). However, due to inaccuracies in the model-based stepping controller, two more steps are taken to finally drive the robot back to walk-in-place symmetric walking as shown in Figure 74(f).

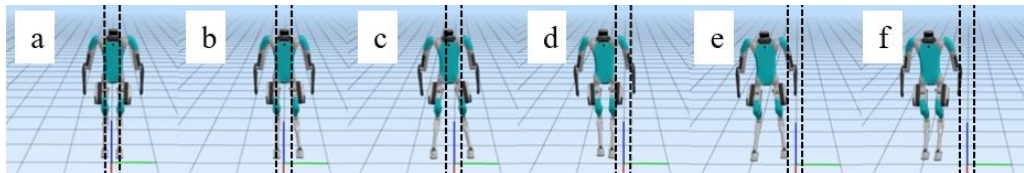


Figure 74: Lateral asymmetric stepping simulation.

To compare the LIPM and the NH-LIPM, Equation 6.1 was derived for the LIPM, H^L , and the NH-LIPM, H^N . Plugging in the values of the input parameters yields the following values for p^3 .

$$p_3 \approx H^L(p_0, x^{*+}, v^*, x_1^-) = -0.1885 \quad (6.2)$$

$$p_3 \approx H^N(p_0, x^{*+}, v^*, x_1^-) = -0.1513 \quad (6.3)$$

The actual value of p_3 was

$$p_3 = -0.1257 \quad (6.4)$$

giving an error of 0.0628 m for the LIPM and 0.0372 m for the NH-LIPM. This improvement using the NH-LIPM is expected as this model better captures the dynamics of the robot and is therefore better suited for developing asymmetric stepping controllers to step over obstacles laterally.

6.4.1 Lateral Asymmetric Stabilization Using Toe Torques

In some cases, the input x_1^- may yield infeasible results. Such is the case shown in Figure 75. The simulation shows that a large x_1^- at step (b) can produce too large of a velocity at step (c) that the input x_2^- is not enough to drive the velocity back to the stable symmetric velocity v^* . The stepping controller tries to further slow down the velocity of the COM by placing the left foot as far right as possible, reaching the foot collision limit as shown in Figure 75(d). The instability further propagates leading to a fall at (g). To add stability to the asymmetric

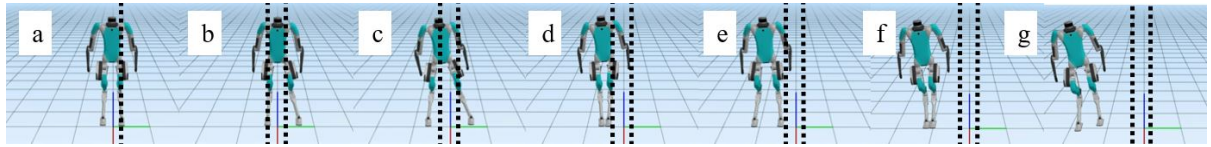


Figure 75: Instability arises when the lateral velocity is too high.

stepping controller, an ankle torque, τ_a is applied on the stance foot. The ankle torque adds resistance to slow down the lateral velocity as the COM moves toward the right. In Figure 76(c) the right toe motors insert a roll torque of 20Nm to slow down v_3^- . In (d), the left toe motors insert a roll torque of 20Nm to slow down v_4^- . Finally, in (f), the robot is back to symmetric walking.

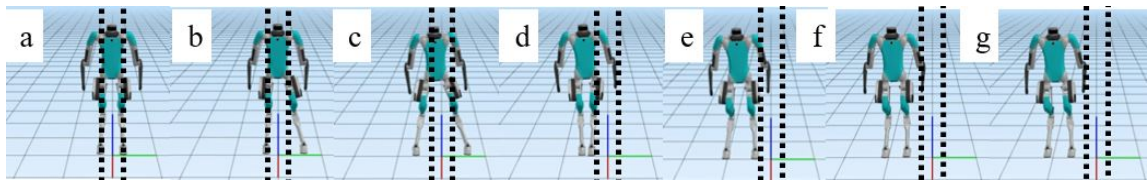


Figure 76: Stabilization of the lateral motion can be stabilized using toe/ankle torques.

Figure 77 shows the COM lateral position for the trial where the stance leg toe joints are completely passive (a) and where the bias torque is applied (b). From the plots, we see that even after three steps, the robot with passive toe joints continues to move toward the right eventually leading to a fall. The robot with actuated toe joints is able to slow down the right-moving velocity of the COM and regain its cyclic stability in six steps after the asymmetric step.

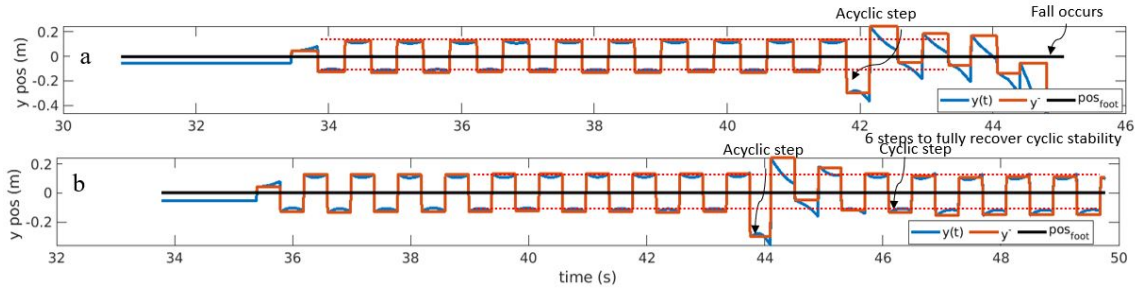


Figure 77: (a) Simulation data of lateral asymmetric stepping with instability leading to a fall. (b) Simulation data of lateral asymmetric stepping where instability is stabilized.

6.4.2 Lateral Asymmetric Stepping on Hardware

An experiment was conducted to test the asymmetric lateral stepping approach on hardware. The frontal and lateral NH-LIPM stepping was used to walk the robot in place as shown in Figure 78(a). The step width is set to 0.2 m and is marked by the dotted yellow lines in the figure. The nominal states of the symmetric gait are represented as v^* , x^{*-} , and x^{*+} . The user commands the robot to take a wider step with the push of a button on the gamepad overriding the stepping controller commands. At this time the robot takes a step that is 0.08 m wider than the nominal step width as shown in Figure 78(b). The state x^- at this step is equal to -0.18 m, as pointed out by the arrow in Figure 80, producing subsequent asymmetric dynamics. The stepping controller is then reactivated and a step to the right is taken to drive the state of the robot v_3^- back to the nominal v^* as shown in (c). However, this step does not stabilize the robot and an additional two steps, (d) and (e), are taken to reach stability in step (f). Figure 79 shows a trial when the robot successfully regains its stability in three steps.

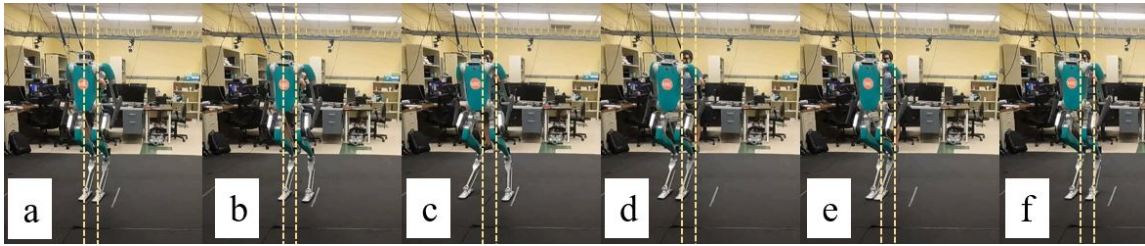


Figure 78: Asymmetric lateral stepping hardware experiment where a wide step is taken at (b) and stabilizes after four steps at (f).

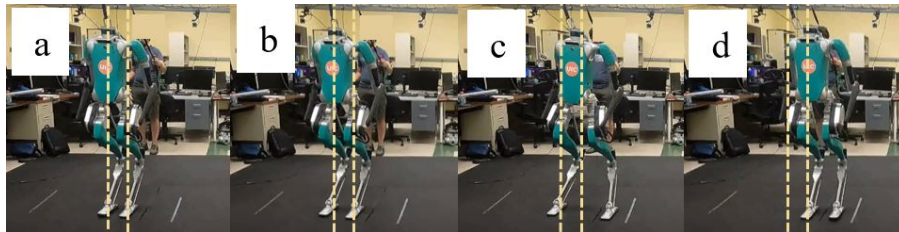


Figure 79: Asymmetric lateral stepping hardware experiment where a wide step is taken at (b) and stabilizes after two steps at (d).

Figure 80 shows the state data collected from an experimental trial. The nominal states are $v^* = 0.16m/s$, $x^{*-} = 0.10m$, and $x^{*+} = 0.10m$ as shown in the y -vel and y -pos data while the robot walks in place with a stable gait. The lateral asymmetric stepping controller is activated five times in this trial. The arrows in the y -com plot show the desired translation direction. The first two sub-trials translate the robot first to the left and then to the right after taking asymmetric steps that are $0.03m$ wider than the nominal $0.10m$. The following three steps translate the robot's COM further because a control state that is about $0.08m$ greater than the nominal $x^{*-} = 0.1m$ is used.

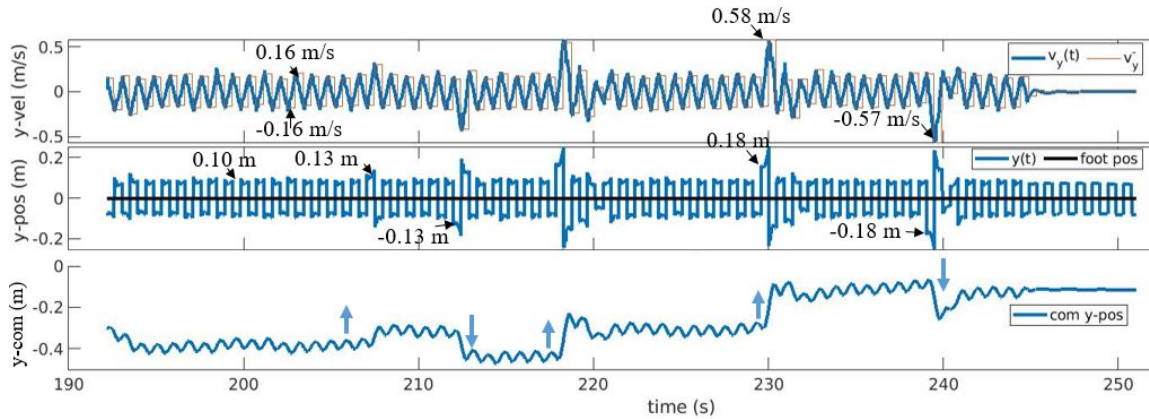


Figure 80: Hardware experiment data of the lateral (y axis) COM states.

6.5 Model Predictive Control

Model Predictive Control (MPC) is an advanced control methodology that has gained substantial interest in robotics, with a particular focus on controlling bipedal robots. MPC offers a powerful framework for optimizing control actions over a finite time horizon while accounting for complex system dynamics and constraints. MPC is a strategy that formulates an optimization problem over a finite time horizon, aiming to minimize a cost function while satisfying dynamic system constraints. In the realm of bipedal robotics, this predictive control strategy has exhibited remarkable capabilities in addressing challenges such as dynamic balance, trajectory tracking, obstacle avoidance, and adapting to varying terrains.

In this study, MPC is used to find the optimal foot placement necessary to step over a ground obstacle in the line of motion of the robot. The framework involves formulating an optimization problem aimed at minimizing a cost function while adhering to feasible state and dynamics constraints. The cost function is utilized to minimize deviations in both velocity and

step length from the nominal values displayed during stable walking, employing a sum of squared differences function.

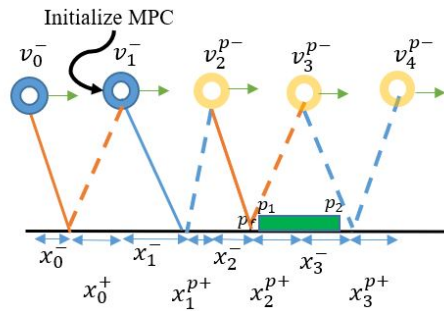


Figure 81: MPC initialization used to plan control states to step over an obstacle in three steps.

MPC is applied within a finite horizon, with a window length of three steps as shown in Figure 81. This horizon dictates the optimization of foot placement for the subsequent three steps, ensuring that the robot successfully traverses an obstacle obstructing its path. Initialization of the MPC occurs once the robot is close enough for a feasible solution from the optimization process, signaling the attainability of stepping over the obstacle in three steps. The predicted states are depicted in yellow in the figure and include predicted positions x_1^{p+} , x_2^{p+} , and x_3^{p+} as well as predicted velocities v_2^{p-} , v_3^{p-} , and v_4^{p-} . The controllable states x_1^- , x_2^- ,

and x_3^- are chosen through the MPC to optimize the cost function and enforce dynamic equality constraints and state inequality constraints. The step lengths for the three steps are

$$w_1 = x_0^+ - x_1^- \quad (6.5)$$

$$w_2 = x_1^+ - x_2^- \quad (6.6)$$

$$w_3 = x_2^+ - x_3^- \quad (6.7)$$

The control flow for a single step is shown in Figure 82. During stable walking, the stepping controller dictates the step width w_k which is fed into the low-level controller to send torque commands to the robot to achieve the desired step width at the end of the step. A camera or other form of distance detection is used to measure the distance between the robot and the obstacle edges p_1 and p_2 . Once the robot is close enough to the obstacle to yield a feasible MPC optimization, the MPC bypasses the stepping controller and overrides the foot placement commands from the stepping controller for the following three steps. The MPC outputs the foot placement for the next three steps with a single optimization pass. After the three steps are completed, the regular stepping controller is reactivated. Approaching another obstacle will reactivate the MPC and the cycle repeats itself.

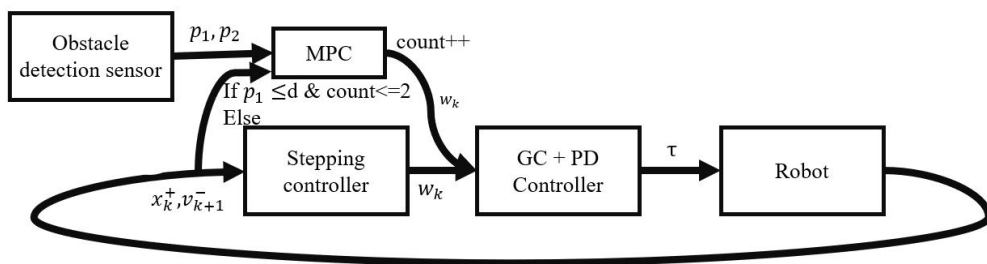


Figure 82: The MPC algorithm flow chart

Early initiation of the MPC is cautioned against, as it may yield infeasible solutions due to the obstacle being too far to meet the imposed constraints.

The optimization problem can be expressed as

$$\underset{\mathbf{v}^-, \mathbf{w}}{\text{minimize}} \quad \left(\sum_{i=k}^{i=3} e_i (\tilde{v}_{i+1}^- - v_{i+1}^-)^2 + \sum_{i=k}^{i=3} f_i (\tilde{w} - w_i)^2 \right) \quad (6.8)$$

$$\text{subject to: } \mathbf{x}_i^+ = C_T \mathbf{x}_i^- + T_c S_T \mathbf{v}_i^- \quad (6.9)$$

$$\mathbf{v}_{i+1}^- = S_T / T_c \mathbf{x}_i^- + C_T \mathbf{v}_i^- \quad (6.10)$$

$$\sum_{i=k}^{i=2} w_i < p_1 \quad (6.11)$$

$$\sum_{i=k}^{i=3} w_i > p_2 \quad (6.12)$$

$$\mathbf{LB} \leq \mathbf{v}^-, \mathbf{x}^-, \mathbf{x}^+ \leq \mathbf{UB} \quad (6.13)$$

where $\mathbf{v}^- = [v_2^-, v_3^-, v_4^-]^T$, $\mathbf{x}^- = [x_1^-, x_2^-, x_3^-]^T$, $\mathbf{x}^+ = [x_1^+, x_2^+, x_3^+]^T$, and $\mathbf{w} = [w_1, w_2, w_3]^T$ are the optimization parameters at the MPC initialization when $k = 1$. The step lengths at every step are given by $\mathbf{w} = [w_1, w_2, w_3]^T$ where $w_i = \mathbf{x}_i^+ - \mathbf{x}_{i+1}^-$. Eq. (Equation 6.9) and (Equation 6.10) are the discretized analytical step-to-step dynamics map from the LIPM. Eq. (Equation 6.11) is the inequality constraint enforcing the foothold location of the second to last step to be before the proximal edge of the obstacle. Eq. (Equation 6.12) is the inequality constraint enforcing the foothold location of the last step to be after the distal edge of the obstacle.

The lower (LB) and upper bounds (UB) must be properly selected to reduce the chance of failure of the MPC framework. To reduce the likelihood of foot slippage, the boundaries of \mathbf{x}^- and \mathbf{x}^+ are

$$-h \cdot \sin(0.7227) < (\mathbf{x}^-, \mathbf{x}^+) < h \cdot \sin(0.7227) \quad (6.14)$$

where h is the COM height of the robot and the angle 0.7227 ensures the normal force remains within 75% of the normal force experienced when \mathbf{x}^- or \mathbf{x}^+ is zero (e.g. the COM is directly over the stance foot).

The NLopt C library was used to carry out the optimization computations [52]. The algorithm used was the gradient-based sequential quadratic programming (SLSQP), commonly used for nonlinearly constrained gradient-based optimization supporting inequality and equality constraints. A maximum iteration limit of 100 was set with a tolerance of 1E-6.

The MPC framework was applied using the LIPM equality constraints to solve the foot placement of three steps to traverse over an obstacle without stepping on it. Figure 83 illustrates the three predicted states in yellow as well as the optimal foot placement obtained from the MPC optimization. The optimization outputs the size of the three consecutive steps such that the states $x_1^+, x_2^+, x_3^+, v_2^-, v_3^-, v_4^-$ are feasible. Figure 84 shows a simulation where the MPC framework was used to step over four obstacles.

The top row of Figure 85 shows the frontal (x) velocity of the COM of the robot during the simulated trial. The pink circle outlines the time when the robot steps over an obstacle.

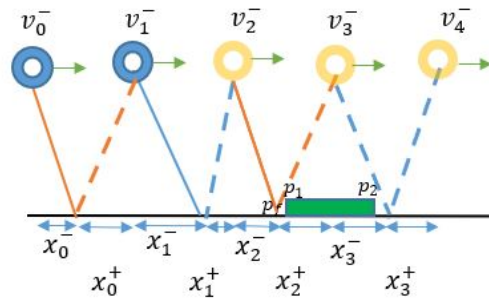


Figure 83: Single-Stage Multi-Step MPC illustration of predictive planning for three steps.

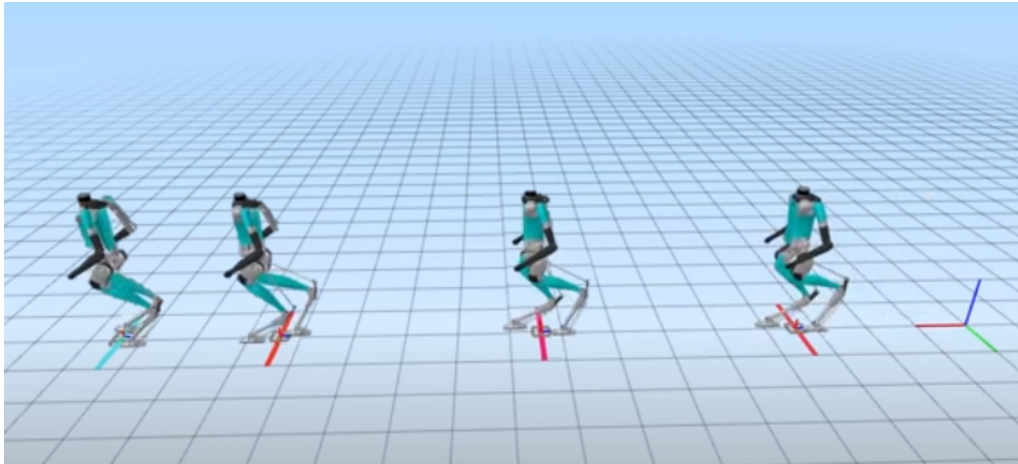


Figure 84: Single-Stage Multi-Step MPC framework applied to walking over obstacles in simulation.

The nominal walking velocity of the robot during the trial was 0.1 m/s. To successfully clear an obstacle, the COM velocity must increase to allow the robot to translate a longer distance and achieve a longer step size that simultaneously clears the obstacle and slows down the COM. The obstacles are 5 cm wide and 2 cm tall bars placed over the floor. A buffer zone of 17.5 cm away from each edge of the obstacle was added to account for the length of the foot during foot placement. The total obstacle length is therefore 40 cm long (5 cm bar length + 2x 17.5 cm buffers).

The fourth row of Figure 85 shows the step length w for the three consecutive steps where step lengths w_1 and w_2 are before the obstacle and the third step length w_3 steps over the obstacle. The second row depicts the predicted states, $x_1^{p+}, x_2^{p+}, x_3^{p+}$, from the MPC optimization. The third row depicts the predicted states, $v_1^{p+}, v_2^{p+}, v_3^{p+}$, from the MPC optimization.

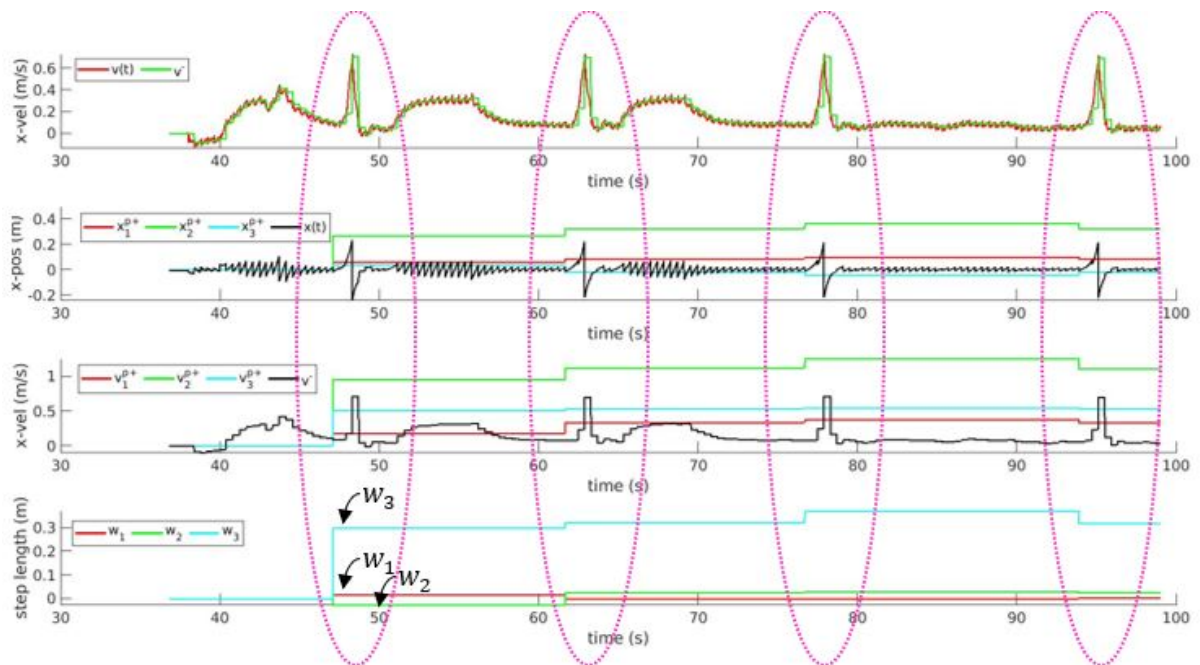


Figure 85: LIPM-MPC results showing, from top to bottom, the COM velocity, the predicted x_k^+ states, the predicted v_k^+ states, and the control inputs (step widths).

Figure 86 shows a closer look at the MPC predicted states for the second obstacle. The red, green, and cyan lines show the predictions from the MPC. The top figure shows the prediction of the states x_k^+ . The black line depicts the position of the COM, $x(t)$. The red, green, and blue dots show the actual states of the robot. The vertical distance between a dot and its respective

colored line represents the prediction error. In the figure, the predicted states x_1^{p+} and x_3^{p+} are very close to the actual states, however, the predicted state x_2^{p+} is not as close. Similarly, the bottom plot shows the prediction of the states v_k^+ . The black line depicts the actual v_k^- . The red, green, and blue dots show the actual states of the robot. The vertical distance between a dot and its respective colored line represents the prediction error.

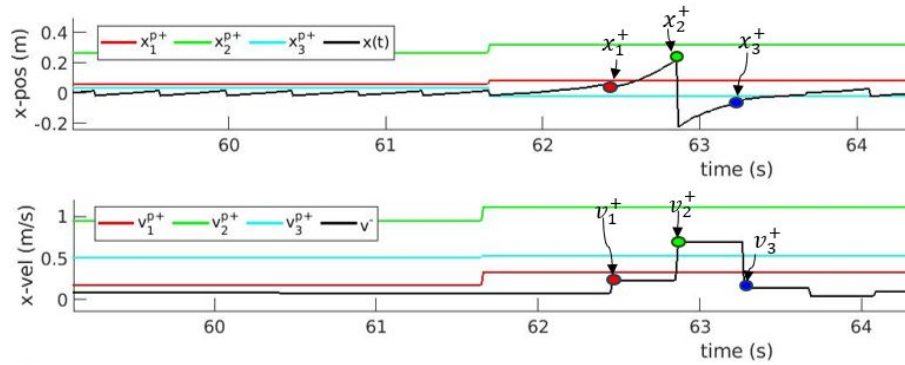


Figure 86: LIPM-MPC results showing, from top to bottom, the predicted x_k^+ states, the predicted v_k^+ states.

6.5.1 MPC Error Propagation

Due to inaccuracies in the LIPM predictions, there will be errors that propagate through the future states of the robot. Assuming the coefficients of the LIPM are a_{11} , a_{12} , a_{21} , and a_{22} the

states x_1^+ and v_1^+ will deviate from the LIPM prediction by error terms ε_{11} and ε_{12} respectively as shown below.

$$x_1^+ = a_{11}x_1^- + a_{12}v_1^- + \varepsilon_{11} \quad (6.15)$$

$$v_1^+ = a_{21}x_1^- + a_{22}v_1^- + \varepsilon_{12} \quad (6.16)$$

The second states, x_2^+ and v_2^+ will then accumulate the previous error from the velocity approximation, ε_{12} , and additional error terms, ε_{21} and ε_{22} will be collected as shown below.

$$x_2^+ = a_{11}x_2^- + a_{12} \left(v_1^{p+} + \varepsilon_{12} \right) + \varepsilon_{21} \quad (6.17)$$

$$v_2^+ = a_{21}x_2^- + a_{22} \left(v_1^{p+} + \varepsilon_{12} \right) + \varepsilon_{22} \quad (6.18)$$

Finally, the third and final states in the MPC, x_3^+ and v_3^+ will further accumulate the previous error from the velocity approximation, and additional error terms ε_{31} and ε_{32} will be collected as shown below.

$$x_3^+ = a_{11}x_3^- + a_{12} \left(v_2^{p+} + a_{11}\varepsilon_{12} + \varepsilon_{21} \right) + \varepsilon_{31} \quad (6.19)$$

$$v_3^+ = a_{21}x_3^- + a_{22} \left(v_2^{p+} + a_{22}\varepsilon_{12} + \varepsilon_{21} \right) + \varepsilon_{32} \quad (6.20)$$

After the three steps, the actual states of the robot can be represented by the predicted states x_3^{p+} and v_3^{p+} plus the accumulated errors as shown below.

$$x_3^+ = x_3^{p+} + a_{12} (a_{11}\varepsilon_{12} + \varepsilon_{21}) + \varepsilon_{31} \quad (6.21)$$

$$v_3^+ = v_3^{p+} + a_{22} (a_{22}\varepsilon_{12} + \varepsilon_{21}) + \varepsilon_{32} \quad (6.22)$$

It is therefore more critical to have a precise model for MPC than for model-based stepping control as the errors will propagate and expand with each predicted state.

6.5.2 NH-LIPM MPC

To reduce the model inaccuracies and error propagation of the MPC framework, an MPC was also developed using the NH-LIP model. This analytical model was extracted from walking simulations to better capture the dynamics of the robot. Figure 87 shows the results from one simulation trial. Figure 88 shows a closer look at the actual and predicted states for stepping over the third obstacle.

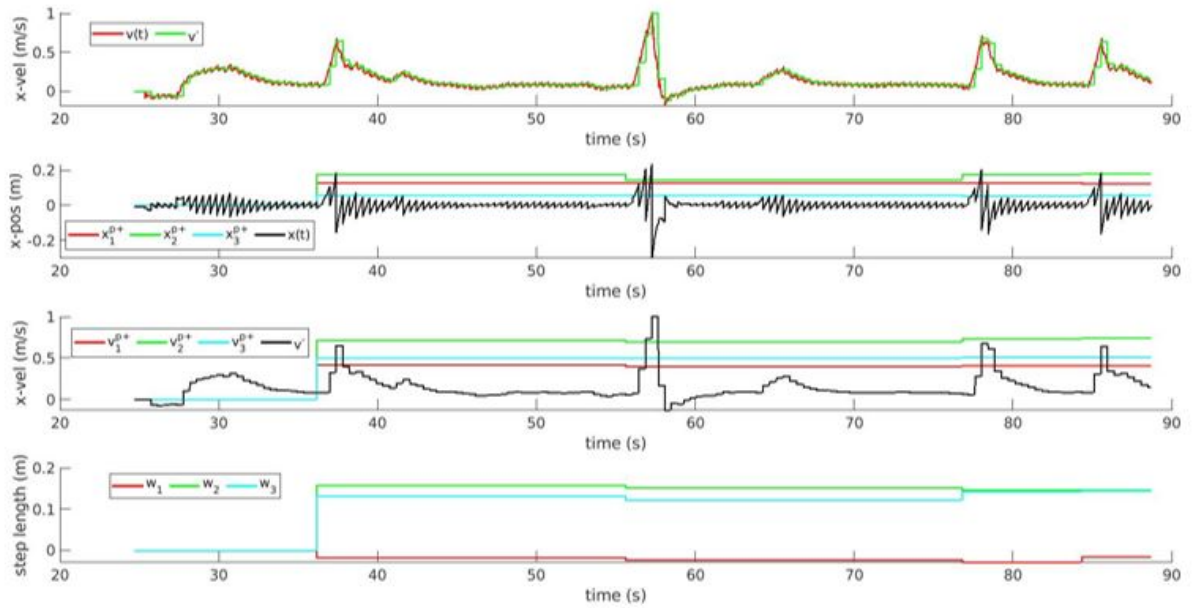


Figure 87: NH-LIPM-MPC results showing, from top to bottom, the COM velocity, the predicted x_k^+ states, the predicted v_k^+ states, and the control inputs (step widths).

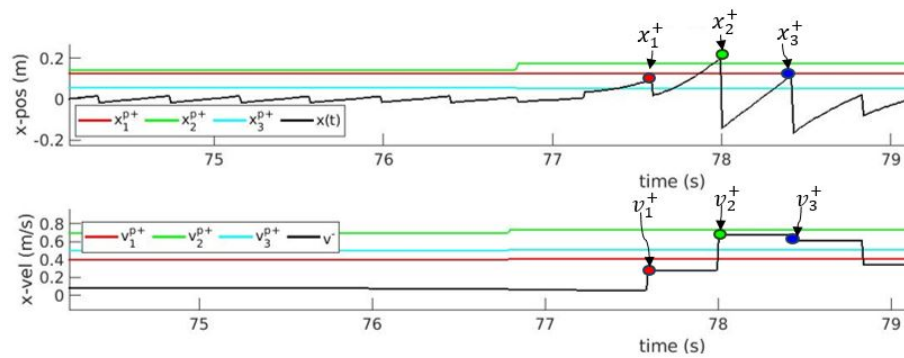


Figure 88: NH-LIPM-MPC results showing, from top to bottom, the predicted x_k^+ states, the predicted v_k^+ states for the third obstacle.

6.5.3 MPC Model Comparisons

The error propagation was evaluated for the LIPM MPC and the NH-LIPM MPC. Table II shows the average prediction errors for each state predicted by the MPC. On average, the NH-LIPM did a better job at state prediction than the LIPM for both v^- and x^+ . The percent improvement using the NH-LIPM is shown on the bottom row of the table.

	v_2^- (m/s)	v_3^- (m/s)	v_4^- (m/s)	x_1^+ (m)	x_2^+ (m)	x_3^+ (m)
LIPM	-0.1	-0.2265	-0.2954	-0.0129	-0.0401	-0.1192
NH-LIPM	0.0721	-0.1521	-0.1849	-0.0056	-0.0184	-0.0673
%Imprv.	27%	28%	27%	53%	54%	43%

TABLE II: Model prediction MAE comparison of MPC showing improvement using the NH-LIPM MPC over the LIPM MPC.

6.5.4 Multi-Stage Receding Horizon MPC

To reduce the error propagating from the model discrepancies, a Multi-Stage Receding Horizon Control (RHC) MPC framework is proposed. Figure 89 shows a summary of the framework. The first stage of the MPC is exactly like the Single-Stage MPC. The control inputs w_1 , w_2 , and w_3 are obtained from the optimization problem, and the following three steps are predicted. The second stage begins after the robot takes its first optimal MPC step and the prediction horizon reduces from three steps to two steps. The second MPC optimization predicts the states x_1^{p+} ,

x_2^{p+} , v_2^{p-} , and v_3^{p-} and solves for the optimal control inputs w_1 and w_2 . The third and final stage follows and the prediction horizon further reduces to one step. The third MPC optimization predicts the states x_1^{p+} , and v_2^{p-} and solves for the optimal control input w_1 .

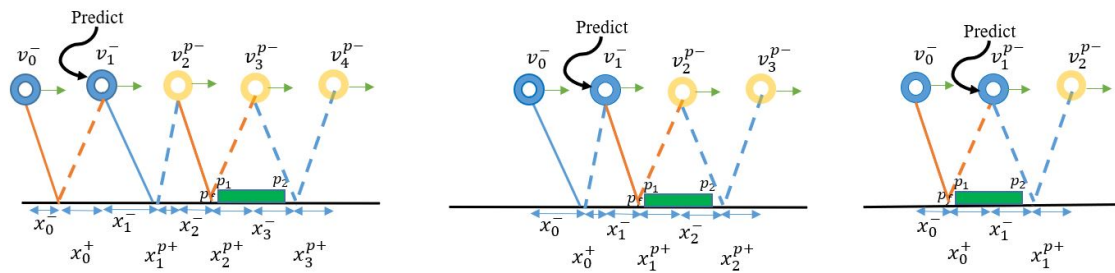


Figure 89: Multi-stage receding horizon MPC used to step over an obstacle in three steps.

A NH-LIPM multi-stage RHC MPC was used to plan the motion of the robot walking over four consecutive floor obstacles in simulation. Figure 90 shows the robot walking at a nominal step length followed by the states when it steps over each of the four floor obstacles.

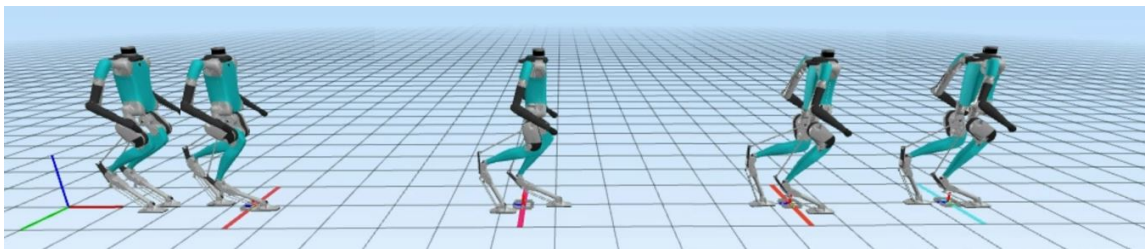


Figure 90: A NH-LIPM multi-stage RHC MPC was used to plan the motion of the robot walking over four consecutive floor obstacles.

Figure 91 shows a closer look at the MPC stepping control to clear the second obstacle. In stage (a) the robot walks with a cyclic gait towards the obstacle at a nominal velocity of 0.1 m/s. At the start of stage (b), the MPC is triggered and solves for the optimal control inputs w_1 , w_2 and w_3 . At stage (c) the control input w_1 is employed and the robot takes a very short step. At the same time, the second stage of the MPC solves for the optimal control inputs of the following two steps, w_1 and w_2 . At stage (d), the control input w_1 is employed and the third stage of the MPC solves for the optimal control input of the following step, w_1 . At stage (e) the final MPC control input, w_1 , is used to step over the obstacle safely. At this stage, the stepping controller is reactivated and dictates the footstep location for the following step as shown in (f). In stage (g) the robot begins to transition back to the symmetric gait.

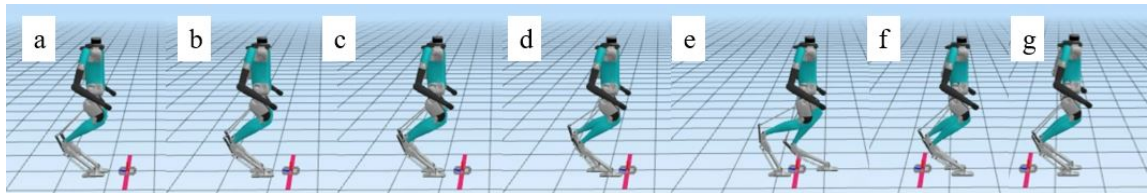


Figure 91: A closer look at the robot states leading up to the second obstacle where the MPC stages occur at (b),(c), and (d).

Figure 92 shows a closeup look at the states $x(t)$ and v^- of the robot and the prediction states from the multi-stage RHC MPC. The red green and blue dots mark the prediction and actual states of the first, second, and third steps of the horizon respectively. As expected, the red dots match up very close to each other along the y-axis indicating that the predictions x_1^{p+}

and v_1^{p+} of the first stage are very close to the actual states x_1^+ and v_1^+ . The green dots and blue dots are less lined up with each other. This is expected as the control inputs for the second and third steps from the first stage MPC are not enforced. The MPC solves for the optimal step size w_1 . The value of the predicted state x_1^{p+} is greater than the value of the states displayed during cyclic walking preceding this step. This indicates that with the optimal step size, the COM is predicted to fall further in front of the stance foot. The actual state x_1^+ , confirms the prediction. Similarly the predicted state v_1^{p+} is greater than the nominal velocity displayed during cyclic walking preceding this step. This indicates that with the optimal step size, the COM is predicted to accelerate. The actual state v_1^+ , confirms the prediction.

In the second stage of the MPC, the predicted states are marked with a red and green circle for the following two steps respectively. The value of the predicted state x_1^{p+} is greater than that of the previous prediction. This indicates that the optimal step size during the second stage will cause the COM to move further in front of the stance foot. The actual state x_2^+ confirms this prediction although there is some error displayed. This indicates that even, the improved model fails to capture the full dynamics of the robot. Similarly, the predicted state v_2^{p+} predicts that at the optimal step size, the velocity will further accelerate. The actual state v_2^+ confirms the prediction but there is also a small deviation in the prediction.

In the third and final stage of the MPC, the predicted states are marked with a red diamond. The actual states x_3^+ and v_3^+ closely resemble the predicted states x_1^{p+} and v_1^{p+} in this stage.

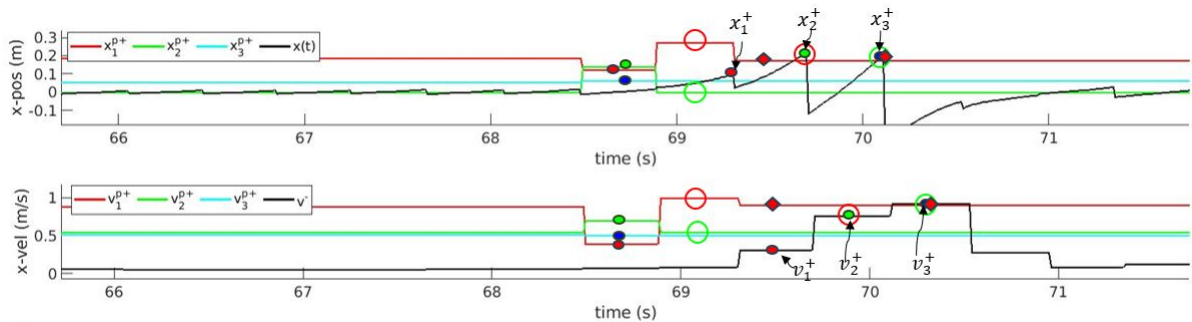


Figure 92: Multi-stage RHC MPC state estimates.

6.5.5 Single-stage MPC vs. Multi-stage MPC Comparison

To test the accuracy of the models, ten simulation experiments were conducted with each MPC framework. Each experiment consisted of using MPC to walk over four obstacles as seen in Figure 90. Table III shows a comparison between predictions of the single-stage MPC and the multi-stage MPC frameworks. Overall, the multi-stage MPC did better at predicting the states v_3^- , v_4^- , x_2^+ , and x_3^+ and marginally worse at predicting v_2^- and x_1^+ than the single-stage MPC. A big benefit of the multi-stage MPC is that the final state prediction is substantially more precise than the single-stage MPC. This is critical, as the last step may determine whether the robot trips over the obstacle or not.

	v_2^- (m/s)	v_3^- (m/s)	v_4^- (m/s)	x_1^+ (m)	x_2^+ (m)	x_3^+ (m)
Single-stage	0.0721	-0.1521	-0.1894	-0.0156	-0.0184	-0.0673
Multi-stage	-0.0778	-0.1447	0.0538	-0.018	-0.0152	0.0376
%Imprv.	-8%	5%	72%	-15%	17%	44%

TABLE III: Model prediction average error comparison of MPC showing improvement using the multi-stage RHC MPC over the single-stage MPC in predicting state v_3^- , v_4^- , x_2^+ and x_3^+ .

6.5.6 Analytical Discrete Finite-Horizon Control Algorithm

Although the single-stage and multi-stage MPCs discussed in the previous section have proven to work for motion planning in an environment with obstacles, one major limitation is the amount of manual tuning that must be done on the optimization boundaries and constraints. Furthermore, the single-stage MPC framework propagates prediction errors that grow with each step and might be further amplified at faster walking speeds. To address these limitations an analytical discrete finite-horizon control framework is proposed. This framework eliminates the initial two-stage MPCs of the multi-stage MPC and replaces them with analytical controllers derived from the optimal feasible space of the dynamics. The analytical controllers drive the robot's dynamics to a feasible start state where a one-step horizon MPC can achieve a feasible and optimal footstep to safely step over an obstacle.

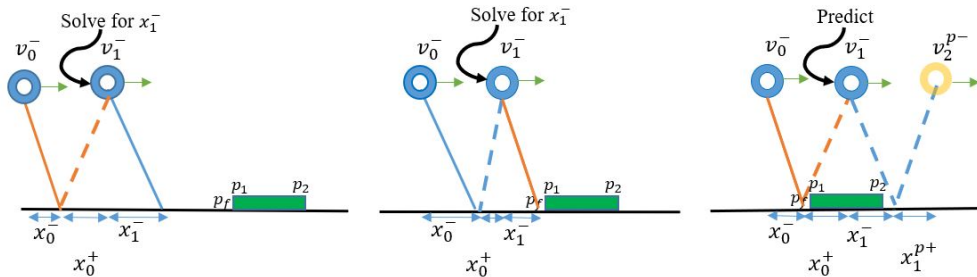


Figure 93: Analytical Discrete Finite-Horizon Control to step over an obstacle.

The first objective was to find the optimal distance d between the stance foot and the proximal edge of the obstacle prior to stepping over the obstacle. To achieve this, a numerical

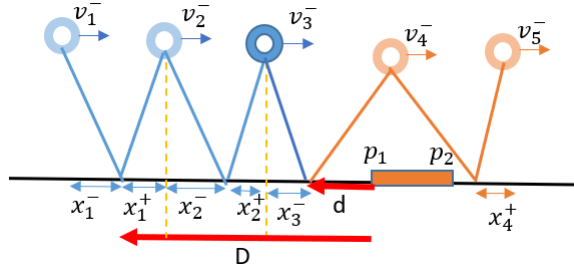


Figure 94: Finite-horizon control parameters in an environment with an obstacle.

simulation was conducted using the optimization problem from Equation 6.8 to determine the feasible space as a function of initial states and obstacle distance. The training set included a range of initial state x_3^- between $-0.15m$ and $0.05m$ and initial state v_3^- between $0m/s$ and $0.7m/s$. The distance d was varied between $0m$, where the foot is at the proximal boundary of the obstacle, to a distance of $0.25m$. Because we want to optimize the step taken to step over the obstacle, we can eliminate the constraint from Equation 6.9 as the step number is $k = 3$. Subsequently, a linear function was fitted to the data using a least squares approach, resulting in an analytical equation that describes the center of the feasible space as depicted in Figure 95. The plane equation is formulated as follows

$$U_3 = f(U_1, U_2) \quad (6.23)$$

where U_k are the states (x_k^-, v_k^-) . The system of equations becomes

$$x_3^- = e_1 + e_2 v_3^- + e_3 d \quad (6.24)$$

where e_1 , e_2 , and e_3 are coefficients. A plane equation with coefficients $e_1 = 0.1119$, $e_2 = -0.2896$, and $e_3 = 0.1297$ was formulated giving the analytical equation of the center of the

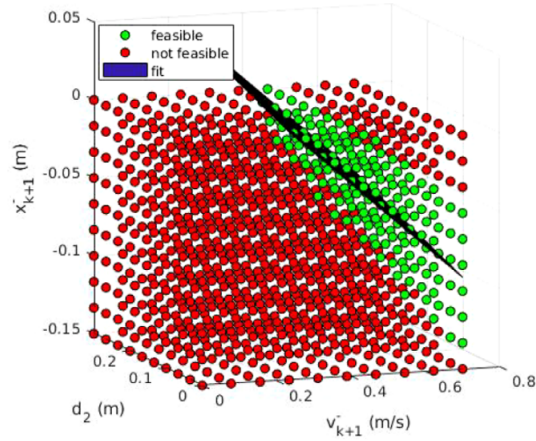


Figure 95: Feasible space of the optimization function and the analytical center plane.

feasible space. This approach allows for a comprehensive understanding of the relationships between the initial states, obstacle distance, and the corresponding feasible region.

The analytical equation describing the center of the feasible space can be used to drive the states to feasibility. This can be done using discrete finite-horizon control of the form

$$U_3 = f(U_1, U_2) \quad (6.25)$$

where U_k are the states (x_k^-, v_k^-) . The system of equations becomes

$$x_2^+ = g_1(x_2^-, v_2^-) \quad (6.26)$$

$$v_2^+ = g_2(x_2^-, v_2^-) \quad (6.27)$$

$$x_1^+ - x_2^- + x_2^+ - x_3^- = D - d \quad (6.28)$$

$$x_3^- = e_1 + e_2 v_3^- + e_3 d \quad (6.29)$$

where g_1 and g_2 are the analytical maps of the step-to-step dynamics. D is the distance at which the finite-horizon controller begins. To find this distance, the feasible space of the three-step horizon optimization problem from Equation 6.8 with initial states x_1^- and v_1^- is used to fit a plane equation through the center of the feasible space reducing the squared distance between the feasible points and the bounds as shown in Figure 96. The analytical optimal distance D is now a function of x_1^- and v_1^- and coefficients $b_1 = 0.23416$, $b_2 = 0.97559$, and $b_3 = 3.4902$.

$$\hat{D} = b_1 + b_2 v_1^- + b_3 x_1^- \quad (6.30)$$

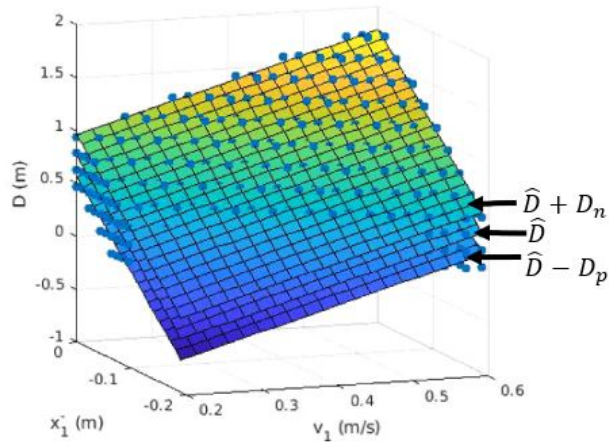


Figure 96: A plane fitted to the center of the feasible space of the obstacle distance parameter D and initial states x_1^- and v_1^- .

Next, the feasible space of D was used to heuristically choose the distance d as a function of the optimal distance \hat{D} and the actual distance D away from the obstacle. d was chosen for a

range of values of D such that the parameters x_2^- , x_2^+ , x_3^- and v_3^- are favorable (close to those displayed in cyclic stepping). A line was then fitted in the form of

$$d = c_1 + c_2(D - \hat{D}) \quad (6.31)$$

where c are the data selection regression coefficients $c_1 = 0.12667$ and $c_2 = 0.646670$.

Because it is difficult to ensure that the distance $D = \hat{D}$, e.g. when walking at a faster gait with longer steps, the buffers D_n and D_p are added as functions of the walking velocity. A faster gait will therefore have a larger buffer than a slower gait. These buffers are shown in Figure 96 creating feasible volume with the planes $\hat{D} + D_N$ and $\hat{D} - D_p$ as the boundaries. The size of the buffers is selected and tuned through simulation trials. A finite state horizon control is achievable as long as the robot steps between the points $\hat{D} - D_p$ and $\hat{D} + D_n$. The analytical discrete finite-horizon controller will dictate the proper state inputs x_2^- and x_3^- needed to arrive at a distance d with feasible input states to the single-stage MPC used to safely step over the obstacle. The system of equations from Equation 6.26- Equation 6.29 along with Equation 6.30 and Equation 6.31 give six equations necessary to solve for the six unknowns: x_2^- , x_2^+ , v_2^+ , \hat{D} , d , and x_3^- in the first step of the finite-horizon controller. The analytical function used for the first step can be represented as A_1 which is a function of distance D , current states x_1^+ and v_2^- as well as the nominal states x^{*+} and v^* displayed during stable symmetric walking.

$$x_2^- = A_1(D, x_1^+, v_2^-, x^{*+}, v^*) \quad (6.32)$$

In the second step Equation 6.28 reduces to the equation

$$x_2^+ - x_3^- = D - d \quad (6.33)$$

and together with Equation 6.29 can be used to solve for the unknowns d and x_3^- . The analytical function used for the second step can be represented as A_2 which is a function of the distance D and the current states x_{2+} and v_3^-

$$x_3^- = A_2(D, x_{2+}, v_3^-) \quad (6.34)$$

In the third step, a single-stage MPC identical to the final stage of the multi-stage MPC framework is employed to solve for the optimal step size to safely step over the obstacle.

The analytical discrete finite-horizon control algorithm can be used to navigate over an obstacle where D_k and D_{k+1} are the distance from the stance foot to the obstacle at step k , s_c is the step size of cyclic-state walking. The flow chart of the finite-horizon control algorithm is shown in Figure 97.

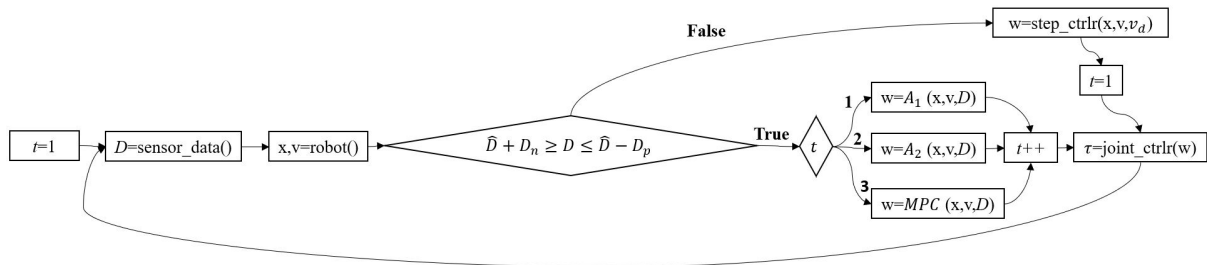


Figure 97: Analytical Discrete Finite-Horizon Control Algorithm Block Diagram

A numerical simulation of the states of the robot as it walks towards an obstacle was formulated using MATLAB. The buffers D_p and D_n were given a value of 0.05 m. Figure 98 depicts the robot's footprint trail during a simulation using the finite-horizon algorithm with varying initial states and distance D to the obstacle shown in blue. For every ten simulations, the velocity increases from 0.1m/s to 0.7m/s. Before the finite-horizon controller A_1 is activated, footsteps are cyclic, indicated by the black footprint. When the controller A_1 is activated, the state x_1 can either be symmetric which happens when the cyclic footprint falls inside the bounds $\hat{D} + D_n$ and $\hat{D} - D_p$ (depicted by the circled black footprint) or it can be asymmetric which happens when a cyclic footprint would violate the constraint $D > \hat{D} - D_p$, in which case the foot is placed at a distance equal to $\hat{D} - D_p$ (depicted by the cyan footprint). The analytical controller A_1 solves for the footprint of the next step (red footprint). The second stage controller A_2 is activated at the next step and it solves for the footprint of the following step (green footprint). The states of the robot at the following step can be used to solve the final step using the MPC single-stage optimization framework.

Due to model discrepancies between the analytical model dynamics and robot dynamics, A bias term β is selected and added to Equation 6.29 to achieve feasibility. The equation becomes

$$x_3^- = a_1 + a_2 v_3^- + a_3 d + \beta \quad (6.35)$$

where β is a velocity-dependent constant that can be manually tuned to achieve a feasible solution. The more precise the model is, the smaller β will be.

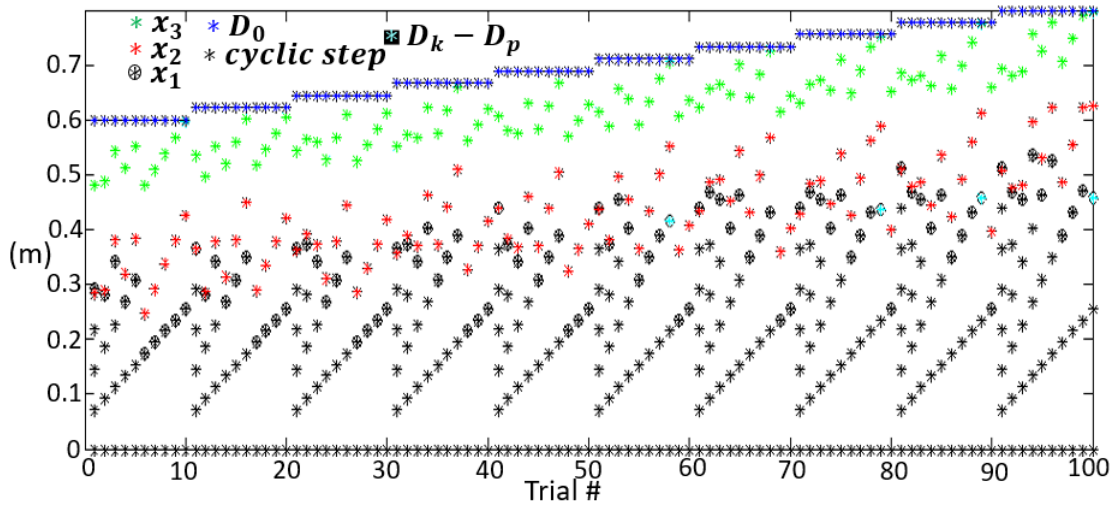


Figure 98: Numerical simulation footprint trail of the analytical finite-horizon control algorithm.

6.5.6.1 Analytical Discrete Finite-Horizon Control Simulation Experiment

The analytical finite-horizon control was tested in simulation at various walking speeds. After several tuning trials, the tunable parameters D_n , D_p , and β were set as shown in Table IV. These values maximized the likelihood of success for every trial.

v_2^- (m/s)	D_n (m)	D_p (m)	β (m)
0.1	-0.1	0.2	0.05
0.2	0.11	0.2	0.01
0.3	0.12	0.2	-0.02
0.4	0.2	0.05	-0.02

TABLE IV: Table of Tunable Coefficients

Once tuning was completed, the analytical discrete finite-horizon controller was used to walk over four floor obstacles at different walking speeds. Figure 99 shows snapshots of the simulations showing a state where the robot walks with a symmetric gait and the state where the robot steps over the obstacle.

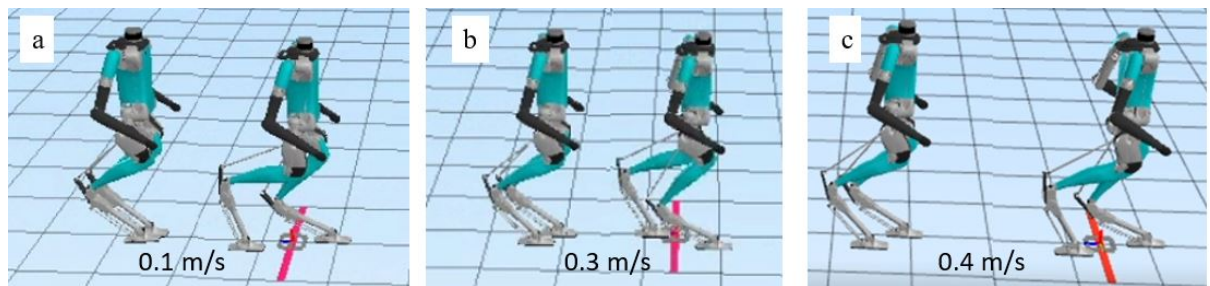


Figure 99: The analytical discrete finite-horizon controller was used to walk over four floor obstacles at different walking speeds.

Figure 100 shows the state data collected during one of the trials where the robot walked with a nominal velocity of 0.5 m/s. The stage plot shows the time segments where the respective controllers are active. The robot begins walking toward the obstacle using the model-based stepping controller. As the robot reaches a distance \hat{D} the first analytical controller, A_1 , overwrites the stepping controller and is used to solve for the state input x_2^- depicted with a blue line in the control plot. For the following step, the controller A_2 is used to solve for the control state x_3^- depicted with an orange line in the control plot. Finally, to step over the obstacle the single-stage MPC optimization is applied to solve for the optimal control state x_4^- depicted with a yellow line in the control plot.

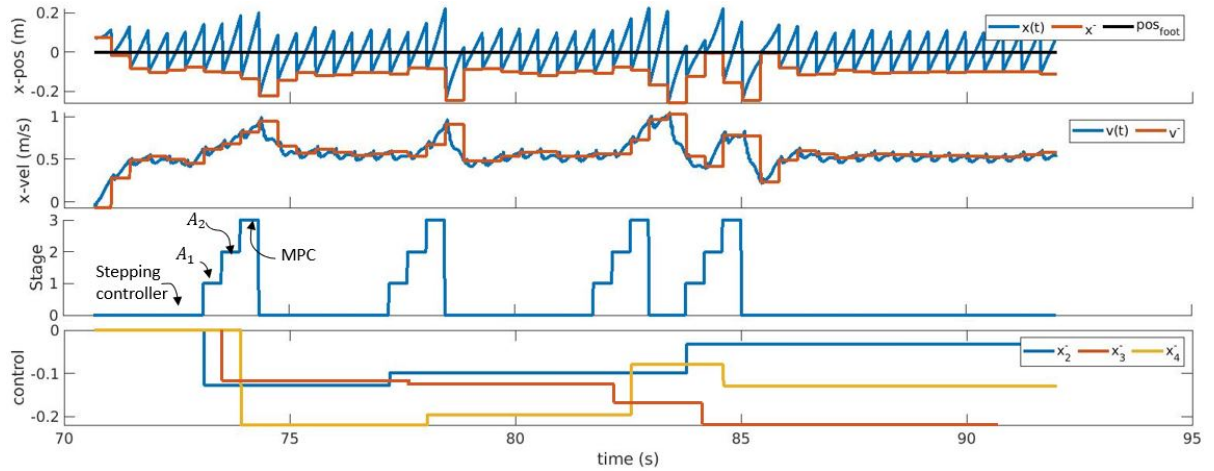


Figure 100: X-position, x-velocity, stage number, and control value plots of the obstacle avoidance simulation using the analytical discrete finite-horizon controller.

The x-pos and x-vel plots show the behavior of the position and velocity of the COM during the trial. To step over the obstacles, the velocity increases from the nominal as indicated by the velocity peaks occurring during stages 1, 2, and 3. The control states x^- tend to increase in magnitude after stage 3 indicating that the MPC controller gives a large optimal step size.

6.5.7 Asymmetric Stepping Controller Comparison

To compare the asymmetric stepping controllers discussed in this chapter, the controllers were used to walk over a total of eight obstacles at a nominal walking speed of 0.2 m/s in simulation. Each obstacle was assigned a success flag if the robot successfully stepped over the obstacle without touching it. A failure was recorded if the feet of the robot touched the obstacle, the robot fell down, or if the optimization yielded an infeasible solution. Table V shows the percentage of success for each of the controllers tested.

	Single Stage MPC	Multi-stage MPC	Analytical Finite Horizon
% success	72.5%	85%	93.8%

TABLE V: Comparison of the successful trials of the three proposed MPC methods where the analytic finite horizon controller had the greatest success.

The analytical finite horizon controller yielded the greatest success as only five failures were recorded. One failure was a fall and the remaining four failures were due to stepping backward after stepping over the obstacle and consequently stepping on the obstacle with the back step.

6.5.8 Conclusion

To summarize, this discussed various methods that are useful for asymmetric stepping. Stable symmetric stepping is useful when walking in straight lines at constant speeds with minimal motion constraints. A feedback controller can stabilize the motion of the robot giving it stability. However, there may be times when walking in straight lines or at constant speeds can limit the task that a robot must complete. For such cases, it is important to study ways in which asymmetric steps can be incorporated during walking to avoid stepping on dangerous regions such as ditches or uneven areas on the ground. The analytical dynamic equation or step-to-step map can be used to predict the behavior of the robot. Knowing the future behavior of the robot lets us plan the proper controls such that the robot safely achieves the states necessary to traverse a constrained environment and regain stability when necessary.

Section 6.3 showed how taking a short step that deviates from the symmetric stepping displayed when using a model-based stepping controller followed by a constrained optimization solver for foot placement enables the robot to step over undesired regions on the floor. Hardware experimentation shows that Digit can reach a frontal ground clearance of nearly 50 cm

when walking at a nominal speed of 0.1 m/s. The dynamics function is incorporated into the optimization as an equality constraint and the cost function ensures that the dynamics of the robot come as close as possible to the desired nominal values.

Section 6.4 explored asymmetric stepping along the lateral direction. More specifically it looked at a strategy to move laterally swiftly. To accomplish this, a wide step is taken in the direction opposite to the direction of the desired movement. This causes the velocity of the COM to increase in the direction of desired movement effectively shifting the COM laterally. The stepping controller then drives the states back to symmetric stability. Sometimes, attempting to move laterally too much can destabilize the robot too much for the stepping controller to stabilize. Subsection 6.4.1 proposes the use of toe torque to add stability to the stepping controller and help stabilize the robot dynamics. Subsection 6.4.2 shows the hardware results of the lateral asymmetric stepping controller.

Finally, section 6.5 proposes three model predictive control frameworks to control the robot when walking over obstacles. Single-stage multi-step MPC uses the modeled dynamics of the robot to predict the state of the robot for three steps and solve for the optimal foot placement in each of the three steps to successfully walk without touching the obstacle. It was shown that the data-driven NH-LIPM did a better job at predicting the behavior of the dynamics during the MPC formulation than the LIPM. Although this approach requires solving only one optimization problem per obstacle, modeling errors tend to propagate throughout the planning horizon. Multi-stage multi-step MPC is then proposed where a new plan is developed at every stage when walking over an obstacle. This effectively reduces the error propagation and yields

overall more feasible results. However, having to optimize three times per obstacle also increases the amount of tuning that has to be done in order to effectively use the framework. Furthermore, it increases the chances that the optimization problem does not converge to a feasible result as a result of inadequate tuning. A final analytical discrete finite-horizon control algorithm was proposed to tackle some of these limitations. This approach looks at the feasible space predicted from the modeled dynamics and designs analytical functions that map the robot states to the feasible space through a control state x^- . This approach proved to yield better results than the single and multi-stage MPCs. The optimization stage of the proposed algorithm only failed to converge to a feasible solution one out of eighty times.

CHAPTER 7

MODEL ADAPTIVE CONTROL USING THE RECURSIVE LEAST SQUARES (RLS) ALGORITHM AND MODEL INTEGRAL FEEDBACK

7.1 Introduction

In the realm of developing stepping controllers for bipedal robots, a prevalent strategy involves employing simplified models that encapsulate vital dynamic characteristics. Within the context of this research project, the Linear Inverted Pendulum Model (LIPM) has been a focal point, alongside its data-driven derivatives such as the Non-Homogeneous LIPM (NH-LIP) and the Forced-LIPM (F-LIPM). These models assume the capacity to effectively capture the positional and velocity behaviors of the robot's center of mass (COM). However, a critical limitation surfaces when these enhanced LIPM variants are solely reliant on data acquired during trained trials, constraining their accuracy to the trained parameter space. Beyond this space, the risk of model failure looms large. Moreover, when the physical attributes of the robot evolve—such as increased weight or shifting COM due to payload transport—the models may become obsolete. To surmount these challenges, this chapter introduces an adaptive model framework. The core premise revolves around real-time modification of model parameters to mitigate the likelihood of failure when a static model is no longer applicable. This chapter unveils two distinct approaches to this end. The first approach leverages a Recursive Least Squares Algorithm to continually update the NH-LIPM parameters, ensuring adaptability. The second approach introduces a

integral feedback controller into the model-based stepping controller. This integral component introduces a constant bias that aligns model dynamics with the evolving robot dynamics.

7.2 Recursive Least Squares (RLS)

The RLS algorithm was utilized in this study to enable real-time adaptation of the discrete S2S model parameters. The nonhomogeneous term (h in Equation 5.6) of the NH-LIPM can be represented as \hat{y} in the linear form.

$$\hat{\mathbf{y}}^T = [\boldsymbol{\theta} \mathbf{x}^T] \quad (7.1)$$

$$\mathbf{y} = [p_{k+1}, v_{k+1}] \quad (7.2)$$

$$\mathbf{x} = [p_k, v_k] \quad (7.3)$$

$$\boldsymbol{\theta} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \quad (7.4)$$

y contains the predicted center of mass position and velocity error states $[p_{k+1}, v_{k+1}]$, x are the current error states, and θ are the model parameters. The RLS LIPM would then take the discrete form of $\boldsymbol{\eta}_k = \zeta_k(\mathbf{s}^-) + \hat{\mathbf{y}}^T$.

An overview of the algorithm can be found in section 5.4.5. To evaluate the performance of the RLS adaptive controller, the model parameters of the LIPM were tampered with to increase the modeling errors of the nominal model. In another trial, the damping of the stance toe actuators was increased to add unmodeled behavior to the robot dynamics.

The LIPM is shown in Equation 7.5. Its derivation is explained in section 5.3.2.1. The LIPM can be enhanced by adding the RLS parameters into the matrix of coefficients as shown in Equation 7.6. The control state x_k^{L-} and the velocity state v_k^- can then be solved by inverting the matrix of coefficients and multiplying it by the desired states v_k^+ and x_k^+ as shown in Equation 7.7. However, the state v_k^- is a known state. Therefore, a decision has to be made on whether to control x_k^+ or v_k^+ . For this study, the state v_k^+ is controlled for velocity tracking. The stepping controller used for the RLS experimentation is shown in Equation 7.8. This is a foot-strike corrected controller with feedback as described in section 4.4.3.

$$\zeta(\mathbf{s}_k^-) = \begin{bmatrix} x_k^{L+} \\ v_k^{L+} \end{bmatrix} = \begin{bmatrix} C_T & T_c S_T \\ S_T/T_c & C_T \end{bmatrix} \cdot \begin{bmatrix} x_k^- \\ v_k^- \end{bmatrix} \quad (7.5)$$

$$\tilde{\zeta}_R(\mathbf{s}_k^-) = \begin{bmatrix} x_k^{L+} \\ v_k^{L+} \end{bmatrix} = \begin{bmatrix} C_T + a_{11} & T_c S_T + a_{12} \\ S_T/T_c + a_{21} & C_T + a_{22} \end{bmatrix} \cdot \begin{bmatrix} x_k^- \\ v_k^- \end{bmatrix} \quad (7.6)$$

$$\mathbf{s}_k^- = \begin{bmatrix} x_k^{L-} \\ v_k^{L-} \end{bmatrix} = \begin{bmatrix} C_T + a_{11} & T_c S_T + a_{12} \\ S_T/T_c + a_{21} & C_T + a_{22} \end{bmatrix}^{-1} \cdot \mathbf{s}_k^{L+} \quad (7.7)$$

$$u_k^{rls} = x^R(t) - x_k^{L-} + \mathbf{k}_s(\mathbf{s}(t)^{\mathbf{R}} - \mathbf{s}_k^{L+}) \quad (7.8)$$

To assess the RLS controller's adaptive tuning capability for model parameters, we conducted a velocity-tracking trial and compared its performance to that of the nominal model (a corrupted

version of the LIPM used to increase modeling errors). The tracking and model errors were evaluated according to Equation 7.9 and Equation 7.10.

$$E_{track} = v^{R-} - v_{des}^- \quad (7.9)$$

$$\mathbf{E}_{model} = \mathbf{s}^{R+} - \mathbf{s}^{L+} \quad (7.10)$$

7.3 Model Integral Feedback Control (MIF)

Another useful way to enhance the adaptive model is by adding an adaptive bias term to the model of the form

$$\tilde{\zeta}_F(\mathbf{s}_k^-) = \zeta(\mathbf{s}_k^-) + \mathbf{B} \quad (7.11)$$

where the adaptive bias \mathbf{B} corrects for modeling inaccuracies. The adaptation can be done using a summing integral on the model error. The new adaptive model is of the form shown in Equation 7.12 where \mathbf{K}_I is a gain matrix whose eigenvalues are less than one. The gain matrix \mathbf{K}_I is assigned a value of 10% the matrix of coefficients of the LIPM, but can be manually tuned for optimal performance. An integral windup limit of 0.01 is added as a constraint to prevent the integral term from accumulating excessively when the control error remains constant or near a saturation limit. The stepping controller used for the model integral feedback control experimentation is shown in Equation 7.14. The controller block diagram is shown in ?? where

the adaptive model is outlined with dashed lines. This is a discrete model-based controller with feedback as described in section 4.4.2.1.

$$\tilde{\zeta}_F(\mathbf{s}_k^-) = \zeta(\mathbf{s}_k^-) + \mathbf{K}_I \cdot \sum (\mathbf{s}_{k-1}^{\mathbf{R}^+} - \tilde{\zeta}_F(\mathbf{s}_{k-1}^-)) \quad (7.12)$$

$$\mathbf{s}_k^- = \begin{bmatrix} x_k^{L-} \\ v_k^{L-} \end{bmatrix} = \begin{bmatrix} C_T & T_c S_T \\ S_T/T_c & C_T \end{bmatrix}^{-1} \cdot \left[\mathbf{s}_k^{L+} - \mathbf{K}_I \cdot \sum (\mathbf{s}_{k-1}^{\mathbf{R}^+} - \tilde{\zeta}_F(\mathbf{s}_{k-1}^-)) \right] \quad (7.13)$$

$$u_k^f = x^{L+} - x_k^{L-} + \mathbf{k}_s(\mathbf{s}(t)^{\mathbf{R}} - \mathbf{s}_k^{L+}) \quad (7.14)$$

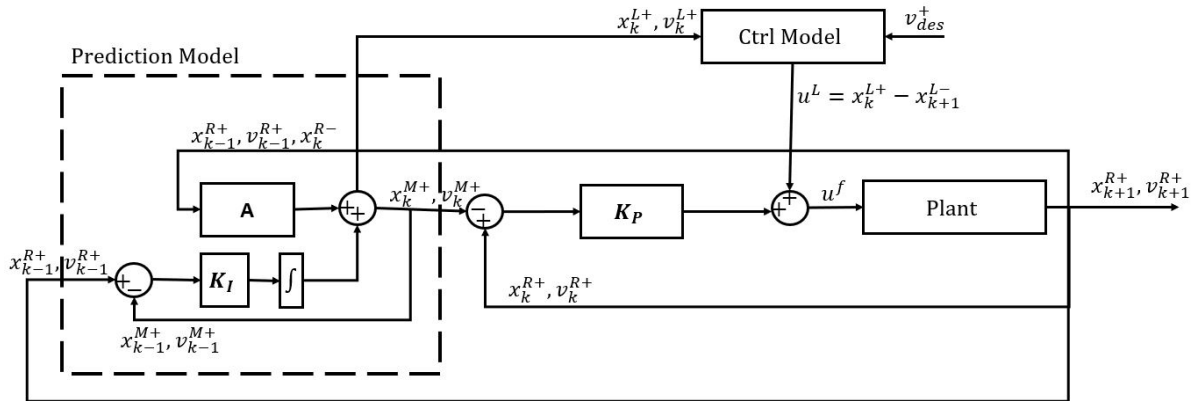


Figure 101: Block diagram of the MIF controller.

To assess the MIF controller's adaptive bias tuning capability, a velocity-tracking trial is conducted and its performance is compared to that of the nominal model (a modified version of the LIPM to increase modeling errors). The tracking and model errors were evaluated according to Equation 7.9 and Equation 7.10.

7.4 Results

A velocity-tracking simulation experiment was designed to follow a reference velocity at 0 m/s, 0.1m/s, 0.2 m/s, 0.4 m/s and 0.6 m/s. The altered LIPM nominal controller displayed a tracking MAE of 0.114 m/s. It displayed a position modeling MAE of 0.045 m and a velocity modeling MAE of 0.07 m/s. The RLS adaptive controller showed great improvements with a tracking MAE of 0.0137 m/s, an improvement of 88% over the nominal model. It displayed a position modeling MAE of 0.015 m and a velocity modeling MAE of 0.0098 m/s, corresponding to improvements of 67% and 87% respectively.

In the second experiment, to further alter the robot dynamics the stance toe damping of the robot was increased to 30% of its maximum value. The robot then walked using the nominal controller at a desired walking velocity of 0.6 m/s before switching to the RLS adaptive controller after six seconds. The robot then walked for another six seconds using the adaptive RLS control law. The results are shown in Figure 102. The velocity tracking error is depicted by the green arrow in the velocity plot showing the large gap between the desired state v_{des}^- and the actual state v^- . The velocity modeling error is depicted by the red arrow in the velocity plot showing the difference between the predicted state of the nominal model v^{L+} and the actual state v^- . The position plot shows the position modeling error depicted by the red arrows pointing at the gap between the model predicted state x^{L+} and the actual state x^+ . The dotted black line indicates the activation of the RLS adaptive controller. At this point, the model parameters begin to correct themselves to reduce the prediction errors from the model. The velocity plot shows the robot state v^- and the model predicted state v^{L+} converge near the desired tracking

velocity indicating that the tracking error and the velocity model error have decreased. Similarly, the position plot shows the robot state x^+ reaches the yellow line that represents the position predicted state x^{L+} . Convergence is achieved in 12 steps and the nominal RLS parameter values converge to -0.11, -0.016, 0.406, and -0.03 for a_{11} , a_{12} , a_{21} and a_{22} respectively as shown in the parameter plot.

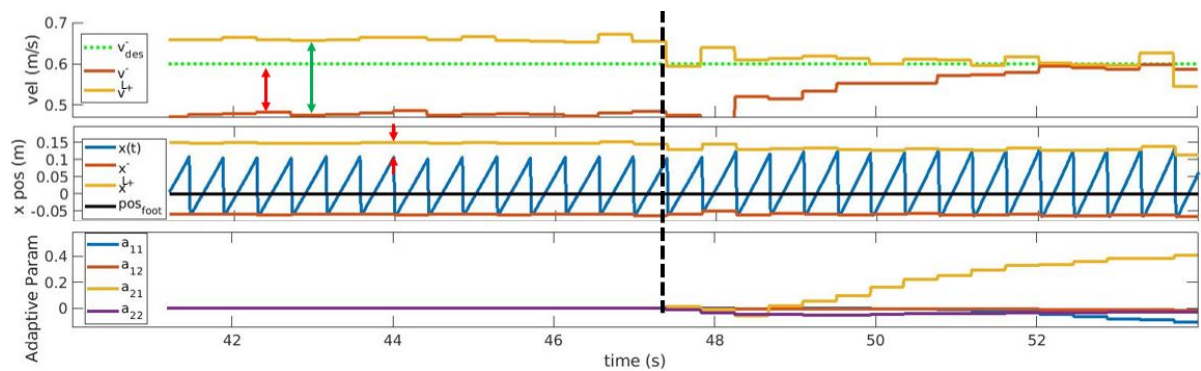


Figure 102: Plots of the x-velocity, x-position, and RLS parameter values of the 0.6 m/s forward walking experimental trial showcasing adaptive RLS ability to reduce modeling and tracking errors by updating the model parameters.

The velocity-tracking simulation experiment was also done on the MIFC controller and compared against the nominal altered LIPM controller. The MIFC adaptive controller showed a tracking MAE of 0.042 m/s, an improvement of 63% over the nominal controller. It displayed a position modeling MAE of 0.006 m and a velocity modeling MAE of 0.0046 m/s, corresponding to improvements of 87% and 93% respectively.

In the second experiment, the altered LIPM controller was used in simulation to walk at a desired walking velocity of 0.6 m/s before switching to the MIFC adaptive controller after six seconds. The robot then walked for another six seconds using the adaptive MIFC control law. The results are shown in Figure 103. The velocity tracking error is depicted by the green arrow in the velocity plot showing the large overshoot of the state v^- . The velocity modeling error is depicted by the red arrow in the velocity plot showing the difference between the predicted state of the nominal model v^{L+} and the actual state v^- . The position plot shows the position modeling error depicted by the red arrows pointing at the gap between the model predicted state x^{L+} and the actual state x^+ . The dotted black line indicates the activation of the MIFC adaptive controller. At this point, the adaptive bias \mathbf{B} begins searching for a value that reduces the model prediction errors. The velocity plot shows the robot state v^- and the model predicted state v^{L+} start to converge shortly after adaptation begins. There is a small undershoot of the robot and model states before converging to a velocity just under the desired indicating that the tracking error and the velocity model error have decreased. Similarly, the position plot shows the robot state x^+ reaches the yellow line that represents the position predicted state x^{L+} . Position convergence is achieved in 12 steps and the velocity stabilizes after 15 steps. The adaptive bias parameter converges to $\mathbf{B} = [0.045, 0.045]^T$ as shown in the parameter plot.

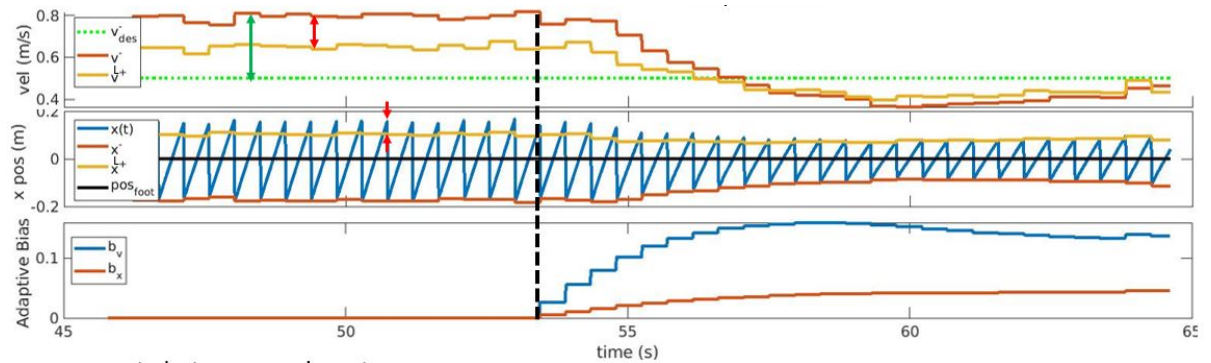


Figure 103: Plots of the x-velocity, x-position, and RLS parameter values of the 0.6 m/s forward walking experimental trial showcasing adaptive RLS ability to reduce modeling and tracking errors by updating the model parameters.

7.5 Conclusion

In this chapter, we introduced two adaptive methods designed to enhance model accuracy, subsequently leading to notable improvements in velocity tracking performance. The Recursive Least Squares (RLS) adaptive controller exhibited its capacity to effectively tune model parameter coefficients, progressively driving modeling errors toward convergence with real-world data. Simultaneously, the Model-Integrated Feedback Control (MIFC) adaptive controller introduced a tunable bias term, providing a flexible means to account for modeling inaccuracies. However, a limitation we encountered was the potential state-dependence of the bias term, raising questions about its compatibility with model predictive control—a direction ripe for further investigation.

The successes of both adaptive controllers in significantly ameliorating the accuracy of our adapted models within a mere twelve steps underscore the promise of adaptive control techniques. Looking ahead, future research endeavors should expand the horizon by exploring alter-

native adaptive methods and their integration. The application of RLS to nonlinear parameter coefficients represents a compelling avenue for further inquiry, unlocking the potential for more robust and versatile model adaptability. Additionally, the examination of model adaptability in dynamic, real-time scenarios is an exciting direction that holds great potential for enhancing control strategies for a wide array of applications.

CHAPTER 8

SOFTWARE FRAMEWORK

8.1 Introduction

This chapter aims to provide a comprehensive overview of the software framework used to design a controller for Digit. It is worth noting that the software framework developed was built entirely from scratch, using only a very basic sample code provided by Agility Robotics as a starting point. Developing the software became a critical task in my dissertation research, as it enabled me to implement and test different control algorithms.

Digit is equipped with an embedded computer for real-time communication between the controller and the robot. The low-level API provides access to critical parameters such as motor torque, damping, and velocity commands, as well as sensor data, including joint encoders and IMU data stored as C-structured data.

To create an accurate model of the robot, the physics engine MuJoCo is used, which allows us to load a model of the robot and update it in real-time using the sensor information. This model provides us with physics parameters such as the robot's center of mass and enables us to perform inverse dynamics calculations easily by giving access to the mass-inertia matrix and gravity and Coriolis vectors.

This chapter will provide a detailed description of the implementation of the software framework, including the communication protocol between the controller and the embedded computer,

the MuJoCo model of the robot, and the inverse dynamics calculations used to generate control signals. This framework will be implemented and tested in simulations and real-world environments. This chapter aims to provide a solid foundation for the future development of controllers for biped robots using similar software frameworks.

8.2 Communication Diagram

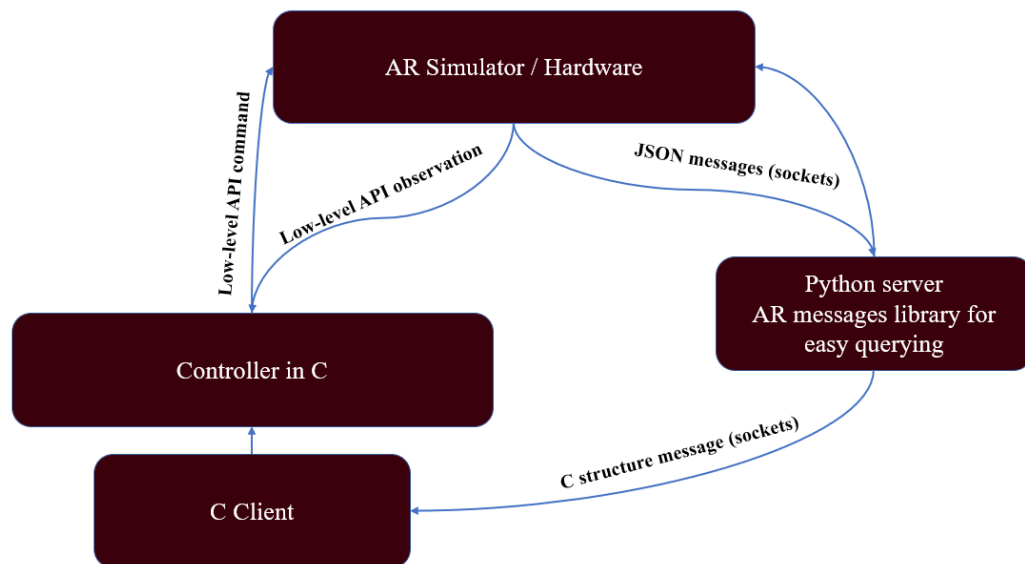


Figure 104: The example figure C

Figure 104 shows the basic communication structure between the robot and the controller. The robot computer sends and receives JSON messages through sockets. Agility Robotics provided a sample Python code in their proprietary SDK with built-in functions to send and receive messages to the robot through websockets. A Python server was created to broadcast information via sockets sent in a C-structured data format to a C client to establish communication

between the Python SDK and the C program. The C client then shares the information received from the Python server with the main controller program. More information about this type of protocol can be found here [53]. The type of information that would be read from the JSON messages and used for the control includes information such as the position of objects in the world environment. This is especially critical for obstacle avoidance and allows us to test walking controllers in a constrained environment.

8.3 Main Program

The C program consists of a main file that initializes all functions and variables necessary for the program to run. This includes multithreading using pthreads. The main file contains a continuous loop that runs several functions in series.

One such function is the "update_states" function, which updates the physics model of the robot using sensor data received from the robot. Another important function is the "estimate_loop" function, which updates important estimates used in the controller, such as the center of mass velocity with respect to the stance foot frame or the linear and angular velocities of the base of the robot with respect to the stance foot frame.

Additionally, the program includes "gravity-compensation" functions that update the gravity compensation torques, which are solved using inverse dynamics on the physics model. The "control_loop" function contains the low-level controller of the robot, which updates the motor torque commands and includes several finite-state machine functions.

To ensure that the motor torque limits are met, the "clamp_command" function clamps torques that exceed the limits. Lastly, the "data_collection" function writes important data to a .csv file.

Overall, this C program involves several complex functions that are critical to the operation of the bipedal robot, including updating the physics model, estimating important parameters, and controlling the motor torque commands.

CHAPTER 9

CONCLUSION

From a simple pendulum to the intricate domain of bipedal robotics, this dissertation has explored the intriguing parallels that underlie both systems. In doing so, we have found the connection between these seemingly disparate entities. The utilization of a robot model such as Digit, with its center of mass (COM) closely mirroring the dynamics of an inverted pendulum during the single stance phase, has afforded us a unique vantage point for dissecting the intricacies of pendulum dynamics within the context of bipedal locomotion. This convergence of worlds has not only broadened our understanding but has also bestowed practical insights into the realm of walking control, where the lessons from pendulum dynamics find surprising resonance.

Chapter 2 is the gateway into pendulum dynamics. Here, we show that the innate dynamics of a pendulum can be augmented through external forces that are akin to the controls we wield in the realm of robotics. In this space, the concept of the Poincare map emerges which paints a discrete map of a system's states and their evolution from one cycle to the next. In the case of a swinging pendulum, the states can be the angle and velocity of the pendulum. In a bipedal system whose behavior loosely mimics the behavior of an inverted pendulum, the Poincare map can be used to map the step-to-step dynamics. Furthermore, if the height of the inverted pendulum is kept constant such as in the case when the legs are telescopic, the equations of

motion of the inverted pendulum become linear and hence the step-to-step dynamic equations are also linear which is where the Linear Inverted Pendulum gets its name.

In Chapter 3, we explored a 5-link planar biped system, deriving its equations of motion using the Euler-Lagrange method. Through numerical simulations in MATLAB, we gathered data from biped walking trials. The focus sharpened on the Poincaré map, a powerful tool to understand the robot's dynamics. Using a wide data set of initial states and control inputs, we employed polynomial regression analysis to construct an analytical model. Support vector machine classification delineated analytically feasible boundaries. The pinnacle of the efforts unveiled the ability to optimize bipedal walking control, enabling the robot to navigate unconstrained environments and terrains featuring ditches.

Chapter 4 introduces the bipedal robot Digit. Digit is a 30-degree-of-freedom bipedal robot with 20 actuated joints. Unlike many humanoid robots, Digit is more lightweight and therefore more agile. Because of torque limitations on the robot, static walking can be challenging on Digit. A more dynamic approach is proposed in the chapter by modeling the robot as an inverted pendulum with a point mass. The LIPM is a good place to start to design stepping controllers for the robot. The MuJoCo physics simulator is used to collect stepping data using the LIPM-based controller. The data is used to then extract an analytical nonlinear function that better maps the step-to-step map of the robot dynamics. Similar to Chapter 3, SVM classification can be used to extract an analytically feasible boundary of the control parameters and initial states of the robot. The analytical models extracted were used to develop different types of model-based controllers. Each controller achieved stable walking, but the model-based foot strike-corrected

controller with feedback was better overall as it was able to achieve the most accurate velocity tracking and could better handle perturbations during walking. The analytical step-to-step map along with the analytical boundary were then used to design a QCQP which was used to safely walk over obstacles.

Chapter 5 introduces the concept of the non-homogeneous LIPM (NH-LIPM). This method of modeling the dynamics of Digit uses the general LIPM plus a non-homogeneous or forcing term that enhances the model. The idea is that even though the robot dynamics deviate from the LIPM dynamics, we can still model it as an LIP with other forcing elements influencing its dynamics behavior. A general NH-LIPM can be extracted from walking training trials in simulation and on hardware. This nonlinear term enhances the velocity-tracking ability of the controller when compared to the LIPM-based controller baseline. Furthermore, this study shows that we can model other control inputs such as ankle damping and ankle torque as a forced-NH-LIPM (F-NH-LIPM). The F-NH-LIPM can then be used for walking control using foot placement and ankle torque as control inputs. The study shows that adding an extra control input gives control to two states simultaneously. Toe torques can also be used to stabilize a controller by driving the robot's continuous dynamics toward the continuous LIPM dynamics. The study also shows how the recursive least squares algorithm can be used to develop an adaptive NH-LIPM capable of model adaptation. Testing is done in simulation and in hardware.

Chapter 6 dives into the asymmetric dynamics of the robot. Walking in an open space free of constraints can be achieved with stable symmetric stepping. This type of control is generally simple and stability can be obtained by choosing proper feedback gains that stabilize

the controller. However, walking in a constrained environment requires more precision than stability. This chapter explores the methods by which studying the dynamics of the robot can guide the development of asymmetric stepping control. Model predictive control (MPC) is proposed as a method to find the optimal foot placement that achieves stepping in a precise manner while staying safe from slips or falls. Frontal and lateral symmetric stepping trials are done on hardware illustrating the ability to take longer or wider steps quickly and safely by imposing constraints and optimizing foot placement. Three MPC frameworks are proposed to handle the task of walking over ground obstacles. The discrete finite-horizon MPC achieved better results as it only requires one optimization per obstacle and is driven by analytical functions that dictate the foot placement as functions of current states for the two steps leading up to an obstacle.

Chapter 7 revisits model adaptive control using RLS adaptation and Model Integral Feedback Control (MIFC) to improve models using online data. Improving the model leads to improvements in the steady state control of the system. Having an improved analytical model also enhances the performance of an MPC where the analytical model is treated as a constraint in the optimization problem. Both approaches showed promising results as they were able to correct the altered models to drive the model error and tracking error toward zero.

Chapter 8 describes the software framework used for control and simulation. The project files can be found in my GitHub account <https://github.com/shinerboy/Dissertation>.

APPENDICES

Appendix A

COPYRIGHT PERMISSIONS

Chapter 3, ASME 2021 International Design Engineering Technical Conferences (ASME
Publisher)

[< Publications & Submissions](#) < [Journals](#) < [Information for Authors](#) < [Journal Guidelines](#) < [Rights and Permissions](#)

Rights and Permissions

ASME Journals Digital Submission Tool Guidelines and Information

Rights and Permissions

Assignment of Copyright

ASME requests that authors/copyright owners assign copyright to ASME in order for a journal paper to be published by ASME. Authors exempt from this request are direct employees of the U.S. Government, whereby papers are not subject to copyright protection in the U.S., or non-U.S. government employees, whose governments hold the copyright to the paper.

For more information on copyright, please view the [Copyright Transfer information page](#).

Retained Rights of Authors

Authors retain all proprietary rights in any idea, process, procedure, or articles of manufacture described in the Paper, including the right to seek patent protection for them. Authors may perform, lecture, teach, conduct related research, display all or part of the Paper, and create derivative works in print or electronic format. Authors may reproduce and distribute the Paper for non-commercial purposes only. Non-commercial applies only to the sale of the paper per se. For all copies of the Paper made by Authors, Authors must acknowledge ASME as original publisher and include the names of all author(s), the publication title, and an appropriate copyright notice that identifies ASME as the copyright holder.

Permissions

Once your paper has been published by ASME, you may wish to submit it for inclusion in a non-ASME publication or to incorporate some or all of its elements in another work. Since ASME is the legal holder of copyright for its papers, it will be necessary for you to secure the permission of the copyright holder to have its material published in another source.

In this case, for permission to have your paper - in whole or in part, as is or adapted - published elsewhere, [please submit your request here](#).

Questions

The ASME Publishing staff is available to discuss current ASME policy on permissions and rights. Please feel free to contact us with any questions or comments at: permissions@asme.org.

Appendix A (Continued)

Chapter 4, Proceedings of the 2022 IEEE-RAS International Conference on Humanoid Robots (IEEE Publisher)



Sign in/Register



Quadratically constrained quadratic programs using approximations of the step-to-step dynamics: application on a 2D model of Digit

Conference Proceedings: 2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)

Author: Ernesto Hernandez Hinojosa

Publisher: IEEE

Date: 28 November 2022

Copyright © 2022, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

- 1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
- 2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
- 3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

- 1) The following IEEE copyright/credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
- 2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
- 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.


If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK

CLOSE WINDOW


Appendix A (Continued)

Chapter 5, Proceedings of the 2023 IEEE-RAS International Conference on Humanoid Robots (IEEE Publisher). Since the paper hasn't been published yet, I am reattaching the thesis reuse guidelines from the conference the year before.



RightsLink

[Sign in/Register](#)
?
🔍



Quadratically constrained quadratic programs using approximations of the step-to-step dynamics: application on a 2D model of Digit

Conference Proceedings: 2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)

Author: Ernesto Hernandez Hinojosa

Publisher: IEEE

Date: 28 November 2022

Copyright © 2022, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

- 1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
- 2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
- 3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

- 1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
- 2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
- 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK
CLOSE WINDOW

© 2023 Copyright - All Rights Reserved | [Copyright Clearance Center, Inc.](#) | [Privacy statement](#) | [Data Security and Privacy](#) | [For California Residents](#) | [Terms and Conditions](#)
 Comments? We would like to hear from you. E-mail us at customercare@copyright.com

CITED LITERATURE

1. Hernandez-Hinojosa, E.: Humanoid stepping. <https://youtu.be/-UL-wkv4XF8>, March 2013 2021.
2. Kajita, S., Kanehiro, F., Kaneko, K., Yokoi, K., and Hirukawa, H.: The 3d linear inverted pendulum mode: A simple modeling for a biped walking pattern generation. In International Conference on Intelligent Robots and Systems, Hawaii, USA, pages 239–246, 2001.
3. Grizzle, J. W. and Chevallereau, C.: Virtual constraints and hybrid zero dynamics for realizing underactuated bipedal locomotion. arXiv preprint arXiv:1706.01127, 2017.
4. Kuo, A. D.: Choosing your steps carefully. IEEE Robotics & Automation Magazine, 14(2):18–29, 2007.
5. Hobbelen, D. and Wisse, M.: Limit cycle walking. Humanoid Robots Human-like Machines, pages 277–294, 2007.
6. McGeer, T.: Passive dynamic walking. The International Journal of Robotics Research, 9(2):62–82, 1990.
7. Barnett, N. V. and Lammert, A. C.: Dynamic stability of passive dynamic walking following unexpected perturbations. Journal of biomechanical engineering, 145(4):044501, 2023.
8. Xiong, X. and Ames, A. D.: Orbit characterization, stabilization and composition on 3d underactuated bipedal walking via hybrid passive linear inverted pendulum model. In 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 4644–4651, 2019.
9. Xiong, X. and Ames, A.: 3d underactuated bipedal walking via h-lip based gait synthesis and stepping stabilization, 2021.
10. Dingwell, J. B. and Kang, H. G.: Differences between local and orbital dynamic stability during human walking. Journal of biomechanical engineering, 129(4):586–593, 2007.

11. Xie, Z., Berseth, G., Clary, P., Hurst, J., and van de Panne, M.: Feedback control for cassie with deep reinforcement learning. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1241–1246. IEEE, 2018.
12. Kuindersma, S., Deits, R., Fallon, M., Valenzuela, A., Dai, H., Permenter, F., Koolen, T., Marion, P., and Tedrake, R.: Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot. Autonomous robots, 40(3):429–455, 2016.
13. Collins, S. and Ruina, A.: A bipedal walking robot with efficient and human-like gait. In Proceeding of 2005 International Conference on Robotics and Automation, Barcelona, Spain, 2005.
14. Bhounsule, P. A., Cortell, J., Grewal, A., Hendriksen, B., Karszen, J. D., Paul, C., and Ruina, A.: Low-bandwidth reflex-based control for lower power walking: 65 km on a single battery charge. International Journal of Robotics Research, 2014.
15. Wisse, M. and Van der Linde, R. Q.: Delft pneumatic bipeds, volume 34. Springer Science & Business Media, 2007.
16. Flynn, L., Jafari, R., and Mukherjee, R.: Active synthetic-wheel biped with torso. IEEE Transactions on Robotics, 26(5):816–826, 2010.
17. Dertien, E.: Dynamic walking with dribbel. Robotics & Automation Magazine, IEEE, 13(3):118–122, 2006.
18. Dingwell, J. B., Kang, H. G., and Marin, L. C.: The effects of sensory loss and walking speed on the orbital dynamic stability of human walking. Journal of biomechanics, 40(8):1723–1730, 2007.
19. Strogatz, S.: Nonlinear dynamics and chaos. Addison-Wesley Reading, 1994.
20. Garcia, M., Chatterjee, A., Ruina, A., and Coleman, M.: The simplest walking model: Stability, complexity, and scaling. ASME J. of Biomech. Eng., 120:281–288, 1998.
21. Kuo, A. D.: Stabilization of lateral motion in passive dynamic walking. The International journal of robotics research, 18(9):917–930, 1999.

22. Bhounsule, P. A., Ruina, A., and Stiesberg, G.: Discrete-decision continuous-actuation control: balance of an inverted pendulum and pumping a pendulum swing. Journal of Dynamic Systems, Measurement, and Control, 137(5):051012, 2015.
23. Feng, S., Whitman, E., Xinjilefu, X., and Atkeson, C. G.: Optimization-based full body control for the darpa robotics challenge. Journal of Field Robotics, 32(2):293–312, 2015.
24. Grizzle, J., Abba, G., and Plestan, F.: Asymptotically stable walking for biped robots: Analysis via systems with impulse effects. IEEE Transactions on Automatic Control, 46(1):51–64, 2001.
25. Westervelt, E. R. and Grizzle, J. W.: Design of asymptotically stable walking for a 5-link planar biped walker via optimization. In Proceeding of 2005 International Conference on Robotics and Automation, Washington DC, USA, 2002.
26. Hereid, A., Cousineau, E. A., Hubicki, C. M., and Ames, A. D.: 3d dynamic walking with underactuated humanoid robots: A direct collocation framework for optimizing hybrid zero dynamics. In 2016 IEEE International Conference on Robotics and Automation (ICRA), pages 1447–1454. IEEE, 2016.
27. Chevallereau, C., Grizzle, J., and Shih, C.: Asymptotically stable walking of a five-link underactuated 3-d bipedal robot. IEEE Transactions on Robotics, 25(1):37–50, 2009.
28. Xiong, X. and Ames, A.: 3d underactuated bipedal walking via h-hip based gait synthesis and stepping stabilization. arXiv preprint arXiv:2101.09588, 2021.
29. Bhounsule, P. A., Kim, M., and Alaeddini, A.: Approximation of the step-to-step dynamics enables computationally efficient and fast optimal control of legged robots,. In ASME 2020 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. American Society of Mechanical Engineers, 2020.
30. Kuo, A.: Energetics of actively powered locomotion using the simplest walking model. Journal of Biomechanical Engineering, 124:113–120, 2002.
31. Kajita, S. and Tani, K.: Study of dynamic biped locomotion on rugged terrain-theory and basic experiment. In Proc. of the IEEE International Conference on Robotics and Automation, Sacramento, California, USA, pages 741–746, 1991.

32. Collins, S., Wisse, M., and Ruina, A.: A three-dimensional passive-dynamic walking robot with two legs and knees. The International Journal of Robotics Research, 20(7):607–615, 2001.
33. Westervelt, E., Grizzle, J., and Koditschek, D.: Hybrid zero dynamics of planar biped walkers. IEEE Transactions on Automatic Control, 48:42–56, 2003.
34. Seipel, J., Kvalheim, M., Revzen, S., Sharbafi, M. A., and Seyfarth, A.: Conceptual models of legged locomotion. In Bioinspired Legged Locomotion, pages 55–131. Elsevier, 2017.
35. Pratt, J., Carff, J., Drakunov, S., and Goswami, A.: Capture point: A step toward humanoid push recovery. In 2006 6th IEEE-RAS international conference on humanoid robots, pages 200–207. IEEE, 2006.
36. Hernandez-Hinojosa, E., Satici, A., and Bhounsule, P. A.: Optimal Control of a 5-Link Biped Using Quadratic Polynomial Model of Two-Point Boundary Value Problem. volume Volume 8B: 45th Mechanisms and Robotics Conference (MR) of International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, 08 2021.
37. Xiong, X. and Ames, A.: 3-d underactuated bipedal walking via h-lip based gait synthesis and stepping stabilization. IEEE Transactions on Robotics, 2022.
38. Castillo, G. A., Weng, B., Zhang, W., and Hereid, A.: Robust feedback motion policy design using reinforcement learning on a 3d digit bipedal robot. In 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 5136–5143. IEEE, 2021.
39. Zhou, P., Zanoni, A., and Masarati, P.: Projection continuation for minimal coordinate set dynamics of constrained systems. In ECCOMAS Thematic Conference on Multibody Dynamics, pages 184–196. Budapest University of Technology and Economics, 2021.
40. Hinojosa, E. H.: <https://youtu.be/mniabg2jgea>, Oct. 2022.
41. Hinojosa, E. H., Torres, D., and Bhounsule, P. A.: Quadratically constrained quadratic programs using approximations of the step-to-step dynamics: application on a 2d model of digit. In 2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids), pages 96–103, 2022.

42. Gao, Y., Gong, Y., Paredes, V., Hereid, A., and Gu, Y.: Time-varying alip model and robust foot-placement control for underactuated bipedal robotic walking on a swaying rigid surface. In 2023 American Control Conference (ACC), pages 3282–3287, 2023.
43. Hrr, J., Pratt, J., Chew, C.-M., Herr, H., and Pratt, G.: Adaptive virtual model control of a bipedal walking robot. In Proceedings. IEEE International Joint Symposia on Intelligence and Systems (Cat. No.98EX174), pages 245–251, 1998.
44. Paredes, V. C. and Hereid, A.: Resolved motion control for 3d underactuated bipedal walking using linear inverted pendulum dynamics and neural adaptation. In 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 6761–6767, 2022.
45. Ackermann, M. and Van den Bogert, A.: Optimality principles for model-based prediction of human gait. Journal of biomechanics, 43:1055–1060, 2010.
46. Hinojosa, E. H.: <https://youtu.be/bnp14d2bjci>, Oct. 2023.
47. Guan, Y., Sian, N., and Yokoi, K.: Motion planning for humanoid robots stepping over obstacles. In 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 363–369, 2005.
48. Yagi, M. and Lumelsky, V.: Biped robot locomotion in scenes with unknown obstacles. In Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C), volume 1, pages 375–380 vol.1, 1999.
49. Park, H.-W., Wensing, P. M., and Kim, S.: Jumping over obstacles with mit cheetah 2. Robotics and Autonomous Systems, 136:103703, 2021.
50. Bjelonic, M., Grandia, R., Geilinger, M., Harley, O., Medeiros, V. S., Pajovic, V., Jelavic, E., Coros, S., and Hutter, M.: Offline motion libraries and online mpc for advanced mobility skills. The International Journal of Robotics Research, 41(9-10):903–924, 2022.
51. Hennick, C.: Leaps, bounds, and backflips. Boston Dynamics, 2017.
52. Johnson, S. G.: The NLopt nonlinear-optimization package. <https://github.com/stevengj/nlopt>, 2007.

53. Tomar, N.: Socket-programming-server-in-c-and-client-in-python. <https://github.com/nikhilroxtomar/Socket-Programming-Server-in-C-and-Client-in-Python.git>.

Torres, D., Hernandez Hinojosa, E., Bhounsule, P. A., "Control of a Bipedal Walking Using Partial Feedback Linearization and Gaussian Process Regression-Based of the Step-to-Step Map", ASME-International Design Engineering & Technical Conference, Boston, MA, Aug 20-23, 2023.

Hernandez Hinojosa, E., Torres, D., & Bhounsule, P. A. (2022, November). Quadratically constrained quadratic programs using approximations of the step-to-step dynamics: application on a 2D model of Digit. In 2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids) (pp. 96-103). IEEE.

Echeveste, S., Hernandez-Hinojosa, E., & Bhounsule, P. A. (2021, October). Event-Based, Intermittent, Discrete Adaptive Control for Speed Regulation of Artificial Legs. In *Actuators* (Vol. 10, No. 10, p. 264). MDPI.

Hernandez-Hinojosa, E., Satici, A., & Bhounsule, P. A. (2021, August). Optimal Control of a 5-Link Biped Using Quadratic Polynomial Model of Two-Point Boundary Value Problem. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference* (Vol. 85451). American Society of Mechanical Engineers.

Bhounsule, P. A., Hernandez Hinojosa, E., Alaeddini, A., "One-Step Deadbeat Control of a 5-Link Biped Using Data-Driven Nonlinear Approximation of the Step-to-Step Dynamics". *MDPI Robotics* 9(4): 90, 2020.

Hernandez Hinojosa, E., Nawn, C. D., Abbott, C., Astorga, D., Jain, P., Restrepo, D., & Hood, R. L. (2020). "Bioinspired robotic exoskeleton for endotracheal intubation". *Journal of Materials Research and Technology*, 9(6), 12167-12176.

Hernandez Hinojosa, E., Warhmund, C., Lamoureux, K., Lee, E., Sanchez, I., Matthews, W., & Jafari, A. (2018). "A novel treadmill that can bilaterally adjust the vertical surface stiffness". *IEEE/ASME Transactions on Mechatronics*, 23(5), 2338-2346.

Guevara, J., Romo, J., Hernandez, E., & Guevara, N. V. (2018). Identification of receptor ligands in apo B100 reveals potential functional domains. *The protein journal*, 37, 548-571.

AWARDS

Graduate Student Award for Exceptional Research Promise, University of Illinois at Chicago, 2023

American Heart Association Predoctoral Research Fellowship, 2021

Graduate Student Professional Development Awards, University of Texas at San Antonio, 2019

Annual Innovation Awards, University of Texas at San Antonio, 2018